

# **The Open Master Hearing Aid (openMHA)**

**4.16.0**

## **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

<b>1 Overview</b>	<b>1</b>
1.1 Structure . . . . .	1
1.2 Platform Services and Conventions . . . . .	2
<b>2 Deprecated List</b>	<b>4</b>
<b>3 Module Documentation</b>	<b>4</b>
3.1 Concept of Variables and Data Exchange in the openMHA . . . . .	4
3.2 Writing openMHA Plugins. A step-by-step tutorial . . . . .	6
3.3 The MHA Framework interface . . . . .	22
3.4 Communication between algorithms . . . . .	23
3.5 Error handling in the openMHA . . . . .	28
3.6 The openMHA configuration language . . . . .	30
3.7 The openMHA Toolbox library . . . . .	31
3.8 Vector and matrix processing toolbox . . . . .	33
3.9 Complex arithmetics in the openMHA . . . . .	58
3.10 Fast Fourier Transform functions . . . . .	70
<b>4 Namespace Documentation</b>	<b>78</b>
4.1 ac2isl Namespace Reference . . . . .	78
4.2 ac_proc Namespace Reference . . . . .	78
4.3 acmon Namespace Reference . . . . .	79
4.4 acsave Namespace Reference . . . . .	79
4.5 addsndfile Namespace Reference . . . . .	79
4.6 ADM Namespace Reference . . . . .	81
4.7 audiometerbackend Namespace Reference . . . . .	82
4.8 AuditoryProfile Namespace Reference . . . . .	84
4.9 coherence Namespace Reference . . . . .	84
4.10 cpupload Namespace Reference . . . . .	85
4.11 dbasync_native Namespace Reference . . . . .	85
4.12 dc Namespace Reference . . . . .	86
4.13 dc_simple Namespace Reference . . . . .	86
4.14 delay Namespace Reference . . . . .	89
4.15 delaysum Namespace Reference . . . . .	89
4.16 delaysum_spec Namespace Reference . . . . .	89
4.17 double2acvar Namespace Reference . . . . .	90
4.18 DynComp Namespace Reference . . . . .	90
4.19 equalize Namespace Reference . . . . .	91
4.20 fader_wave Namespace Reference . . . . .	92
4.21 fftfbpow Namespace Reference . . . . .	92
4.22 fftfilter Namespace Reference . . . . .	92
4.23 fftfilterbank Namespace Reference . . . . .	94
4.24 fshift Namespace Reference . . . . .	94
4.25 fshift_hilbert Namespace Reference . . . . .	95
4.26 gain Namespace Reference . . . . .	95
4.27 gsc_adaptive_stage Namespace Reference . . . . .	95
4.28 gtfb_analyzer Namespace Reference . . . . .	96
4.29 level_matching Namespace Reference . . . . .	96
4.30 Isl2ac Namespace Reference . . . . .	96
4.31 matlab_wrapper Namespace Reference . . . . .	97

4.32	matrixmixer Namespace Reference . . . . .	98
4.33	mconv Namespace Reference . . . . .	98
4.34	MHA_AC Namespace Reference . . . . .	98
4.35	mha_error_helpers Namespace Reference . . . . .	99
4.36	MHA_TCP Namespace Reference . . . . .	100
4.37	mha_tcp Namespace Reference . . . . .	103
4.38	mhachain Namespace Reference . . . . .	104
4.39	MHAEvents Namespace Reference . . . . .	104
4.40	MHAFilter Namespace Reference . . . . .	104
4.41	MHAIOJack Namespace Reference . . . . .	109
4.42	MHAIOJackdb Namespace Reference . . . . .	110
4.43	MHAIOPortAudio Namespace Reference . . . . .	110
4.44	mhaioutils Namespace Reference . . . . .	111
4.45	MHAJack Namespace Reference . . . . .	111
4.46	MHAKernel Namespace Reference . . . . .	114
4.47	MHAMultiSrc Namespace Reference . . . . .	115
4.48	MHAOvlFilter Namespace Reference . . . . .	115
4.49	MHAOvlFilter::barkscale Namespace Reference . . . . .	116
4.50	MHAOvlFilter::FreqScaleFun Namespace Reference . . . . .	117
4.51	MHAOvlFilter::ShapeFun Namespace Reference . . . . .	120
4.52	MHAParser Namespace Reference . . . . .	122
4.53	MHAParser::StrCnv Namespace Reference . . . . .	128
4.54	MHAPlugin Namespace Reference . . . . .	134
4.55	MHAPlugin_Resampling Namespace Reference . . . . .	134
4.56	MHAPlugin_Split Namespace Reference . . . . .	135
4.57	MHASignal Namespace Reference . . . . .	136
4.58	MHASndFile Namespace Reference . . . . .	150
4.59	MHATableLookup Namespace Reference . . . . .	150
4.60	MHAUtils Namespace Reference . . . . .	150
4.61	MHAWindow Namespace Reference . . . . .	153
4.62	multibandcompressor Namespace Reference . . . . .	155
4.63	noise_psd_estimator Namespace Reference . . . . .	155
4.64	overlapadd Namespace Reference . . . . .	156
4.65	plingploing Namespace Reference . . . . .	156
4.66	PluginLoader Namespace Reference . . . . .	157
4.67	plugins Namespace Reference . . . . .	158
4.68	plugins::hoertech Namespace Reference . . . . .	158
4.69	plugins::hoertech::acrec Namespace Reference . . . . .	158
4.70	rmslevel Namespace Reference . . . . .	158
4.71	rohBeam Namespace Reference . . . . .	159
4.72	route Namespace Reference . . . . .	160
4.73	shadowfilter_begin Namespace Reference . . . . .	160
4.74	shadowfilter_end Namespace Reference . . . . .	161
4.75	smooth_cepstrum Namespace Reference . . . . .	161
4.76	smoothgains_bridge Namespace Reference . . . . .	161
4.77	testplugin Namespace Reference . . . . .	161
4.78	windnoise Namespace Reference . . . . .	161
<b>5</b>	<b>Class Documentation</b>	<b>162</b>
5.1	ac2isl::ac2isl_t Class Reference . . . . .	162
5.2	ac2isl::cfg_t Class Reference . . . . .	166
5.3	ac2isl::save_var_base_t Class Reference . . . . .	169

5.4	ac2isl::save_var_t< T > Class Template Reference . . . . .	171
5.5	ac2isl::save_var_t< mha_complex_t > Class Reference . . . . .	175
5.6	ac2isl::type_info Struct Reference . . . . .	179
5.7	ac2osc_t Class Reference . . . . .	180
5.8	ac2wave_if_t Class Reference . . . . .	185
5.9	ac2wave_t Class Reference . . . . .	188
5.10	ac_mul_t Class Reference . . . . .	191
5.11	ac_proc::interface_t Class Reference . . . . .	196
5.12	acConcat_wave Class Reference . . . . .	199
5.13	acConcat_wave_config Class Reference . . . . .	202
5.14	acmon::ac_monitor_t Class Reference . . . . .	203
5.15	acmon::acmon_t Class Reference . . . . .	207
5.16	acPooling_wave Class Reference . . . . .	211
5.17	acPooling_wave_config Class Reference . . . . .	215
5.18	acsave::acsave_t Class Reference . . . . .	218
5.19	acsave::cfg_t Class Reference . . . . .	222
5.20	acsave::mat4head_t Struct Reference . . . . .	224
5.21	acsave::save_var_t Class Reference . . . . .	225
5.22	acSteer Class Reference . . . . .	228
5.23	acSteer_config Class Reference . . . . .	231
5.24	acTransform_wave Class Reference . . . . .	233
5.25	acTransform_wave_config Class Reference . . . . .	237
5.26	adaptive_feedback_canceller Class Reference . . . . .	240
5.27	adaptive_feedback_canceller_config Class Reference . . . . .	244
5.28	addsndfile::addsndfile_if_t Class Reference . . . . .	250
5.29	addsndfile::level_adapt_t Class Reference . . . . .	255
5.30	addsndfile::resampled_soundfile_t Class Reference . . . . .	257
5.31	addsndfile::sndfile_t Class Reference . . . . .	259
5.32	addsndfile::waveform_proxy_t Class Reference . . . . .	260
5.33	ADM::ADM< F > Class Template Reference . . . . .	261
5.34	ADM::Delay< F > Class Template Reference . . . . .	265
5.35	ADM::Linearphase_FIR< F > Class Template Reference . . . . .	268
5.36	adm_if_t Class Reference . . . . .	271
5.37	adm_rtconfig_t Class Reference . . . . .	275
5.38	algo_comm_t Struct Reference . . . . .	279
5.39	alsa_base_t Class Reference . . . . .	286
5.40	alsa_dev_par_parser_t Class Reference . . . . .	289
5.41	alsa_t< T > Class Template Reference . . . . .	291
5.42	altconfig_t Class Reference . . . . .	295
5.43	altplugs_t Class Reference . . . . .	300
5.44	analysepath_t Class Reference . . . . .	306
5.45	analysispath_if_t Class Reference . . . . .	310
5.46	attenuate20_t Class Reference . . . . .	313
5.47	audiometerbackend::audiometer_if_t Class Reference . . . . .	315
5.48	audiometerbackend::level_adapt_t Class Reference . . . . .	318
5.49	audiometerbackend::Inn3rdoct_t Class Reference . . . . .	320
5.50	audiometerbackend::signal_gen_t Class Reference . . . . .	322
5.51	audiometerbackend::sine_t Class Reference . . . . .	323
5.52	AuditoryProfile::fmap_t Class Reference . . . . .	324
5.53	AuditoryProfile::parser_t Class Reference . . . . .	325
5.54	AuditoryProfile::parser_t::ear_t Class Reference . . . . .	327

5.55	AuditoryProfile::parser_t::fmap_t Class Reference . . . . .	329
5.56	AuditoryProfile::profile_t Class Reference . . . . .	331
5.57	AuditoryProfile::profile_t::ear_t Class Reference . . . . .	332
5.58	bccalib_interface_t Class Reference . . . . .	333
5.59	calibrator_runtime_layer_t Class Reference . . . . .	335
5.60	calibrator_t Class Reference . . . . .	338
5.61	calibrator_variables_t Class Reference . . . . .	341
5.62	cfg_t Class Reference . . . . .	343
5.63	coherence::cohflt_if_t Class Reference . . . . .	347
5.64	coherence::cohflt_t Class Reference . . . . .	349
5.65	coherence::vars_t Class Reference . . . . .	353
5.66	combc_if_t Class Reference . . . . .	355
5.67	combc_t Class Reference . . . . .	357
5.68	comm_var_t Struct Reference . . . . .	359
5.69	complex_scale_channel_t Class Reference . . . . .	361
5.70	cpupload::cpupload_cfg_t Class Reference . . . . .	363
5.71	cpupload::cpupload_if_t Class Reference . . . . .	366
5.72	db_if_t Class Reference . . . . .	369
5.73	db_t Class Reference . . . . .	371
5.74	dbasync_native::db_if_t Class Reference . . . . .	373
5.75	dbasync_native::dbasync_t Class Reference . . . . .	376
5.76	dbasync_native::delay_check_t Class Reference . . . . .	379
5.77	dc::dc_if_t Class Reference . . . . .	380
5.78	dc::dc_t Class Reference . . . . .	382
5.79	dc::dc_vars_t Class Reference . . . . .	387
5.80	dc::dc_vars_validator_t Class Reference . . . . .	391
5.81	dc_simple::dc_if_t Class Reference . . . . .	392
5.82	dc_simple::dc_t Class Reference . . . . .	396
5.83	dc_simple::dc_t::line_t Class Reference . . . . .	399
5.84	dc_simple::dc_vars_t Class Reference . . . . .	400
5.85	dc_simple::dc_vars_validator_t Class Reference . . . . .	403
5.86	dc_simple::level_smoothen_t Class Reference . . . . .	404
5.87	delay::interface_t Class Reference . . . . .	407
5.88	delaysum::delaysum_wave_if_t Class Reference . . . . .	409
5.89	delaysum::delaysum_wave_t Class Reference . . . . .	412
5.90	delaysum_spec::delaysum_spec_if_t Class Reference . . . . .	414
5.91	delaysum_spec::delaysum_t Class Reference . . . . .	416
5.92	doasvm_classification Class Reference . . . . .	417
5.93	doasvm_classification_config Class Reference . . . . .	421
5.94	doasvm_feature_extraction Class Reference . . . . .	423
5.95	doasvm_feature_extraction_config Class Reference . . . . .	426
5.96	double2acvar::double2acvar_t Class Reference . . . . .	429
5.97	dropgen_t Class Reference . . . . .	433
5.98	droptect_t Class Reference . . . . .	436
5.99	ds_t Class Reference . . . . .	440
5.100	dynamiclib_t Class Reference . . . . .	442
5.101	DynComp::dc_afterburn_rt_t Class Reference . . . . .	446
5.102	DynComp::dc_afterburn_t Class Reference . . . . .	449
5.103	DynComp::dc_afterburn_vars_t Class Reference . . . . .	452
5.104	DynComp::gaintable_t Class Reference . . . . .	454
5.105	equalize::cfg_t Class Reference . . . . .	459

5.106 equalize::freqgains_t Class Reference . . . . .	460
5.107 example1_t Class Reference . . . . .	463
5.108 example2_t Class Reference . . . . .	465
5.109 example3_t Class Reference . . . . .	469
5.110 example4_t Class Reference . . . . .	473
5.111 example5_t Class Reference . . . . .	477
5.112 example6_t Class Reference . . . . .	478
5.113 example7_t Class Reference . . . . .	481
5.114 expression_t Class Reference . . . . .	482
5.115 fader_if_t Class Reference . . . . .	483
5.116 fader_wave::fader_wave_if_t Class Reference . . . . .	485
5.117 fader_wave::level_adapt_t Class Reference . . . . .	488
5.118 fftfbpow::fftfbpow_interface_t Class Reference . . . . .	490
5.119 fftfbpow::fftfbpow_t Class Reference . . . . .	493
5.120 fftfilter::fftfilter_t Class Reference . . . . .	495
5.121 fftfilter::interface_t Class Reference . . . . .	497
5.122 fftfilterbank::fftfb_interface_t Class Reference . . . . .	500
5.123 fftfilterbank::fftfb_plug_t Class Reference . . . . .	503
5.124 fshift::fshift_config_t Class Reference . . . . .	505
5.125 fshift::fshift_t Class Reference . . . . .	508
5.126 fshift_hilbert::frequency_translator_t Class Reference . . . . .	511
5.127 fshift_hilbert::hilbert_shifter_t Class Reference . . . . .	514
5.128 fw_t Class Reference . . . . .	519
5.129 fw_vars_t Class Reference . . . . .	528
5.130 gain::gain_if_t Class Reference . . . . .	529
5.131 gain::scaler_t Class Reference . . . . .	532
5.132 gsc_adaptive_stage::gsc_adaptive_stage Class Reference . . . . .	533
5.133 gsc_adaptive_stage::gsc_adaptive_stage_if Class Reference . . . . .	541
5.134 gtfb_analyzer::gtfb_analyzer_cfg_t Struct Reference . . . . .	545
5.135 gtfb_analyzer::gtfb_analyzer_t Class Reference . . . . .	549
5.136 gtfb_simd_cfg_t Class Reference . . . . .	552
5.137 gtfb_simd_t Class Reference . . . . .	558
5.138 gtfb_simple_rt_t Class Reference . . . . .	560
5.139 gtfb_simple_t Class Reference . . . . .	566
5.140 hanning_ramps_t Class Reference . . . . .	570
5.141 identity_t Class Reference . . . . .	572
5.142 iirfilter_t Class Reference . . . . .	574
5.143 io_alsa_t Class Reference . . . . .	575
5.144 io_asterisk_fwc_b_t Class Reference . . . . .	580
5.145 io_asterisk_parser_t Class Reference . . . . .	585
5.146 io_asterisk_sound_t Class Reference . . . . .	592
5.147 io_asterisk_t Class Reference . . . . .	596
5.148 io_file_t Class Reference . . . . .	601
5.149 io_lib_t Class Reference . . . . .	607
5.150 io_parser_t Class Reference . . . . .	612
5.151 io_tcp_fwc_b_t Class Reference . . . . .	617
5.152 io_tcp_parser_t Class Reference . . . . .	622
5.153 io_tcp_sound_t Class Reference . . . . .	629
5.154 io_tcp_sound_t::float_union Union Reference . . . . .	634
5.155 io_tcp_t Class Reference . . . . .	634
5.156 io_wrapper Class Reference . . . . .	639

5.157 <code>latex_doc_t</code> Class Reference . . . . .	641
5.158 <code>level_matching::channel_pair</code> Class Reference . . . . .	644
5.159 <code>level_matching::level_matching_config_t</code> Class Reference . . . . .	647
5.160 <code>level_matching::level_matching_t</code> Class Reference . . . . .	650
5.161 <code>levelmeter_t</code> Class Reference . . . . .	654
5.162 <code>lpc</code> Class Reference . . . . .	657
5.163 <code>lpc_bl_predictor</code> Class Reference . . . . .	660
5.164 <code>lpc_bl_predictor_config</code> Class Reference . . . . .	664
5.165 <code>lpc_burglattice</code> Class Reference . . . . .	666
5.166 <code>lpc_burglattice_config</code> Class Reference . . . . .	669
5.167 <code>lpc_config</code> Class Reference . . . . .	672
5.168 <code>lsl2ac::cfg_t</code> Class Reference . . . . .	675
5.169 <code>lsl2ac::lsl2ac_t</code> Class Reference . . . . .	677
5.170 <code>lsl2ac::save_var_t</code> Class Reference . . . . .	682
5.171 <code>matlab_wrapper::callback</code> Class Reference . . . . .	689
5.172 <code>matlab_wrapper::matlab_wrapper_rt_cfg_t</code> Class Reference . . . . .	691
5.173 <code>matlab_wrapper::matlab_wrapper_t</code> Class Reference . . . . .	693
5.174 <code>matlab_wrapper::matlab_wrapper_t::wrapped_plugin_t</code> Class Reference . . . . .	699
5.175 <code>matlab_wrapper::types&lt; T &gt;</code> Struct Template Reference . . . . .	707
5.176 <code>matlab_wrapper::types&lt; MHA_SPECTRUM &gt;</code> Struct Reference . . . . .	707
5.177 <code>matlab_wrapper::types&lt; MHA_WAVEFORM &gt;</code> Struct Reference . . . . .	708
5.178 <code>matrixmixer::cfg_t</code> Class Reference . . . . .	709
5.179 <code>matrixmixer::matmix_t</code> Class Reference . . . . .	710
5.180 <code>mconv::MConv</code> Class Reference . . . . .	713
5.181 <code>MHA_AC::ac2matrix_helper_t</code> Class Reference . . . . .	717
5.182 <code>MHA_AC::ac2matrix_t</code> Class Reference . . . . .	719
5.183 <code>MHA_AC::acspace2matrix_t</code> Class Reference . . . . .	721
5.184 <code>MHA_AC::double_t</code> Class Reference . . . . .	725
5.185 <code>MHA_AC::float_t</code> Class Reference . . . . .	727
5.186 <code>MHA_AC::int_t</code> Class Reference . . . . .	729
5.187 <code>MHA_AC::spectrum_t</code> Class Reference . . . . .	731
5.188 <code>MHA_AC::stat_t</code> Class Reference . . . . .	733
5.189 <code>MHA_AC::waveform_t</code> Class Reference . . . . .	735
5.190 <code>mha_audio_descriptor_t</code> Struct Reference . . . . .	737
5.191 <code>mha_audio_t</code> Struct Reference . . . . .	739
5.192 <code>mha_channel_info_t</code> Struct Reference . . . . .	740
5.193 <code>mha_complex_t</code> Struct Reference . . . . .	741
5.194 <code>mha_complex_test_array_t</code> Struct Reference . . . . .	742
5.195 <code>mha_dblbuf_t&lt; FIFO &gt;</code> Class Template Reference . . . . .	743
5.196 <code>mha_direction_t</code> Struct Reference . . . . .	751
5.197 <code>mha_drifter_fifo_t&lt; T &gt;</code> Class Template Reference . . . . .	752
5.198 <code>MHA_Error</code> Class Reference . . . . .	760
5.199 <code>mha_fifo_if_t&lt; T &gt;</code> Class Template Reference . . . . .	763
5.200 <code>mha_fifo_lw_t&lt; T &gt;</code> Class Template Reference . . . . .	766
5.201 <code>mha_fifo_posix_threads_t</code> Class Reference . . . . .	770
5.202 <code>mha_fifo_t&lt; T &gt;</code> Class Template Reference . . . . .	772
5.203 <code>mha_fifo_thread_guard_t</code> Class Reference . . . . .	779
5.204 <code>mha_fifo_thread_platform_t</code> Class Reference . . . . .	780
5.205 <code>mha_real_test_array_t</code> Struct Reference . . . . .	784
5.206 <code>mha_rt_fifo_element_t&lt; T &gt;</code> Class Template Reference . . . . .	784
5.207 <code>mha_rt_fifo_t&lt; T &gt;</code> Class Template Reference . . . . .	786

5.208 mha_spec_t Struct Reference . . . . .	790
5.209 mha_stash_environment_variable_t Class Reference . . . . .	792
5.210 MHA_TCP::Async_Notify Class Reference . . . . .	794
5.211 mha_tcp::buffered_socket_t Class Reference . . . . .	795
5.212 MHA_TCP::Client Class Reference . . . . .	797
5.213 MHA_TCP::Connection Class Reference . . . . .	799
5.214 MHA_TCP::Event_Watcher Class Reference . . . . .	807
5.215 MHA_TCP::OS_EVENT_TYPE Struct Reference . . . . .	810
5.216 MHA_TCP::Server Class Reference . . . . .	811
5.217 mha_tcp::server_t Class Reference . . . . .	815
5.218 MHA_TCP::sock_init_t Class Reference . . . . .	821
5.219 MHA_TCP::Sockaccept_Event Class Reference . . . . .	821
5.220 MHA_TCP::Sockread_Event Class Reference . . . . .	822
5.221 MHA_TCP::Sockwrite_Event Class Reference . . . . .	823
5.222 MHA_TCP::Thread Class Reference . . . . .	824
5.223 MHA_TCP::Timeout_Event Class Reference . . . . .	829
5.224 MHA_TCP::Timeout_Watcher Class Reference . . . . .	830
5.225 MHA_TCP::Wakeup_Event Class Reference . . . . .	832
5.226 mha_tictoc_t Struct Reference . . . . .	835
5.227 mha_wave_t Struct Reference . . . . .	836
5.228 mhachain::chain_base_t Class Reference . . . . .	838
5.229 mhachain::mhachain_t Class Reference . . . . .	842
5.230 mhachain::plugs_t Class Reference . . . . .	843
5.231 mhaconfig_t Struct Reference . . . . .	847
5.232 MHAEvents::connector_base_t Class Reference . . . . .	849
5.233 MHAEvents::connector_t< receiver_t > Class Template Reference . . . . .	852
5.234 MHAEvents::emitter_t Class Reference . . . . .	855
5.235 MHAEvents::patchbay_t< receiver_t > Class Template Reference . . . . .	857
5.236 MHAFilter::adapt_filter_param_t Class Reference . . . . .	859
5.237 MHAFilter::adapt_filter_state_t Class Reference . . . . .	860
5.238 MHAFilter::adapt_filter_t Class Reference . . . . .	862
5.239 MHAFilter::blockprocessing_polyphase_resampling_t Class Reference . . . . .	864
5.240 MHAFilter::complex_bandpass_t Class Reference . . . . .	868
5.241 MHAFilter::diff_t Class Reference . . . . .	872
5.242 MHAFilter::fftfilter_t Class Reference . . . . .	873
5.243 MHAFilter::fftfilterbank_t Class Reference . . . . .	878
5.244 MHAFilter::filter_t Class Reference . . . . .	883
5.245 MHAFilter::gamma_flt_t Class Reference . . . . .	888
5.246 MHAFilter::iir_filter_state_t Class Reference . . . . .	893
5.247 MHAFilter::iir_filter_t Class Reference . . . . .	894
5.248 MHAFilter::iir_ord1_real_t Class Reference . . . . .	898
5.249 MHAFilter::o1_ar_filter_t Class Reference . . . . .	902
5.250 MHAFilter::o1flt_lowpass_t Class Reference . . . . .	906
5.251 MHAFilter::o1flt_maxtrack_t Class Reference . . . . .	909
5.252 MHAFilter::o1flt_mintrack_t Class Reference . . . . .	911
5.253 MHAFilter::partitioned_convolution_t Class Reference . . . . .	913
5.254 MHAFilter::partitioned_convolution_t::index_t Struct Reference . . . . .	917
5.255 MHAFilter::polyphase_resampling_t Class Reference . . . . .	919
5.256 MHAFilter::resampling_filter_t Class Reference . . . . .	924
5.257 MHAFilter::smoothspec_t Class Reference . . . . .	926
5.258 MHAFilter::thirdoctave_analyzer_t Class Reference . . . . .	930

5.259 MHAFilter::transfer_function_t Struct Reference . . . . .	933
5.260 MHAFilter::transfer_matrix_t Struct Reference . . . . .	936
5.261 MHAIOJack::io_jack_t Class Reference . . . . .	937
5.262 MHAIOJackdb::io_jack_t Class Reference . . . . .	944
5.263 MHAIOPortAudio::device_info_t Class Reference . . . . .	952
5.264 MHAIOPortAudio::io_portaudio_t Class Reference . . . . .	955
5.265 MHAIOPortAudio::stream_info_t Class Reference . . . . .	961
5.266 MHAJack::client_avg_t Class Reference . . . . .	963
5.267 MHAJack::client_noncont_t Class Reference . . . . .	967
5.268 MHAJack::client_t Class Reference . . . . .	970
5.269 MHAJack::port_t Class Reference . . . . .	980
5.270 MHAKernel::algo_comm_class_t Class Reference . . . . .	984
5.271 MHAKernel::comm_var_map_t Class Reference . . . . .	990
5.272 MHAMultiSrc::base_t Class Reference . . . . .	991
5.273 MHAMultiSrc::channel_t Class Reference . . . . .	993
5.274 MHAMultiSrc::channels_t Class Reference . . . . .	993
5.275 MHAMultiSrc::spectrum_t Class Reference . . . . .	994
5.276 MHAMultiSrc::waveform_t Class Reference . . . . .	996
5.277 MHAOvIFilter::band_descriptor_t Class Reference . . . . .	997
5.278 MHAOvIFilter::barkscale::bark2hz_t Class Reference . . . . .	999
5.279 MHAOvIFilter::barkscale::hz2bark_t Class Reference . . . . .	1000
5.280 MHAOvIFilter::fftfb_ac_info_t Class Reference . . . . .	1001
5.281 MHAOvIFilter::fftfb_t Class Reference . . . . .	1002
5.282 MHAOvIFilter::fftfb_vars_t Class Reference . . . . .	1006
5.283 MHAOvIFilter::fscale_bw_t Class Reference . . . . .	1010
5.284 MHAOvIFilter::fscale_t Class Reference . . . . .	1012
5.285 MHAOvIFilter::fspacing_t Class Reference . . . . .	1014
5.286 MHAOvIFilter::overlap_save_filterbank_analytic_t Class Reference . . . . .	1018
5.287 MHAOvIFilter::overlap_save_filterbank_t Class Reference . . . . .	1019
5.288 MHAOvIFilter::overlap_save_filterbank_t::vars_t Class Reference . . . . .	1022
5.289 MHAOvIFilter::scale_var_t Class Reference . . . . .	1024
5.290 MHAParser::base_t Class Reference . . . . .	1026
5.291 MHAParser::base_t::replace_t Class Reference . . . . .	1040
5.292 MHAParser::bool_mon_t Class Reference . . . . .	1041
5.293 MHAParser::bool_t Class Reference . . . . .	1043
5.294 MHAParser::c_ifc_parser_t Class Reference . . . . .	1046
5.295 MHAParser::commit_t< receiver_t > Class Template Reference . . . . .	1049
5.296 MHAParser::complex_mon_t Class Reference . . . . .	1051
5.297 MHAParser::complex_t Class Reference . . . . .	1053
5.298 MHAParser::entry_t Class Reference . . . . .	1055
5.299 MHAParser::expression_t Class Reference . . . . .	1056
5.300 MHAParser::float_mon_t Class Reference . . . . .	1057
5.301 MHAParser::float_t Class Reference . . . . .	1059
5.302 MHAParser::int_mon_t Class Reference . . . . .	1062
5.303 MHAParser::int_t Class Reference . . . . .	1065
5.304 MHAParser::keyword_list_t Class Reference . . . . .	1067
5.305 MHAParser::kw_t Class Reference . . . . .	1071
5.306 MHAParser::mcomplex_mon_t Class Reference . . . . .	1075
5.307 MHAParser::mcomplex_t Class Reference . . . . .	1077
5.308 MHAParser::mfloat_mon_t Class Reference . . . . .	1079
5.309 MHAParser::mfloat_t Class Reference . . . . .	1081

5.310 MHAParser::mhaconfig_mon_t Class Reference . . . . .	1084
5.311 MHAParser::mhapluginloader_t Class Reference . . . . .	1086
5.312 MHAParser::mint_mon_t Class Reference . . . . .	1091
5.313 MHAParser::mint_t Class Reference . . . . .	1093
5.314 MHAParser::monitor_t Class Reference . . . . .	1096
5.315 MHAParser::parser_t Class Reference . . . . .	1098
5.316 MHAParser::range_var_t Class Reference . . . . .	1104
5.317 MHAParser::string_mon_t Class Reference . . . . .	1109
5.318 MHAParser::string_t Class Reference . . . . .	1111
5.319 MHAParser::variable_t Class Reference . . . . .	1114
5.320 MHAParser::vcomplex_mon_t Class Reference . . . . .	1116
5.321 MHAParser::vcomplex_t Class Reference . . . . .	1118
5.322 MHAParser::vfloat_mon_t Class Reference . . . . .	1121
5.323 MHAParser::vfloat_t Class Reference . . . . .	1123
5.324 MHAParser::vint_mon_t Class Reference . . . . .	1126
5.325 MHAParser::vint_t Class Reference . . . . .	1128
5.326 MHAParser::vstring_mon_t Class Reference . . . . .	1130
5.327 MHAParser::vstring_t Class Reference . . . . .	1132
5.328 MHAParser::window_t Class Reference . . . . .	1134
5.329 mhaplug_cfg_t Class Reference . . . . .	1138
5.330 MHAPlugin::cfg_node_t< runtime_cfg_t > Class Template Reference . . . . .	1139
5.331 MHAPlugin::config_t< runtime_cfg_t > Class Template Reference . . . . .	1142
5.332 MHAPlugin::plugin_t< runtime_cfg_t > Class Template Reference . . . . .	1147
5.333 MHAPlugin_Resampling::resampling_if_t Class Reference . . . . .	1153
5.334 MHAPlugin_Resampling::resampling_t Class Reference . . . . .	1156
5.335 MHAPlugin_Split::domain_handler_t Class Reference . . . . .	1159
5.336 MHAPlugin_Split::dummy_threads_t Class Reference . . . . .	1165
5.337 MHAPlugin_Split::posix_threads_t Class Reference . . . . .	1167
5.338 MHAPlugin_Split::split_t Class Reference . . . . .	1172
5.339 MHAPlugin_Split::splitted_part_t Class Reference . . . . .	1179
5.340 MHAPlugin_Split::thread_platform_t Class Reference . . . . .	1184
5.341 MHAPlugin_Split::uni_processor_t Class Reference . . . . .	1187
5.342 mhaserver_t Class Reference . . . . .	1189
5.343 mhaserver_t::tcp_server_t Class Reference . . . . .	1193
5.344 MHASignal::async_rmslevel_t Class Reference . . . . .	1195
5.345 MHASignal::delay_spec_t Class Reference . . . . .	1198
5.346 MHASignal::delay_t Class Reference . . . . .	1199
5.347 MHASignal::delay_wave_t Class Reference . . . . .	1202
5.348 MHASignal::doublebuffer_t Class Reference . . . . .	1203
5.349 MHASignal::fft_t Class Reference . . . . .	1207
5.350 MHASignal::hilbert_fftw_t Class Reference . . . . .	1212
5.351 MHASignal::hilbert_t Class Reference . . . . .	1214
5.352 MHASignal::loop_wavefragment_t Class Reference . . . . .	1216
5.353 MHASignal::matrix_t Class Reference . . . . .	1221
5.354 MHASignal::minphase_t Class Reference . . . . .	1233
5.355 MHASignal::quantizer_t Class Reference . . . . .	1234
5.356 MHASignal::ringbuffer_t Class Reference . . . . .	1236
5.357 MHASignal::schroeder_t Class Reference . . . . .	1240
5.358 MHASignal::spectrum_t Class Reference . . . . .	1244
5.359 MHASignal::stat_t Class Reference . . . . .	1250
5.360 MHASignal::subsample_delay_t Class Reference . . . . .	1252

5.361 MHASignal::uint_vector_t Class Reference . . . . .	1255
5.362 MHASignal::waveform_t Class Reference . . . . .	1259
5.363 MHASndFile::sf_t Class Reference . . . . .	1272
5.364 MHASndFile::sf_wave_t Class Reference . . . . .	1273
5.365 MHATableLookup::linear_table_t Class Reference . . . . .	1274
5.366 MHATableLookup::table_t Class Reference . . . . .	1279
5.367 MHATableLookup::xy_table_t Class Reference . . . . .	1281
5.368 MHAWindow::bartlett_t Class Reference . . . . .	1286
5.369 MHAWindow::base_t Class Reference . . . . .	1287
5.370 MHAWindow::blackman_t Class Reference . . . . .	1289
5.371 MHAWindow::fun_t Class Reference . . . . .	1291
5.372 MHAWindow::hamming_t Class Reference . . . . .	1292
5.373 MHAWindow::hanning_t Class Reference . . . . .	1293
5.374 MHAWindow::rect_t Class Reference . . . . .	1295
5.375 MHAWindow::user_t Class Reference . . . . .	1296
5.376 multibandcompressor::fffb_plug_t Class Reference . . . . .	1298
5.377 multibandcompressor::interface_t Class Reference . . . . .	1300
5.378 multibandcompressor::plugin_signals_t Class Reference . . . . .	1303
5.379 nlms_t Class Reference . . . . .	1305
5.380 noise_psd_estimator::noise_psd_estimator_if_t Class Reference . . . . .	1308
5.381 noise_psd_estimator::noise_psd_estimator_t Class Reference . . . . .	1311
5.382 noise_t Class Reference . . . . .	1314
5.383 osc2ac_t Class Reference . . . . .	1317
5.384 osc_server_t Class Reference . . . . .	1320
5.385 osc_variable_t Class Reference . . . . .	1322
5.386 overlapadd::overlapadd_if_t Class Reference . . . . .	1326
5.387 overlapadd::overlapadd_t Class Reference . . . . .	1331
5.388 parser_int_dyn Class Reference . . . . .	1334
5.389 plingploing::if_t Class Reference . . . . .	1336
5.390 plingploing::plingploing_t Class Reference . . . . .	1340
5.391 plug_t Class Reference . . . . .	1345
5.392 plug_wrapper Class Reference . . . . .	1346
5.393 plug_wrapperl Class Reference . . . . .	1348
5.394 plugin_interface_t Class Reference . . . . .	1350
5.395 pluginbrowser_t Class Reference . . . . .	1352
5.396 plugindescription_t Class Reference . . . . .	1355
5.397 pluginlib_t Class Reference . . . . .	1356
5.398 PluginLoader::config_file_splitter_t Class Reference . . . . .	1358
5.399 PluginLoader::fourway_processor_t Class Reference . . . . .	1361
5.400 PluginLoader::mhapluginloader_t Class Reference . . . . .	1364
5.401 pluginloader_t Class Reference . . . . .	1372
5.402 plugins::hoertech::acrec::acrec_t Class Reference . . . . .	1373
5.403 plugins::hoertech::acrec::acwriter_t Class Reference . . . . .	1378
5.404 proc_counter_t Class Reference . . . . .	1384
5.405 rmslevel::mon_t Class Reference . . . . .	1386
5.406 rmslevel::rmslevel_if_t Class Reference . . . . .	1388
5.407 rmslevel::rmslevel_t Class Reference . . . . .	1390
5.408 rohBeam::configOptions Struct Reference . . . . .	1393
5.409 rohBeam::rohBeam Class Reference . . . . .	1395
5.410 rohBeam::rohConfig Class Reference . . . . .	1402
5.411 route::interface_t Class Reference . . . . .	1408

5.412 route::process_t Class Reference . . . . .	1411
5.413 rt_nlms_t Class Reference . . . . .	1413
5.414 save_spec_t Class Reference . . . . .	1418
5.415 save_wave_t Class Reference . . . . .	1420
5.416 shadowfilter_begin::cfg_t Class Reference . . . . .	1421
5.417 shadowfilter_begin::shadowfilter_begin_t Class Reference . . . . .	1423
5.418 shadowfilter_end::cfg_t Class Reference . . . . .	1425
5.419 shadowfilter_end::shadowfilter_end_t Class Reference . . . . .	1427
5.420 sine_cfg_t Struct Reference . . . . .	1429
5.421 sine_t Class Reference . . . . .	1431
5.422 smooth_cepstrum::smooth_cepstrum_if_t Class Reference . . . . .	1434
5.423 smooth_cepstrum::smooth_cepstrum_t Class Reference . . . . .	1439
5.424 smooth_cepstrum::smooth_params Class Reference . . . . .	1446
5.425 smoothgains_bridge::overlapadd_if_t Class Reference . . . . .	1449
5.426 smoothgains_bridge::smoothspec_wrap_t Class Reference . . . . .	1452
5.427 softclip_t Class Reference . . . . .	1454
5.428 softclipper_t Class Reference . . . . .	1456
5.429 softclipper_variables_t Class Reference . . . . .	1458
5.430 spec2wave_if_t Class Reference . . . . .	1461
5.431 spec2wave_t Class Reference . . . . .	1463
5.432 spec_fader_t Class Reference . . . . .	1466
5.433 speechnoise_t Class Reference . . . . .	1467
5.434 steerbf Class Reference . . . . .	1470
5.435 steerbf_config Class Reference . . . . .	1473
5.436 testplugin::ac_parser_t Class Reference . . . . .	1475
5.437 testplugin::config_parser_t Class Reference . . . . .	1478
5.438 testplugin::if_t Class Reference . . . . .	1481
5.439 testplugin::signal_parser_t Class Reference . . . . .	1484
5.440 us_t Class Reference . . . . .	1485
5.441 wave2spec_if_t Class Reference . . . . .	1487
5.442 wave2spec_t Class Reference . . . . .	1492
5.443 wavrec_t Class Reference . . . . .	1498
5.444 wavwriter_t Class Reference . . . . .	1501
5.445 windnoise::cfg_t Class Reference . . . . .	1504
5.446 windnoise::if_t Class Reference . . . . .	1509
5.447 windowselector_t Class Reference . . . . .	1513
<b>6 File Documentation</b> . . . . .	<b>1517</b>
6.1 ac2lsl.cpp File Reference . . . . .	1517
6.2 ac2osc.cpp File Reference . . . . .	1518
6.3 ac2wave.cpp File Reference . . . . .	1518
6.4 ac_monitor_type.cpp File Reference . . . . .	1518
6.5 ac_monitor_type.hh File Reference . . . . .	1518
6.6 ac_mul.cpp File Reference . . . . .	1519
6.7 ac_mul.hh File Reference . . . . .	1519
6.8 ac_proc.cpp File Reference . . . . .	1520
6.9 acConcat_wave.cpp File Reference . . . . .	1520
6.10 acConcat_wave.h File Reference . . . . .	1521
6.11 acmon.cpp File Reference . . . . .	1521
6.12 acPooling_wave.cpp File Reference . . . . .	1521
6.13 acPooling_wave.h File Reference . . . . .	1522
6.14 acrec.cpp File Reference . . . . .	1522

6.15	acrec.hh File Reference . . . . .	1522
6.16	acsave.cpp File Reference . . . . .	1522
6.17	acSteer.cpp File Reference . . . . .	1524
6.18	acSteer.h File Reference . . . . .	1524
6.19	acTransform_wave.cpp File Reference . . . . .	1524
6.20	acTransform_wave.h File Reference . . . . .	1525
6.21	adaptive_feedback_canceller.cpp File Reference . . . . .	1525
6.22	adaptive_feedback_canceller.h File Reference . . . . .	1526
6.23	addsndfile.cpp File Reference . . . . .	1526
6.24	adm.cpp File Reference . . . . .	1527
6.25	adm.hh File Reference . . . . .	1528
6.26	altconfig.cpp File Reference . . . . .	1529
6.27	altconfig.hh File Reference . . . . .	1529
6.28	altpugs.cpp File Reference . . . . .	1529
6.29	analysemhaplugin.cpp File Reference . . . . .	1530
6.30	analysispath.cpp File Reference . . . . .	1531
6.31	attenuate20.cpp File Reference . . . . .	1531
6.32	audiometerbackend.cpp File Reference . . . . .	1532
6.33	auditory_profile.cpp File Reference . . . . .	1533
6.34	auditory_profile.h File Reference . . . . .	1533
6.35	browsemhaplugins.cpp File Reference . . . . .	1533
6.36	coherence.cpp File Reference . . . . .	1534
6.37	combinechannels.cpp File Reference . . . . .	1534
6.38	compiler_id.cpp File Reference . . . . .	1535
6.39	compiler_id.hh File Reference . . . . .	1535
6.40	complex_filter.cpp File Reference . . . . .	1536
6.41	complex_filter.h File Reference . . . . .	1536
6.42	complex_scale_channel.cpp File Reference . . . . .	1537
6.43	cpupload.cpp File Reference . . . . .	1537
6.44	db.cpp File Reference . . . . .	1537
6.45	dbasync.cpp File Reference . . . . .	1537
6.46	dc.cpp File Reference . . . . .	1538
6.47	dc.hh File Reference . . . . .	1539
6.48	dc_afterburn.cpp File Reference . . . . .	1539
6.49	dc_afterburn.h File Reference . . . . .	1539
6.50	dc_simple.cpp File Reference . . . . .	1540
6.51	dc_simple.hh File Reference . . . . .	1540
6.52	delay.cpp File Reference . . . . .	1541
6.53	delay.hh File Reference . . . . .	1541
6.54	delaysum_spec.cpp File Reference . . . . .	1541
6.55	delaysum_wave.cpp File Reference . . . . .	1542
6.56	doasvm_classification.cpp File Reference . . . . .	1542
6.57	doasvm_classification.h File Reference . . . . .	1543
6.58	doasvm_feature_extraction.cpp File Reference . . . . .	1543
6.59	doasvm_feature_extraction.h File Reference . . . . .	1543
6.60	doc_appendix.h File Reference . . . . .	1544
6.61	doc_examples.h File Reference . . . . .	1544
6.62	doc_frameworks.h File Reference . . . . .	1544
6.63	doc_general.h File Reference . . . . .	1544
6.64	doc_kernel.h File Reference . . . . .	1544
6.65	doc_matlab.h File Reference . . . . .	1544

6.66	doc_mhamain.h File Reference . . . . .	1544
6.67	doc_parser.h File Reference . . . . .	1544
6.68	doc_plugins.h File Reference . . . . .	1544
6.69	doc_system.h File Reference . . . . .	1544
6.70	doc_toolbox.h File Reference . . . . .	1544
6.71	double2acvar.cpp File Reference . . . . .	1544
6.72	downsample.cpp File Reference . . . . .	1544
6.73	dropgen.cpp File Reference . . . . .	1545
6.74	droptect.cpp File Reference . . . . .	1545
6.75	equalize.cpp File Reference . . . . .	1545
6.76	example1.cpp File Reference . . . . .	1545
6.77	example2.cpp File Reference . . . . .	1545
6.78	example3.cpp File Reference . . . . .	1546
6.79	example4.cpp File Reference . . . . .	1546
6.80	example5.cpp File Reference . . . . .	1546
6.81	example6.cpp File Reference . . . . .	1546
6.82	example7.cpp File Reference . . . . .	1547
6.83	example7.hh File Reference . . . . .	1547
6.84	fader_spec.cpp File Reference . . . . .	1547
6.85	fader_wave.cpp File Reference . . . . .	1547
6.86	fftfbpow.cpp File Reference . . . . .	1548
6.87	fftfilter.cpp File Reference . . . . .	1548
6.88	fftfilterbank.cpp File Reference . . . . .	1549
6.89	fshift.cpp File Reference . . . . .	1549
6.90	fshift.hh File Reference . . . . .	1549
6.91	fshift_hilbert.cpp File Reference . . . . .	1550
6.92	gain.cpp File Reference . . . . .	1550
6.93	gaintable.cpp File Reference . . . . .	1550
6.94	gaintable.h File Reference . . . . .	1551
6.95	generatemhaplugin.doc.cpp File Reference . . . . .	1551
6.96	gsc_adaptive_stage.cpp File Reference . . . . .	1554
6.97	gsc_adaptive_stage.hh File Reference . . . . .	1554
6.98	gsc_adaptive_stage_if.cpp File Reference . . . . .	1554
6.99	gsc_adaptive_stage_if.hh File Reference . . . . .	1554
6.100	gtfb_analyzer.cpp File Reference . . . . .	1554
6.101	gtfb_simd.cpp File Reference . . . . .	1556
6.102	gtfb_simple_bridge.cpp File Reference . . . . .	1561
6.103	hann.cpp File Reference . . . . .	1562
6.104	hann.h File Reference . . . . .	1562
6.105	identity.cpp File Reference . . . . .	1563
6.106	ifftshift.cpp File Reference . . . . .	1563
6.107	ifftshift.h File Reference . . . . .	1563
6.108	iirfilter.cpp File Reference . . . . .	1564
6.109	level_matching.cpp File Reference . . . . .	1564
6.110	level_matching.hh File Reference . . . . .	1565
6.111	levelmeter.cpp File Reference . . . . .	1565
6.112	lpc.cpp File Reference . . . . .	1565
6.113	lpc.h File Reference . . . . .	1566
6.114	lpc_bl_predictor.cpp File Reference . . . . .	1566
6.115	lpc_bl_predictor.h File Reference . . . . .	1567
6.116	lpc_burg-lattice.cpp File Reference . . . . .	1567

6.117 lpc_burg-lattice.h File Reference . . . . .	1568
6.118 Isl2ac.cpp File Reference . . . . .	1568
6.119 Isl2ac.hh File Reference . . . . .	1568
6.120 matlab_wrapper.cpp File Reference . . . . .	1569
6.121 matlab_wrapper.hh File Reference . . . . .	1569
6.122 matrixmixer.cpp File Reference . . . . .	1570
6.123 mconv.cpp File Reference . . . . .	1570
6.124 mha.cpp File Reference . . . . .	1570
6.125 mha.hh File Reference . . . . .	1571
6.126 mha_algo_comm.cpp File Reference . . . . .	1579
6.127 mha_algo_comm.h File Reference . . . . .	1580
6.128 mha_algo_comm.hh File Reference . . . . .	1581
6.129 mha_defs.h File Reference . . . . .	1582
6.130 mha_errno.c File Reference . . . . .	1584
6.131 mha_errno.h File Reference . . . . .	1586
6.132 mha_error.cpp File Reference . . . . .	1587
6.133 mha_error.hh File Reference . . . . .	1588
6.134 mha_event_emitter.h File Reference . . . . .	1589
6.135 mha_events.cpp File Reference . . . . .	1589
6.136 mha_events.h File Reference . . . . .	1589
6.137 mha_fftfb.cpp File Reference . . . . .	1590
6.138 mha_fftfb.hh File Reference . . . . .	1592
6.139 mha_fifo.cpp File Reference . . . . .	1593
6.140 mha_fifo.h File Reference . . . . .	1593
6.141 mha_filter.cpp File Reference . . . . .	1594
6.142 mha_filter.hh File Reference . . . . .	1594
6.143 mha_generic_chain.cpp File Reference . . . . .	1596
6.144 mha_generic_chain.h File Reference . . . . .	1596
6.145 mha_git_commit_hash.cpp File Reference . . . . .	1597
6.146 mha_git_commit_hash.hh File Reference . . . . .	1598
6.147 mha_io_ifc.h File Reference . . . . .	1598
6.148 mha_io_utils.cpp File Reference . . . . .	1601
6.149 mha_io_utils.hh File Reference . . . . .	1601
6.150 mha_multisrc.cpp File Reference . . . . .	1601
6.151 mha_multisrc.h File Reference . . . . .	1601
6.152 mha_os.cpp File Reference . . . . .	1602
6.153 mha_os.h File Reference . . . . .	1604
6.154 mha_parser.cpp File Reference . . . . .	1609
6.155 mha_parser.hh File Reference . . . . .	1613
6.156 mha_plugin.cpp File Reference . . . . .	1617
6.157 mha_plugin.hh File Reference . . . . .	1617
6.158 mha_profiling.c File Reference . . . . .	1621
6.159 mha_profiling.h File Reference . . . . .	1622
6.160 mha_ruby.cpp File Reference . . . . .	1623
6.161 mha_signal.cpp File Reference . . . . .	1624
6.162 mha_signal.hh File Reference . . . . .	1628
6.163 mha_signal_fft.h File Reference . . . . .	1638
6.164 mha_tablelookup.cpp File Reference . . . . .	1638
6.165 mha_tablelookup.hh File Reference . . . . .	1638
6.166 mha_tcp.cpp File Reference . . . . .	1639
6.167 mha_tcp.hh File Reference . . . . .	1642

6.168 mha_tcp_server.cpp File Reference . . . . .	1643
6.169 mha_tcp_server.hh File Reference . . . . .	1643
6.170 mha_toolbox.h File Reference . . . . .	1644
6.171 mha_utils.cpp File Reference . . . . .	1644
6.172 mha_utils.hh File Reference . . . . .	1644
6.173 mha_windowparser.cpp File Reference . . . . .	1644
6.174 mha_windowparser.h File Reference . . . . .	1645
6.175 mhachain.cpp File Reference . . . . .	1646
6.176 mhafw_lib.cpp File Reference . . . . .	1646
6.177 mhafw_lib.h File Reference . . . . .	1646
6.178 MHAIOalsa.cpp File Reference . . . . .	1646
6.179 MHAIOAsterisk.cpp File Reference . . . . .	1651
6.180 MHAIOFile.cpp File Reference . . . . .	1656
6.181 MHAIOJack.cpp File Reference . . . . .	1661
6.182 MHAIOJackdb.cpp File Reference . . . . .	1665
6.183 MHAIOParser.cpp File Reference . . . . .	1669
6.184 MHAIOPortAudio.cpp File Reference . . . . .	1673
6.185 MHAIOTCP.cpp File Reference . . . . .	1678
6.186 mhajack.cpp File Reference . . . . .	1683
6.187 mhajack.h File Reference . . . . .	1684
6.188 mhamain.cpp File Reference . . . . .	1686
6.189 mhaplugloader.cpp File Reference . . . . .	1688
6.190 mhaplugloader.h File Reference . . . . .	1688
6.191 mhasndfile.cpp File Reference . . . . .	1688
6.192 mhasndfile.h File Reference . . . . .	1689
6.193 multibandcompressor.cpp File Reference . . . . .	1690
6.194 nlms_wave.cpp File Reference . . . . .	1690
6.195 noise.cpp File Reference . . . . .	1692
6.196 noise_psd_estimator.cpp File Reference . . . . .	1692
6.197 osc2ac.cpp File Reference . . . . .	1692
6.198 overlapadd.cpp File Reference . . . . .	1693
6.199 overlapadd.hh File Reference . . . . .	1693
6.200 plingploing.cpp File Reference . . . . .	1693
6.201 pluginbrowser.cpp File Reference . . . . .	1694
6.202 pluginbrowser.h File Reference . . . . .	1694
6.203 proc_counter.cpp File Reference . . . . .	1694
6.204 resampling.cpp File Reference . . . . .	1694
6.205 rmslevel.cpp File Reference . . . . .	1694
6.206 rohBeam.cpp File Reference . . . . .	1695
6.207 rohBeam.hh File Reference . . . . .	1695
6.208 route.cpp File Reference . . . . .	1696
6.209 save_spec.cpp File Reference . . . . .	1696
6.210 save_wave.cpp File Reference . . . . .	1696
6.211 shadowfilter_begin.cpp File Reference . . . . .	1697
6.212 shadowfilter_end.cpp File Reference . . . . .	1697
6.213 sine.cpp File Reference . . . . .	1697
6.214 smooth_cepstrum.cpp File Reference . . . . .	1697
6.215 smooth_cepstrum.hh File Reference . . . . .	1698
6.216 smoothgains_bridge.cpp File Reference . . . . .	1698
6.217 softclip.cpp File Reference . . . . .	1699
6.218 spec2wave.cpp File Reference . . . . .	1699

6.219 speechnoise.cpp File Reference . . . . .	1699
6.220 speechnoise.h File Reference . . . . .	1703
6.221 split.cpp File Reference . . . . .	1704
6.222 steerbf.cpp File Reference . . . . .	1705
6.223 steerbf.h File Reference . . . . .	1706
6.224 testalsadvice.c File Reference . . . . .	1706
6.225 testplugin.cpp File Reference . . . . .	1706
6.226 transducers.cpp File Reference . . . . .	1707
6.227 upsample.cpp File Reference . . . . .	1708
6.228 wave2spec.cpp File Reference . . . . .	1708
6.229 wave2spec.hh File Reference . . . . .	1708
6.230 wavrec.cpp File Reference . . . . .	1709
6.231 windnoise.cpp File Reference . . . . .	1709
6.232 windnoise.hh File Reference . . . . .	1709
6.233 windowselector.cpp File Reference . . . . .	1709
6.234 windowselector.h File Reference . . . . .	1709
<b>Index</b>	<b>1711</b>

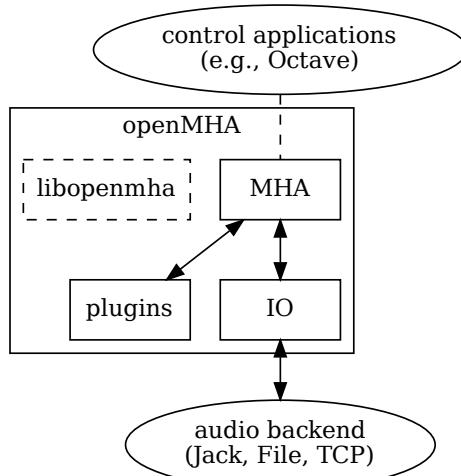
# 1 Overview

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. 601), [MHAIQJack](#) (p. 109), [io\\_parser\\_t](#) (p. 612), [io\\_tcp\\_parser\\_t](#) (p. 622))
- **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  $\text{rms}$  (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  $\text{rms}$  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 Deprecated List

### **Member MHParse::base\_t::base\_t (p. 1030) (const base\_t (p. 1026) &)**

Copying parser nodes makes little sense, avoid wherever possible

## 3 Module Documentation

### 3.1 Concept of Variables and Data Exchange in the openMHA

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

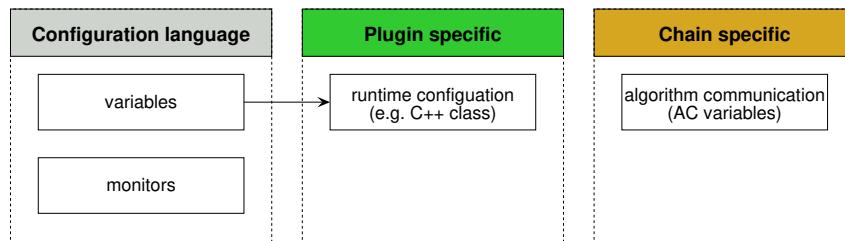


Figure 2 Variable types in the openMHA

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

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

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

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.

### 3.2.1 example1.cpp

The example plugin file `example1.cpp` (p. 1545) demonstrates the easiest way to implement an openMHA Plugin. It attenuates the sound signal in the first channel by multiplying the sound samples with a factor. The plugin class `MHAPlugin::plugin_t` (p. 1147) exports several methods, but only two of them need a non-empty implementation:

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

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

Every plugin implementation should include the '`mha_plugin.hh` (p. 1617)' 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. 1617).

The class `example1_t` (p. 463) inherits from the class `MHAParser::parser_t` (p. 1147), which in turn inherits from `MHAParser::parser_t` (p. 1098) – the configuration language interface in the method "parse". Our plugin class therefore inherits the "parse" method from `MHAParser::parser_t` (p. 1098), which integrates the plugin into the global openMHA configuration tree.

The constructor has to accept two parameters of types `algo_comm_t` (p. 279) and `std::string`, respectively. In this simple example, we do not make use of them.

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

### 3.2.1.1 The prepare method

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

#### Parameters

<code>signal_info</code>	Contains information about the input signal's dimensions, see <code>mhaconfig_t</code> (p. 847).
--------------------------	--

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

### 3.2.1.2 The signal processing method

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

#### Parameters

<code>signal</code>	Pointer to the input signal structure <code>mha_wave_t</code> (p. 836).
---------------------	---

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

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

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

This macro takes care of accessing the C++ class from the C functions required as the plugin's interface. It implements the C funtions and calls the corresponding C++ instance methods. Plugin classes should be derived from the template class **MHAPlugin::plugin\_t** (p. 1147) to be compatible with the C interface wrapper.

This macro also catches C++ exceptions of type **MHA\_Error** (p. 760), 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. 760) 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. 760). This is important, because: (1) C++ exceptions cannot cross the plugin interface, which is in C, and (2) there is no error handling code for your exception classes in the openMHA framework anyways.

## 3.2.2 example2.cpp

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

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

```
class example2_t : public MHAPlugin::plugin_t<int> {
    MHAParser::int_t scale_ch;
    MHAParser::float_t factor;
public:
    example2_t(algo_comm_t iac, const std::string & configured_name);
    void prepare(mhaconfig_t & signal_info);
    void release(void);
    mha_wave_t * process(mha_wave_t * signal);
};
```

## Parameters

<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. 1147) for intergration with the openMHA configuration language. The two data members serve as externally visible configuration variables. All methods of this class have a non-empty implementation.

### 3.2.2.1 Constructor

```
example2_t::example2_t(algo_comm_t iac, const std::string & configured_name)
: MHAPlugIn::plugin_t<int>("This plugin multiplies the sound signal"
    " in one audio channel by a factor", iac),
scale_ch("Index of audio channel to scale. Indices start from 0.",
    "0",
    "[0, [",
factor("The scaling factor that is applied to the selected channel.",
    "0.1",
    "[0, [")
{
    insert_item("channel", &scale_ch);
    insert_item("factor", &factor);
    (void)configured_name; // Ignore 2nd parameter
}
```

The constructor invokes the superclass constructor with a string parameter. This string parameter serves as the help text that describes the functionality of the plugin. The constructor registers configuration variables with the openMHA configuration tree and sets their default values and permitted ranges. The minimum permitted value for both variables is zero, and there is no maximum limit (apart from the limitations of the underlying C data type). The configuration variables have to be registered with the parser node instance using the **MHAParser::parser\_t::insert\_item** (p. 1100) method.

### 3.2.2.2 The prepare method

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

#### Parameters

<i>signal_info</i>	– contains information about the input signal's parameters, see <b>mhaconfig_t</b> (p. 847).
--------------------	--

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

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

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

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

### 3.2.2.3 The release method

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

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

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

### 3.2.2.4 The signal processing method

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

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

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

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

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

Fortunately, for some data types on PC systems, case (c) cannot happen. These are 32bit wide data types with a 4-byte alignment. Therefore, the values in **MHAParser::int\_t** (p. 1065) and **MHAParser::float\_t** (p. 1059) are always consistent, but this is not the case for vectors, strings, or complex values. With these, you can get a mixture of the bit patterns of old and new values, or you can even cause a memory access violation in case a vector or string grows and has to be reallocated to a different memory address.

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

### 3.2.3 example3.cpp

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

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

#### **writeaccess**

- triggered on write access to a configuration variable.

#### **valuechanged**

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

#### **readaccess**

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

#### **prereadaccess**

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

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

### 3.2.3.1 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.

### 3.2.3.2 Method declarations

```
/* Callbacks triggered by Events */
void on_scale_ch_writeaccess();
void on_scale_ch_valuechanged();
void on_scale_ch_readaccess();
void on_prereadaccess();

public:
    example3_t(algo_comm_t iac, const std::string & configured_name);
    void prepare(mhaconfig_t & signal_info);
    void release(void);
    mha_wave_t * process(mha_wave_t * signal);
};
```

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

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

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

### 3.2.3.3 Example 3 constructor

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

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

```

patchbay.connect(&factor.prereadaccess, this,
                 &example3_t::on_prereadaccess);
patchbay.connect(&prepared.prereadaccess, this,
                 &example3_t::on_prereadaccess);
}

```

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

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

#### 3.2.3.4 The prepare method

```

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

```

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

#### 3.2.3.5 The release method

```

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

```

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

#### 3.2.3.6 The signal processing method

```

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

```

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

### 3.2.3.7 The callback methods

```

void example3_t::on_scale_ch_writeaccess()
{
    printf("Write access: Attempt to set scale_ch=%d.\n", scale_ch.data);
    // Can be used to track any writeaccess to the configuration, even
    // if it does not change the value. E.g. setting the name of the
    // sound file in a string configuration variable can cause a sound
    // file player plugin to start playing the sound file from the
    // beginning.
}
void example3_t::on_scale_ch_valuechanged()
{
    if (scale_ch.data & 1)
        throw MHA_Error(__FILE__, __LINE__,
                        "Attempt to set scale_ch to non-even value %d",
                        scale_ch.data);
    // Can be used to recompute a runtime configuration only if some
    // configuration variable actually changed.
}
void example3_t::on_scale_ch_readaccess()
{
    printf("scale_ch has been read.\n");
    // A configuration variable used as an accumulator can be reset
    // after it has been read.
}
void example3_t::on_prereadaccess()
{
    printf("A configuration language variable is about to be read.\n");
    // Can be used to compute the value on demand.
}
MHAPLUGIN_CALLBACKS(example3,example3_t,wave,wave)

```

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

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

### 3.2.4 example4.cpp

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

#### 3.2.4.1 The Prepare method

```

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

```

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

### 3.2.4.2 The signal processing method

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

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

The **mha\_spec\_t** (p. 790) instance stores the complex (**mha\_complex\_t** (p. 741)) spectral signal for positive frequencies only (since the waveform signal is always real). The **num\_frames** member of **mha\_spec\_t** (p. 790) actually denotes the number of STFT bins.

Please note that different from **mha\_wave\_t** (p. 836), a multichannel signal in **mha\_spec\_t** (p. 790) is stored non-interleaved in the signal buffer.

Some arithmetic operations are defined on struct **mha\_complex\_t** (p. 741) to facilitate efficient complex computations. The \*= operator used here (defined for real and for complex arguments) is one of them.

### 3.2.4.3 Connecting the C++ class with the C plugin interface

MHAPLUGIN\_CALLBACKS (example4,example4\_t,spec,spec)

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

## 3.2.5 example5.cpp

Many algorithms use complex operations to transform the user space variables into run time configurations. If this takes a noticeable time (e.g. more than 100-500  $\mu$  sec), the update of the runtime configuration can not take place in the real time processing thread. Furthermore, the parallel access to complex structures may cause unpredictable results if variables are read while only parts of them are written to memory (cf. section **Consistency** (p. 11)). To handle these situations, a special C++ template class **MHAPlugin::plugin\_t** (p. 1147) 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. 477) is the parameter of the template class **MHAPlugin::plugin\_t** (p. 1147). 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. 1147), 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. 857) instance.

```
class plugin_interface_t : public MHAPlugin::plugin_t<example5_t> {
public:
    plugin_interface_t(algo_comm_t iac, const std::string & configured_name);
    mha_spec_t* process(mha_spec_t* );
    void prepare(mhaconfig_t& );
private:
    void update_cfg();
    /* integer variable of MHA-parser: */
    MHAParser::int_t scale_ch;
    /* float variable of MHA-parser: */
    MHAParser::float_t factor;
    /* patch bay for connecting configuration parser
       events with local member functions: */
    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. 760) 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. 760) 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. 477) 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 **MHAEevents::patchbay\_t** (p. 857) instance.

```
plugin_interface_t::plugin_interface_t(algo_comm_t iac, 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. 760) 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. 1146) stores the configuration in a FiFo queue of runtime configurations. Once they are inserted in the FiFo, the **MHAPLUGIN::plugin\_t** (p. 1147) 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. 1620) defines all ANSI-C interface functions and passes them to the corresponding C++ class member functions (partly defined by the **MHAPLUGIN::plugin\_t** (p. 1147) template class). All exceptions of type **MHA\_Error** (p. 760) are caught and transformed into an appropriate error code and error message.

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

### 3.2.6 example6.cpp

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

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

```
example6_t::example6_t(algo_comm_t iac, 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 );
}
```

### 3.2.7 Debugging openMHA plugins

Suppose you would want to step through the code of your openMHA plugin with a debugger. This example details how to use the GDB debugger to inspect the `example6_t::prepare()` (p. 479) and `example6_t::process()` (p. 479) routines of `example6.cpp` (p. 18) example 6.

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

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

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

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

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

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

Set breakpoints in `prepare` and `process` methods, and start execution. Note that specifying the breakpoint by symbol (`example6_t::prepare` (p. 479)) 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 0x7fffff589744a: 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=0x7fffff6a06c0d, wave=0x10a8b550)
  at example6.cpp:162
162      {
(gdb) n
163      poll_config();
(gdb)

```

### 3.2.8 Writing unit tests for openMHA plugins

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

**3.2.8.1 example7** As an example, unit tests for plugin `example7.cpp` (p. 1547) are written, which is functionally the same as plugin `example1.cpp` (p. 1545) (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,"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. 481) 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_();
}
```

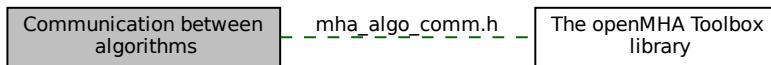
In this test the goal is to assess the main feature of the plugin (`example7_t` (p. 481)), which is the same as in `example1_t` (p. 463), namely altering the signal's first channel by a constant factor of 0.1. The variable `wave_input` is the signal that will be processed by the plugin. In order to assert 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.

### 3.3 The MHA Framework interface

## 3.4 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.*

### 3.4.1 Detailed Description

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

An algorithm communication handle (**algo\_comm\_t** (p. 279)) is passed at initialisation time to the constructor of each plugin class **constructor** (p. 1147). This handle contains a reference handle, **algo\_comm\_t::handle** (p. 280), 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. 359).

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

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

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

1. Plugins publishing AC variables:

- insert all variables during prepare()

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

### 3.4.2 Function Documentation

**3.4.2.1 get\_var\_spectrum()** `mha_spec_t MHA_AC::get_var_spectrum ( algo_comm_t ac, const std::string & name )`

Convert an AC variable into a spectrum.

This function reads an AC variable and tries to convert it into a valid spectrum. The Spectrum variable is 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

**3.4.2.2 get\_var\_waveform()** `mha_wave_t MHA_AC::get_var_waveform ( algo_comm_t ac, const std::string & name )`

Convert an AC variable into a waveform.

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

**Parameters**

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

**Returns**

waveform structure

**3.4.2.3 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

**3.4.2.4 get\_var\_float()**

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

Return value of an floating point scalar AC variable.

**Parameters**

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

**Returns**

Variable value

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

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

**Parameters**

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

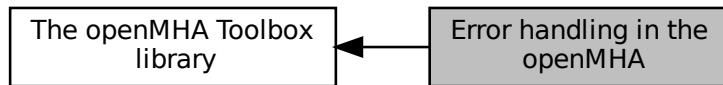
**Returns**

Variable value

### 3.5 Error handling in the openMHA

Errors are reported to the user via the **MHA\_Error** (p. 760) 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. 760).*
- #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. 760) with the values.*

#### Functions

- void **mha\_debug** (const char \*fmt,...) \_\_attribute\_\_((\_\_format\_\_(printf  
Print an info message (stderr on Linux, OutputDebugString in Windows).

##### 3.5.1 Detailed Description

Errors are reported to the user via the **MHA\_Error** (p. 760) exception.

##### 3.5.2 Macro Definition Documentation

###### 3.5.2.1 **MHA\_ErrorMsg** #define MHA\_ErrorMsg( x ) **MHA\_Error**(\_\_FILE\_\_, \_\_LINE\_\_, "%s", x)

Throw an openMHA error with a text message.

**Parameters**

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

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

Assertion macro, which throws an **MHA\_Error** (p. 760).

**Parameters**

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

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

Equality assertion macro, which throws an **MHA\_Error** (p. 760) with the values.

**Parameters**

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

**3.5.3 Function Documentation**

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

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

### 3.6 The openMHA configuration language

openMHA Plugins that should use the openMHA configuration language for their configuration have to be implemented in C++ and need to include **mha\_parser.hh** (p. 1613). 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. 1098) (or **MHAParser::MHAParser::plugin\_t** (p. 1147)), 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. 1100).

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. 1613). 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. 1098) (or **MHAParser::MHAParser::plugin\_t** (p. 1147)), 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. 1100).

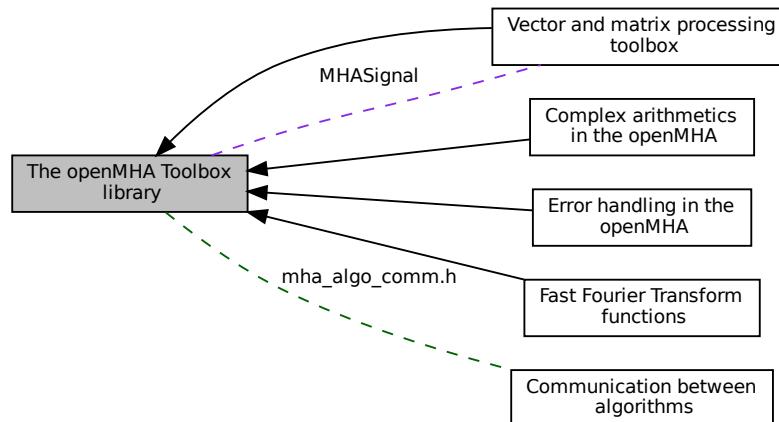
The openMHA Plugin template class **MHAParser::MHAParser::plugin\_t** (p. 1147) together with the Plugin macro **MHAPARSER\_CALLBACKS** (p. 1620) provide the callback mappings and correct inheritance. If your plugin is based on that template class, you simply have to use the **insert\_item** command 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.

### 3.7 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. 760) 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. 836) and [mha\\_spec\\_t](#) (p. 790).*

- **Complex arithmetics in the openMHA**

- **Fast Fourier Transform functions**

#### Files

- file **mha\_algo\_comm.h**

*Header file for Algorithm Communication.*

- file **mha\_filter.hh**

*Header file for IIR filter classes.*

- file **mha\_signal.hh**

*Header file for audio signal handling and processing classes.*

- file **mha\_tablelookup.hh**

*Header file for table lookup classes.*

## Namespaces

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

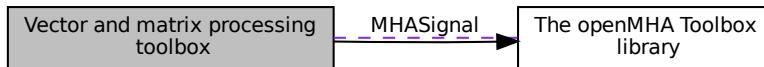
### 3.7.1 Detailed Description

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

## 3.8 Vector and matrix processing toolbox

The vector and matrix processing toolbox consists of a number of classes defined in the namespace **MHASignal** (p. 136), and many functions and operators for use with the structures **mha\_wave\_t** (p. 836) and **mha\_spec\_t** (p. 790).

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. 847)).*
- struct **mha\_audio\_t**  
*An audio fragment in the openMHA (planned as a replacement of **mha\_wave\_t** (p. 836) and **mha\_spec\_t** (p. 790)).*
- class **MHASignal::spectrum\_t**  
*a signal processing class for spectral data (based on **mha\_spec\_t** (p. 790))*
- class **MHASignal::waveform\_t**  
*signal processing class for waveform data (based on **mha\_wave\_t** (p. 836))*
- 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. 836) 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. 836) 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. 790) 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. 790) with its conjugate.*

### 3.8.1 Detailed Description

The vector and matrix processing toolbox consists of a number of classes defined in the namespace **MHASignal** (p. 136), and many functions and operators for use with the structures **mha\_wave\_t** (p. 836) and **mha\_spec\_t** (p. 790).

### 3.8.2 Typedef Documentation

### 3.8.2.1 **mha\_real\_t** `typedef float mha_real_t`

openMHA type for real numbers

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

## 3.8.3 Function Documentation

### 3.8.3.1 **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.

### 3.8.3.2 **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.

**3.8.3.3 bin2freq()** `mha_real_t MHASignal::bin2freq (`

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

conversion from fft bin index to frequency

**Parameters**

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

**Returns**

frequency of fft bin / Hz

**3.8.3.4 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

**3.8.3.5 smp2rad()** `mha_real_t MHASignal::smp2rad (`

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

conversion from delay in samples to phase shift

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

**Parameters**

<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  $\pi$  should not be used. If bin is 0 or nyqvist, returns 0 phase shift.

**3.8.3.6 rad2smp()** `mha_real_t MHASignal::rad2smp (`

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

conversion from phase shift to delay in samples

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

**Parameters**

<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.

**3.8.3.7 dupvec()** `template<class elem_type >`

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

Duplicate last vector element to match desired size.

**Parameters**

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

**Return values**

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

**3.8.3.8 dupvec\_chk()** `template<class elem_type >`

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

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

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

**Parameters**

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

**Return values**

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

---

**3.8.3.9 equal\_dim()** [1/6] `bool equal_dim (`  
`const mha_wave_t & a,`  
`const mha_wave_t & b ) [inline]`

Test for equal dimension of waveform structures.

**3.8.3.10 equal\_dim()** [2/6] `bool equal_dim (`  
`const mha_wave_t & a,`  
`const mhaconfig_t & b ) [inline]`

Test for match of waveform dimension with mhaconfig structure.

**3.8.3.11 equal\_dim()** [3/6] `bool equal_dim (`  
`const mha_spec_t & a,`  
`const mha_spec_t & b ) [inline]`

Test for equal dimension of spectrum structures.

**3.8.3.12 equal\_dim()** [4/6] `bool equal_dim (`  
`const mha_spec_t & a,`  
`const mhaconfig_t & b ) [inline]`

Test for match of spectrum dimension with mhaconfig structure.

**3.8.3.13 equal\_dim()** [5/6] `bool equal_dim (`  
`const mha_wave_t & a,`  
`const mha_spec_t & b ) [inline]`

Test for equal dimension of waveform/spectrum structures.

#### Warning

Waveform structures **mha\_wave\_t** (p. 836) use interleaved data order, while spectrum structures **mha\_spec\_t** (p. 790) use non-interleaved.

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

Test for equal dimension of waveform/spectrum structures.

#### Warning

Waveform structures **mha\_wave\_t** (p. 836) use interleaved data order, while spectrum structures **mha\_spec\_t** (p. 790) use non-interleaved.

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

Numeric integration of a signal vector (real values)

#### Parameters

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

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

Numeric integration of a signal vector (complex values)

#### Parameters

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

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

Return size of a waveform structure.

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

Return size of a spectrum structure.

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

Return size of a waveform structure.

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

Return size of a spectrum structure.

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

Set all values of waveform to zero.

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

Set all values of waveform to zero.

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

Set all values of spectrum to zero.

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

Set all values of spectrum to zero.

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

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

**Parameters**

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

**3.8.3.26 assign() [2/3]** void assign (

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

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

**Parameters**

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

**3.8.3.27 assign() [3/3]** void assign (

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

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

**Parameters**

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

**3.8.3.28 timeshift()** void timeshift (

```
    mha_wave_t & self,  
    int shift )
```

Time shift of waveform chunk.

Shifted areas are filled with zeros.

### Parameters

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

### 3.8.3.29 **value()** [1/8]

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

Access an element of a waveform structure.

### Parameters

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

### Returns

Reference to element

### 3.8.3.30 **value()** [2/8]

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

Constant access to an element of a waveform structure.

### Parameters

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

### Returns

Reference to element

---

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

Access to an element of a spectrum.

#### Parameters

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

#### Returns

Reference to element

---

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

Constant access to an element of a spectrum.

#### Parameters

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

#### Returns

Reference to element

---

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

Access to an element of a waveform structure.

**Parameters**

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

**Returns**

Reference to element

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

Constant access to an element of a waveform structure.

**Parameters**

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

**Returns**

Reference to element

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

Access to an element of a spectrum.

**Parameters**

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

**Returns**

Reference to element

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

Constant access to an element of a spectrum.

**Parameters**

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

**Returns**

Reference to element

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

Converts a **mha\_wave\_t** (p. 836) structure into a `std::vector<float>` (interleaved order).

**Warning**

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

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

Converts a **mha\_wave\_t** (p. 836) structure into a `std::vector< std::vector<float> >` (outer vector represents channels).

**Warning**

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

---

**3.8.3.39 std\_vector\_vector\_complex()** `std::vector<std::vector< mha_complex_t > >`  
`std_vector_vector_complex (`  
 `const mha_spec_t & )`

Converts a **mha\_spec\_t** (p. 790) structure into a `std::vector< std::vector<mha_complex_t> >` (outer vector represents channels).

#### Warning

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

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

Addition operator.

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

Addition operator.

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

Subtraction operator.

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

Subtraction operator.

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

Element-wise multiplication operator.

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

Element-wise multiplication operator.

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

Element-wise multiplication operator.

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

Element-wise multiplication operator.

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

Element-wise multiplication operator.

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

Element-wise division operator.

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

Element-wise division operator.

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

Addition operator.

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

Addition operator.

**3.8.3.53 operator^() mha\_wave\_t& operator^= (**  
 `mha_wave_t & self,`  
 `const mha_real_t & arg )`

Exponent operator.

#### Warning

This overwrites the xor operator!

**3.8.3.54 copy\_channel() [1/2]** `void MHASignal::copy_channel (`  
 `mha_spec_t & self,`  
 `const mha_spec_t & src,`  
 `unsigned sch,`  
 `unsigned dch )`

Copy one channel of a source signal.

### Parameters

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

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

Copy one channel of a source signal.

### Parameters

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

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

Return RMS level of a spectrum channel.

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

Takes into account the 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

**3.8.3.57 colored\_intensity()** `mha_real_t MHASignal::colored_intensity (`

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

Colored spectrum intensity.

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

**Parameters**

<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

**3.8.3.58 maxabs() [1/3]** `mha_real_t MHASignal::maxabs (`

```
const mha_spec_t & s,
unsigned int channel )
```

Find maximal absolute value.

**Parameters**

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

**Returns**

maximum absolute value

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

Return RMS level of a waveform channel.

**Parameters**

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

**Returns**

RMS level in Pa

**3.8.3.60 maxabs() [2/3]** `mha_real_t MHASignal::maxabs (`  
`const mha_wave_t & s,`  
`unsigned int channel )`

Find maximal absolute value.

**Parameters**

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

**Returns**

maximum absolute value

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

Find maximal absolute value.

**Parameters**

<code>s</code>	Input signal
----------------	--------------

**Returns**

maximum absolute value

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

Find maximal value.

**Parameters**

<code>s</code>	Input signal
----------------	--------------

**Returns**

maximum absolute value

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

Find minimal value.

**Parameters**

<code>s</code>	Input signal
----------------	--------------

**Returns**

maximum absolute value

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

Calculate sum of squared values in one channel.

#### Parameters

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

#### Returns

$$\sum x^2$$

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

Calculate sum over all channels of squared values.

#### Parameters

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

#### Returns

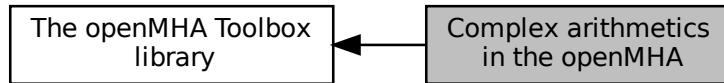
$$\sum x^2$$

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

Replace (!) the value of this **mha\_spec\_t** (p. 790) with its conjugate.

### 3.9 Complex arithmetics in the openMHA

Collaboration diagram for Complex arithmetics in the openMHA:



#### Classes

- struct **mha\_complex\_t**  
*Type for complex floating point values.*

#### Functions

- **mha\_complex\_t & set ( mha\_complex\_t &self, mha\_real\_t real, mha\_real\_t imag=0)**  
*Assign real and imaginary parts to a **mha\_complex\_t** (p. 741) variable.*
- **mha\_complex\_t mha\_complex ( mha\_real\_t real, mha\_real\_t imag=0)**  
*Create a new **mha\_complex\_t** (p. 741) 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. 741) 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. 741).*
- **mha\_complex\_t & expi ( mha\_complex\_t &self, mha\_real\_t angle)**  
*replaces the value of the given **mha\_complex\_t** (p. 741) 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. 741) 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. 741) 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.*

### 3.9.1 Detailed Description

### 3.9.2 Function Documentation

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

Assign real and imaginary parts to a **mha\_complex\_t** (p. 741) variable.

#### Parameters

<i>self</i>	The <b>mha_complex_t</b> (p. 741) variable whose value is about to change.
<i>real</i>	The new real part.
<i>imag</i>	The new imaginary part.

#### Returns

A reference to the changed variable.

---

**3.9.2.2 mha\_complex()** `mha_complex_t mha_complex (`  
 `mha_real_t real,`  
 `mha_real_t imag = 0 ) [inline]`

Create a new **mha\_complex\_t** (p. 741) with specified real and imaginary parts.

#### Parameters

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

#### Returns

The new value.

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

Assign a **mha\_complex\_t** (p. 741) variable from a std::complex.

#### Parameters

<i>self</i>	The <b>mha_complex_t</b> (p. 741) variable whose value is about to change.
<i>stdcomplex</i>	The new complex value.

#### Returns

A reference to the changed variable.

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

Create a std::complex from **mha\_complex\_t** (p. 741).

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

replaces the value of the given **mha\_complex\_t** (p. 741) with  $\exp(i \cdot b)$ .

**Parameters**

<i>self</i>	The <b>mha_complex_t</b> (p. 741) variable whose value is about to change.
<i>angle</i>	The angle in the complex plane [rad].

**Returns**

A reference to the changed variable.

**3.9.2.6 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.

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

Addition of two complex numbers, overwriting the first.

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

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

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

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

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

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

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

Subtraction of two complex numbers, overwriting the first.

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

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

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

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

---

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

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

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

Multiplication of two complex numbers, overwriting the first.

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

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

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

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

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

replaces (!) the value of the given **mha\_complex\_t** (p. 741) with a  $\ast \exp(i\ast b)$

#### Parameters

<b>self</b>	The <b>mha_complex_t</b> (p. 741) variable whose value is about to change.
<b>angle</b>	The imaginary exponent.
<b>factor</b>	The absolute value of the result.

**Returns**

A reference to the changed variable.

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

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

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

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

**Returns**

The square of the absolute value of self.

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

Compute the absolute value of a complex value.

**Returns**

The absolute value of self.

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

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

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

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

**3.9.2.24 safe\_div()** `mha_complex_t& safe_div (`  
`mha_complex_t & self,`  
`const mha_complex_t & other,`  
`mha_real_t eps,`  
`mha_real_t eps2 ) [inline]`

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

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

Division of two complex numbers, overwriting the first.

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

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

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

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

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

Compare two complex numbers for equality.

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

Compare two complex numbers for inequality.

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

Replace (!) the value of this **mha\_complex\_t** (p. 741) with its conjugate.

```
3.9.2.31 _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.

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

Replace the value of this complex with its reciprocal.

---

**3.9.2.33 `_reciprocal()`** `mha_complex_t _reciprocal (`  
`const mha_complex_t & self ) [inline]`

compute the reciprocal of this complex value.

#### Returns

A temporary object holding the reciprocal value.

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

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

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

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

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

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

#### Parameters

<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&lt;mha_real_t&gt;::epsilon</code> )

**Returns**

true if the relative difference is below times\_epsilon \* std::numeric\_limits<mha\_real\_t>::epsilon

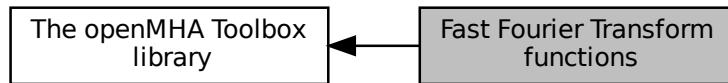
**3.9.2.37 operator<()** `bool operator< (`

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

Compares the absolute values of two complex numbers.

### 3.10 Fast Fourier Transform functions

Collaboration diagram for Fast Fourier Transform functions:



#### Typedefs

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

#### Functions

- `mha_fft_t mha_fft_new (unsigned int n)`  
*Create a new FFT handle.*
- `void mha_fft_free ( mha_fft_t h)`  
*Destroy an FFT handle.*
- `void mha_fft_wave2spec ( mha_fft_t h, const mha_wave_t *in, mha_spec_t *out)`  
*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.*

### 3.10.1 Detailed Description

### 3.10.2 Typedef Documentation

#### 3.10.2.1 **mha\_fft\_t** typedef void\* mha\_fft\_t

Handle for an FFT object.

This FFT object is used by the functions `mha_fft_wave2spec` and `mha_fft_spec2wave`. The 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.

### 3.10.3 Function Documentation

#### 3.10.3.1 **mha\_fft\_new()** **mha\_fft\_t** mha\_fft\_new ( unsigned int n )

Create a new FFT handle.

Parameters

<b>n</b>	FFT length.
----------	-------------

Create a new FFT handle.

Parameters

<b>n</b>	FFT length
----------	------------

Return values

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

**3.10.3.2 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
----------	--------------------------

**3.10.3.3 mha\_fft\_wave2spec() [1/2]** void mha\_fft\_wave2spec (   
     *mha\_fft\_t h,*   
     *const mha\_wave\_t \* in,*   
     *mha\_spec\_t \* out* )

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)

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

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.

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

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

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)

#### 3.10.3.6 mha\_fft\_spec2wave() [2/2]

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

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.

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

Complex to complex FFT (forward).

*sIn* and *sOut* need to have nfft bins (please note that **mha\_spec\_t** (p. 790) typically has nfft/2+1 bins for half-complex representation).

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

#### Parameters

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

```
3.10.3.8 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. 790) typically has nfft/2+1 bins for half-complex representation).

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

#### Parameters

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

### 3.10.3.9 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. 790) typically has nfft/2+1 bins for half-complex representation).

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

#### Parameters

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

### 3.10.3.10 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. 790) typically has nfft/2+1 bins for half-complex representation).

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

#### Parameters

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

---

**3.10.3.11 mha\_fft\_wave2spec\_scale()** void mha\_fft\_wave2spec\_scale (

<b>mha_fft_t</b>	<i>h</i> ,
const <b>mha_wave_t</b> *	<i>in</i> ,
<b>mha_spec_t</b> *	<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.

**3.10.3.12 mha\_fft\_spec2wave\_scale()** void mha\_fft\_spec2wave\_scale (

<b>mha_fft_t</b>	<i>h</i> ,
const <b>mha_spec_t</b> *	<i>in</i> ,
<b>mha_wave_t</b> *	<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.

## 4 Namespace Documentation

### 4.1 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**

#### 4.1.1 Detailed Description

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

#### 4.1.2 Variable Documentation

##### 4.1.2.1 **types** const std::map<int, **type\_info**> ac2isl::types

## 4.2 ac\_proc Namespace Reference

#### Classes

- class **interface\_t**

## 4.3 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**

### 4.3.1 Detailed Description

Namespace for displaying ac variables as parser monitors.

## 4.4 acsave Namespace Reference

### Classes

- class **acsave\_t**
- class **cfg\_t**
- struct **mat4head\_t**
- class **save\_var\_t**

## 4.5 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)

### 4.5.1 Typedef Documentation

**4.5.1.1 level\_adaptor** `typedef MHAPlugin::config_t< level_adapt_t> addsndfile::level_adaptor`

**4.5.1.2 wave\_reader** `typedef MHAPlugin::plugin_t< sndfile_t> addsndfile::wave_reader`

### 4.5.2 Enumeration Type Documentation

**4.5.2.1 addsndfile\_resampling\_mode\_t** `enum addsndfile::addsndfile_resampling_mode_t`

Specifies the resampling mode in **resampled\_soundfile\_t** (p. 257).

#### Enumerator

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

### 4.5.3 Function Documentation

```
4.5.3.1 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]
```

## 4.6 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

### 4.6.1 Function Documentation

```
4.6.1.1 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

**4.6.2 Variable Documentation**

**4.6.2.1 PI** const double ADM::PI = 3.14159265358979312

**4.6.2.2 C** const double ADM::C = 340

**4.6.2.3 DELAY\_FREQ** const double ADM::DELAY\_FREQ = 2000

**4.6.2.4 START\_BETA** const double ADM::START\_BETA = 0.5

**4.7 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)`

### 4.7.1 Typedef Documentation

**4.7.1.1 level\_adaptor** `typedef MHAPlugin::config_t< level_adapt_t > audiometerbackend::level_adaptor`

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

### 4.7.2 Function Documentation

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

**4.7.2.2 return\_sig()** `MHASignal::waveform_t audiometerbackend::return_sig (`  
    `unsigned int sigtype,`  
    `unsigned int fs,`  
    `unsigned int f )`

## 4.8 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.*

### 4.8.1 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.

## 4.9 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)

### 4.9.1 Function Documentation

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

## 4.10 cpuload Namespace Reference

### Classes

- class **cpuload\_cfg\_t**
- class **cpuload\_if\_t**

## 4.11 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)

## 4.11.1 Enumeration Type Documentation

### 4.11.1.1 anonymous enum anonymous enum

## Enumerator

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

### 4.11.2 Function Documentation

**4.11.2.1 `thread_start()`** static void\* dbasync\_native::thread\_start ( void \* *instance* ) [static]

**4.11.2.2 `gcd()`** static unsigned dbasync\_native::gcd ( unsigned *a*, unsigned *b* ) [inline], [static]

## 4.12 dc Namespace Reference

### Classes

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

## 4.13 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.*

### 4.13.1 Typedef Documentation

#### 4.13.1.1 DC `typedef MHAPlugin::plugin_t< dc_t > dc_simple::DC`

#### 4.13.1.2 LEVEL `typedef MHAPlugin::config_t< level_smoothen_t > dc_simple::LEVEL`

### 4.13.2 Function Documentation

#### 4.13.2.1 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

<b>MHA_Error</b> (p. 760)	if the size of <i>v</i> is neither <i>s</i> nor 1.
---------------------------	--

```
4.13.2.2 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

<b>MHA_Error</b> (p. 760)	if size of <i>v</i> is neither <i>s</i> nor 1.
---------------------------	--

```
4.13.2.3 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. 760)	if <i>x</i> == 0.
---------------------------	-------------------

## 4.14 delay Namespace Reference

### Classes

- class **interface\_t**

## 4.15 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.*

### 4.15.1 Detailed Description

This namespace contains the delaysum plugin.

## 4.16 delaysum\_spec Namespace Reference

### Classes

- class **delaysum\_spec\_if\_t**
- class **delaysum\_t**

## 4.17 double2acvar Namespace Reference

### Classes

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

## 4.18 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. 449) class.*
- class **gaintable\_t**  
*Gain table class.*

### Functions

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

### 4.18.1 Detailed Description

dynamic compression related classes and functions

### 4.18.2 Function Documentation

```
4.18.2.1 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 $X$ .
---------------------	--------------------------------

**4.18.2.2 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 $X,Y$ .
---------------------	------------------------------------

**4.19 equalize Namespace Reference****Classes**

- class **cfg\_t**
- class **freqgains\_t**

## 4.20 fader\_wave Namespace Reference

### Classes

- class **fader\_wave\_if\_t**
- class **level\_adapt\_t**

### Typedefs

- typedef **MHAPlugin::plugin\_t< level\_adapt\_t > level\_adaptor**

#### 4.20.1 Typedef Documentation

**4.20.1.1 level\_adaptor** `typedef MHAPlugin::plugin_t< level_adapt_t > fader_wave::level_adaptor`

## 4.21 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.*

#### 4.21.1 Detailed Description

Namespace for the fftfbpow plugin.

## 4.22 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**)

### 4.22.1 Function Documentation

**4.22.1.1 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

**4.22.1.2 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.

## 4.23 fftfilterbank Namespace Reference

### Classes

- class **fftfb\_interface\_t**
- class **fftfb\_plug\_t**

## 4.24 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.*

### 4.24.1 Detailed Description

All types for the fshift plugin live in this namespace.

### 4.24.2 Function Documentation

#### 4.24.2.1 **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

## 4.25 fshift\_hilbert Namespace Reference

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

## Classes

- class `frequency_translator_t`
- class `hilbert_shifter_t`

### 4.25.1 Detailed Description

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

## 4.26 gain Namespace Reference

## Classes

- class `gain_if_t`
- class `scaler_t`

## 4.27 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.*

### 4.27.1 Variable Documentation

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

Small constant to ensure no division by zero occurs.

## 4.28 gtfb\_analyzer Namespace Reference

### Classes

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

## 4.29 level\_matching Namespace Reference

### Classes

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

## 4.30 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** }

### 4.30.1 Enumeration Type Documentation

#### 4.30.1.1 **overrun\_behavior** enum ls12ac::overrun\_behavior [strong]

Enumerator

Discard	
Ignore	

## 4.31 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 >**

### 4.31.1 Detailed Description

Namespace where all classes of the matlab wrapper plugin live.

## 4.32 matrixmixer Namespace Reference

### Classes

- class **cfg\_t**
- class **matmix\_t**

## 4.33 mconv Namespace Reference

### Classes

- class **MConv**

## 4.34 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.*

### 4.34.1 Detailed Description

Functions and classes for Algorithm Communication (AC) support.

## 4.35 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.*

### 4.35.1 Function Documentation

#### 4.35.1.1 **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> .
----------	---

**4.35.1.2 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

## 4.36 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. 799) handles Communication between client and server, is used on both sides.
- class **Event\_Watcher**  
*OS-independent event watcher, uses select on Unix and WaitForMultipleObjects on Windows.*
- struct **OS\_EVENT\_TYPE**
- class **Server**
- class **sock\_init\_t**
- class **Sockaccept\_Event**

- class **Sockread\_Event**  
*Watch socket for incoming data.*
- class **Sockwrite\_Event**
- class **Thread**  
*A very simple class for portable threads.*
- class **Timeout\_Event**
- class **Timeout\_Watcher**  
*OS-independent event watcher with internal fixed-end-time timeout.*
- class **Wakeup\_Event**  
*A base class for asynchronous wakeup events.*

## TypeDefs

- typedef int **SOCKET**

## Functions

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

## Variables

- class **MHA\_TCP::sock\_init\_t** **sock\_initializer**

### 4.36.1 Detailed Description

A Namespace for TCP helper classes.

## 4.36.2 Typedef Documentation

### 4.36.2.1 SOCKET `typedef int MHA_TCP::SOCKET`

## 4.36.3 Function Documentation

### 4.36.3.1 STRERROR() `std::string MHA_TCP::STRERROR (int err)`

Portable conversion from error number to error string.

### 4.36.3.2 HSTRERROR() `std::string MHA_TCP::HSTRERROR (int err)`

Portable conversion from hostname error number to error string.

### 4.36.3.3 N\_ERRNO() `int MHA_TCP::N_ERRNO ( )`

Portable access to last network error number.

### 4.36.3.4 H\_ERRNO() `int MHA_TCP::H_ERRNO ( )`

Portable access to last hostname error number.

**4.36.3.5 G\_ERRNO()** `int MHA_TCP::G_ERRNO ( )`

Portable access to last non-network error number.

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

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

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

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

**4.36.3.8 stime()** `struct timeval MHA_TCP::stime ( double d )`

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

**4.36.4 Variable Documentation****4.36.4.1 sock\_initializer** `class MHA_TCP::sock_init_t MHA_TCP::sock_initializer`**4.37 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.*

#### 4.37.1 Detailed Description

namespace for network communication classes of MHA

### 4.38 mhachain Namespace Reference

#### Classes

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

### 4.39 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.*

#### 4.39.1 Detailed Description

Collection of event handling classes.

### 4.40 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.*

### 4.40.1 Detailed Description

Namespace for IIR and FIR filter classes.

### 4.40.2 Function Documentation

**4.40.2.1 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]

**4.40.2.2 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

**4.40.2.3 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

**4.40.2.4 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

```
4.40.2.5 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 MHAWindow::hanning(54)).
<code>minphase</code>	Flag, true if original phase should be discarded and replaced by a minimal phase function.

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

greatest common divisor

**4.40.2.7 `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.

**4.40.2.8 `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. [760](#)) if no rational resampling factor can be found.

## 4.41 MHAIOJack Namespace Reference

JACK IO.

## Classes

- class **io\_jack\_t**  
*Main class for JACK IO.*

### 4.41.1 Detailed Description

JACK IO.

## 4.42 MHAIOJackdb Namespace Reference

## Classes

- class **io\_jack\_t**  
*Main class for JACK IO.*

## 4.43 MHAIOPortAudio Namespace Reference

## Classes

- class **device\_info\_t**
- class **io\_portaudio\_t**  
*Main class for Portaudio sound IO.*
- class **stream\_info\_t**

## Functions

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

### 4.43.1 Function Documentation

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

---

## 4.44 mhaioutils Namespace Reference

### Functions

- template<typename T >  
T **to\_int\_clamped** (float val)

#### 4.44.1 Function Documentation

**4.44.1.1 to\_int\_clamped()** template<typename T >  
T mhaioutils::to\_int\_clamped (  
    float val )

## 4.45 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.*

#### 4.45.1 Detailed Description

Classes and functions for openMHA and JACK interaction.

#### 4.45.2 Function Documentation

```
4.45.2.1 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.

```
4.45.2.2 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)

```
4.45.2.3 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)

```
4.45.2.4 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)

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

## 4.46 MHAKernel Namespace Reference

### Classes

- class **algo\_comm\_class\_t**
- class **comm\_var\_map\_t**

### Functions

- **algo\_comm\_class\_t \* algo\_comm\_safe\_cast (void \*)**

#### 4.46.1 Function Documentation

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

## 4.47 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**

#### 4.47.1 Detailed Description

Collection of classes for selecting audio chunks from multiple sources.

## 4.48 MHAOvlFilter Namespace Reference

Namespace for overlapping FFT based filter bank classes and functions.

### Namespaces

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

## Classes

- class **band\_descriptor\_t**
- class **fftfb\_ac\_info\_t**
- class **fftfb\_t**  
*FFT based overlapping filter bank.*
- class **fftfb\_vars\_t**  
*Set of configuration variables for FFT-based overlapping filters.*
- class **fscale\_bw\_t**
- class **fscale\_t**
- class **fspacing\_t**  
*Class for frequency spacing, used by filterbank shape generator class.*
- class **overlap\_save\_filterbank\_analytic\_t**
- class **overlap\_save\_filterbank\_t**  
*A time-domain minimal phase filter bank with frequency shapes from **MHAOvlFilter::fftfb\_t** (p. 1002).*
- class **scale\_var\_t**

## Typedefs

- typedef **mha\_real\_t()** **scale\_fun\_t(mha\_real\_t)**

### 4.48.1 Detailed Description

Namespace for overlapping FFT based filter bank classes and functions.

### 4.48.2 Typedef Documentation

#### 4.48.2.1 **scale\_fun\_t** `typedef mha_real_t() MHAOvlFilter::scale_fun_t( mha_real_t )`

## 4.49 MHAOvlFilter::barkscale Namespace Reference

## Classes

- class **bark2hz\_t**
- class **hz2bark\_t**

## Variables

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

### 4.49.1 Variable Documentation

#### 4.49.1.1 `vfreq mha_real_t MHAOvlFilter::barkscale::vfreq`

#### 4.49.1.2 `vbark mha_real_t MHAOvlFilter::barkscale::vbark`

## 4.50 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))`

### 4.50.1 Detailed Description

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

## 4.50.2 Function Documentation

### 4.50.2.1 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

### 4.50.2.2 hz2khz() `mha_real_t MHAOvlFilter::FreqScaleFun::hz2khz (mha_real_t x )`

### 4.50.2.3 hz2octave() `mha_real_t MHAOvlFilter::FreqScaleFun::hz2octave (mha_real_t x )`

### 4.50.2.4 hz2third\_octave() `mha_real_t MHAOvlFilter::FreqScaleFun::hz2third_octave (mha_real_t x )`

### 4.50.2.5 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

**4.50.2.6 hz2bark\_analytic()** `mha_real_t MHAOvlFilter::FreqScaleFun::hz2bark_analytic ( mha_real_t x )`

**4.50.2.7 hz2erb()** `mha_real_t MHAOvlFilter::FreqScaleFun::hz2erb ( mha_real_t x )`

**4.50.2.8 hz2erb\_glasberg1990()** `mha_real_t MHAOvlFilter::FreqScaleFun::hz2erb_glasberg1990 ( mha_real_t x )`

**4.50.2.9 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

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

**4.51 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)**

**4.51.1 Detailed Description**

Shape functions for overlapping filters.

**4.51.2 Function Documentation**

```
4.51.2.1 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]

**4.51.2.2 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]

**4.51.2.3 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]

**4.51.2.4 `expflt()`** `mha_real_t MHAOvlFilter::ShapeFun::expflt ( mha_real_t x )`

**4.51.2.5 `gauss()`** `mha_real_t MHAOvlFilter::ShapeFun::gauss ( mha_real_t x )`

## 4.52 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)

### 4.52.1 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).

### 4.52.2 List of valid MHAParser items

- **Sub-parser:** `parser_t` (p. 1098)
- **Variables:**  
Numeric variables: `int_t` (p. 1065), `vint_t` (p. 1128), `float_t` (p. 1059), `vfloat_t` (p. 1123),  
`mfloat_t` (p. 1081)  
Other variables: `string_t` (p. 1111), `vstring_t` (p. 1132), `kw_t` (p. 1071), `bool_t` (p. 1043)
- **Monitors:**  
Numeric monitors: `int_mon_t` (p. 1062), `vint_mon_t` (p. 1126), `float_mon_t` (p. 1057),  
`vfloat_mon_t` (p. 1121)  
`mfloat_mon_t` (p. 1079)  
`mcomplex_mon_t` (p. 1075)  
Other monitors: `bool_mon_t` (p. 1041), `string_mon_t` (p. 1109), `vstring_mon_t` (p. 1130)

Members can be inserted into the configuration namespace by using `MHAParser::insert_item()` or the `insert_member()` (p. 1617) macro.

### 4.52.3 Typedef Documentation

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

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

**4.52.3.3 opact\_map\_t** `typedef std::map<std::string, opact_t> MHAParser::opact_map_t`

**4.52.3.4 query\_map\_t** `typedef std::map<std::string, query_t> MHAParser::query_map_t`

**4.52.3.5 entry\_map\_t** `typedef std::list<entry_t> MHAParser::entry_map_t`

**4.52.3.6 c\_parse\_cmd\_t** `typedef int(* MHAParser::c_parse_cmd_t) (void *, const char *, char *, unsigned int)`

## 4.52.4 Function Documentation

**4.52.4.1 get\_precision()** `int MHAParser::get_precision ( )`

**4.52.4.2 commentate()** `std::string MHAParser::commentate (const std::string & s )`

**4.52.4.3 trim()** `void MHAParser::trim (std::string & s )`

**4.52.4.4 cfg\_dump()** `std::string MHAParser::cfg_dump (base_t * p, const std::string & pref )`

---

**4.52.4.5 cfg\_dump\_short()** std::string MHAParser::cfg\_dump\_short (

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

**4.52.4.6 all\_dump()** std::string MHAParser::all\_dump (

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

**4.52.4.7 mon\_dump()** std::string MHAParser::mon\_dump (

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

**4.52.4.8 all\_ids()** std::string MHAParser::all\_ids (

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

**4.52.4.9 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

---

**4.52.4.10 envreplace()** void MHAParser::envreplace (

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

## 4.52.5 Variable Documentation

**4.52.5.1 c\_parse\_err\_t** const typedef char\*(\* MHAParser::c\_parse\_err\_t) (void \*, int)

## 4.53 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.*

#### 4.53.1 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.

#### 4.53.2 Function Documentation

##### 4.53.2.1 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

**4.53.2.2 bracket\_balance()** int MHAParser::StrCnv::bracket\_balance ( const std::string & s )

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

Convert from string.

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

Convert from string.

**4.53.2.5 str2val() [3/8]** void MHAParser::StrCnv::str2val ( const std::string & s, mha\_complex\_t & v )

Convert from string.

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

Convert from string.

**4.53.2.7 str2val() [5/8]** void MHAParser::StrCnv::str2val ( const std::string & s, **MHAParser::keyword\_list\_t** & v )

Convert from string.

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

Convert from string.

**4.53.2.9 str2val() [7/8]** template<class arg\_t > void MHAParser::StrCnv::str2val ( const std::string & s, std::vector< arg\_t > & val )

Converter for vector types.

**4.53.2.10 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.

**4.53.2.11 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.

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

Convert to string.

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

Convert to string.

**4.53.2.14 val2str()** [3/13] std::string MHAParser::StrCnv::val2str ( const mha\_complex\_t & v )

Convert to string.

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

Convert to string.

**4.53.2.16 val2str()** [5/13] std::string MHAParser::StrCnv::val2str ( const keyword\_list\_t & v )

Convert to string.

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

Convert to string.

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

Convert to string.

**4.53.2.19 val2str()** [8/13] std::string MHAParser::StrCnv::val2str ( const std::vector< mha\_complex\_t > & v )

Convert to string.

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

Convert to string.

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

Convert to string.

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

Convert to string.

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

Convert to string.

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

Convert to string.

## 4.54 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.*

### 4.54.1 Detailed Description

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

## 4.55 MHAPlugin\_Resampling Namespace Reference

### Classes

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

## 4.56 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. 1179) 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.*

#### 4.56.1 Detailed Description

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

#### 4.56.2 Enumeration Type Documentation

##### 4.56.2.1 anonymous enum anonymous enum

Invalid thread priority.

## Enumerator

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

**4.57 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. 790))*
- 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. 836))*

## 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. 836).*
- **mha\_real\_t lin2db** ( **mha\_real\_t** x, **mha\_real\_t** eps)  
*Conversion from linear scale to dB (no SPL reference)*
- **mha\_real\_t lin2db** ( **mha\_real\_t** x)  
*Conversion from linear scale to dB (no SPL reference)*
- **mha\_real\_t db2lin** ( **mha\_real\_t** x)  
*Conversion from dB scale to linear (no SPL reference)*
- **mha\_real\_t sq2db** ( **mha\_real\_t** x, **mha\_real\_t** eps=0.0f)  
*conversion from squared values to dB (no SPL reference)*
- **mha\_real\_t db2sq** ( **mha\_real\_t** x)  
*conversion from dB to squared values (no SPL reference)*
- **mha\_real\_t pa2dbspl** ( **mha\_real\_t** x, **mha\_real\_t** eps)  
*Conversion from linear Pascal scale to dB SPL.*
- **mha\_real\_t pa2dbspl** ( **mha\_real\_t** x)  
*Conversion from linear Pascal scale to dB SPL.*
- **mha\_real\_t dbspl2pa** ( **mha\_real\_t** x)  
*Conversion from dB SPL to linear Pascal scale.*
- **mha\_real\_t pa22dbspl** ( **mha\_real\_t** x, **mha\_real\_t** eps=0.0f)  
*Conversion from squared Pascal scale to dB SPL.*
- **mha\_real\_t dbspl2pa2** ( **mha\_real\_t** x)  
*conversion from dB SPL to squared Pascal scale*
- **mha\_real\_t smp2sec** ( **mha\_real\_t** n, **mha\_real\_t** srate)  
*conversion from samples to seconds*
- **mha\_real\_t sec2smp** ( **mha\_real\_t** sec, **mha\_real\_t** srate)  
*conversion from seconds to samples*
- **mha\_real\_t bin2freq** ( **mha\_real\_t** bin, unsigned fftlen, **mha\_real\_t** srate)  
*conversion from fft bin index to frequency*
- **mha\_real\_t freq2bin** ( **mha\_real\_t** freq, unsigned fftlen, **mha\_real\_t** srate)  
*conversion from frequency to fft bin index*
- **mha\_real\_t smp2rad** ( **mha\_real\_t** samples, unsigned bin, unsigned fftlen)  
*conversion from delay in samples to phase shift*
- **mha\_real\_t rad2smp** ( **mha\_real\_t** phase\_shift, unsigned bin, unsigned fftlen)  
*conversion from phase shift to delay in samples*
- template<class elem\_type >  
**std::vector< elem\_type > dupvec** (std::vector< elem\_type > vec, unsigned n)  
*Duplicate last vector element to match desired size.*
- template<class elem\_type >  
**std::vector< elem\_type > dupvec\_chk** (std::vector< elem\_type > vec, unsigned n)  
*Duplicate last vector element to match desired size, check for dimension.*
- void **copy\_channel** ( **mha\_spec\_t** &self, const **mha\_spec\_t** &src, unsigned sch, unsigned dch)  
*Copy one channel of a source signal.*

- void **copy\_channel** ( **mha\_wave\_t** &self, const **mha\_wave\_t** &src, unsigned src\_channel, unsigned dest\_channel)
 

*Copy one channel of a source signal.*
- **mha\_real\_t rmslevel** (const **mha\_spec\_t** &s, unsigned int channel, unsigned int fftlen)
 

*Return RMS level of a spectrum channel.*
- **mha\_real\_t colored\_intensity** (const **mha\_spec\_t** &s, unsigned int channel, unsigned int fftlen, **mha\_real\_t** \*sqfreq\_response=nullptr)
 

*Colored spectrum intensity.*
- **mha\_real\_t maxabs** (const **mha\_spec\_t** &s, unsigned int channel)
 

*Find maximal absolute value.*
- **mha\_real\_t rmslevel** (const **mha\_wave\_t** &s, unsigned int channel)
 

*Return RMS level of a waveform channel.*
- **mha\_real\_t maxabs** (const **mha\_wave\_t** &s, unsigned int channel)
 

*Find maximal absolute value.*
- **mha\_real\_t maxabs** (const **mha\_wave\_t** &s)
 

*Find maximal absolute value.*
- **mha\_real\_t max** (const **mha\_wave\_t** &s)
 

*Find maximal value.*
- **mha\_real\_t min** (const **mha\_wave\_t** &s)
 

*Find minimal value.*
- **mha\_real\_t sumsqr\_channel** (const **mha\_wave\_t** &s, unsigned int channel)
 

*Calculate sum of squared values in one channel.*
- **mha\_real\_t sumsqr\_frame** (const **mha\_wave\_t** &s, unsigned int frame)
 

*Calculate sum over all channels of squared values.*
- void **scale** ( **mha\_spec\_t** \*dest, const **mha\_wave\_t** \*src)
- void **limit** ( **mha\_wave\_t** &s, const **mha\_real\_t** &min, const **mha\_real\_t** &max)
 

*Limit the singal in the waveform buffer to the range [min, max].*
- template<class elem\_type >
 elem\_type **kth\_smallest** (elem\_type array[], unsigned n, unsigned k)
 

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

*Fast median search.*
- template<class elem\_type >
 elem\_type **mean** (const std::vector< elem\_type > &data, elem\_type start\_val)
 

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

*Calculate quantile of elements in a vector.*
- void **saveas\_mat4** (const **mha\_spec\_t** &data, const std::string &varname, FILE \*fh)
 

*Save a openMHA spectrum as a variable in a Matlab4 file.*
- void **saveas\_mat4** (const **mha\_wave\_t** &data, const std::string &varname, FILE \*fh)
 

*Save a openMHA waveform as a variable in a Matlab4 file.*
- void **saveas\_mat4** (const std::vector< **mha\_real\_t** > &data, const std::string &varname, FILE \*fh)

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

- void **copy\_permuted** ( **mha\_wave\_t** \*dest, const **mha\_wave\_t** \*src)  
*Copy contents of a waveform to a permuted waveform.*

## Variables

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

### 4.57.1 Detailed Description

Namespace for audio signal handling and processing classes.

### 4.57.2 Function Documentation

**4.57.2.1 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. 836).

#### Parameters

<i>s</i>	Pointer to a <b>mha_wave_t</b> (p. 836) structure
<i>fun</i>	Function to be applied (one argument)

**4.57.2.2 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**

<b>MHA_Error</b> (p. <a href="#">760</a> )	if $\text{eps} < 0$
--	---------------------

**4.57.2.3 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)

**4.57.2.4 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.
----------	-----------

```
4.57.2.5 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

<b>MHA_Error</b> (p. <a href="#">760</a> )	if <i>eps</i> < 0
--	-------------------

```
4.57.2.6 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
----------	----------

```
4.57.2.7 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 &gt;= 0</code>
--------------------------

**Returns**

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

**Exceptions**

<b>MHA_Error</b> (p. 760)	if $\text{eps} < 0$
---------------------------	---------------------

**4.57.2.8 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)

**4.57.2.9 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.
----------------	---------------

**4.57.2.10 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

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

<i>MHA_Error</i> (p. <a href="#">760</a> )	if <i>eps</i> < 0
--	-------------------

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

conversion from dB SPL to squared Pascal scale

#### Parameters

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

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

conversion from samples to seconds

#### Parameters

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

---

**4.57.2.13 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

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

**4.57.2.15 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

**4.57.2.16 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

```
4.57.2.17 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

```
4.57.2.18 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

---

**4.57.2.19 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.

**4.57.2.20 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.

**4.57.2.21 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.

---

**4.57.2.22 `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.

**4.57.2.23 `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.

### 4.57.3 Variable Documentation

**4.57.3.1 `signal_counter`** unsigned long int MHASignal::signal\_counter = 0

Signal counter to produce signal ID strings.

## 4.58 MHASndFile Namespace Reference

### Classes

- class **sf\_t**
- class **sf\_wave\_t**

## 4.59 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.*

### 4.59.1 Detailed Description

Namespace for table lookup classes.

## 4.60 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.*

### 4.60.1 Function Documentation

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

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

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

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

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

**4.60.1.6 `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

**4.60.1.7 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. 741). Returns true iff one or both of real and imaginary part are denormal

**Parameters**

<i>x</i>	[in] A mha_complex_t (p. 741) number
----------	--------------------------------------

**Returns**

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

**4.60.1.8 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. 741) number
----------	--------------------------------------

**Returns**

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

**4.60.1.9 `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**

<b>MHA_Error</b> (p. <a href="#">760</a> )	if $f < 0$
--	------------

## 4.61 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.*

### 4.61.1 Detailed Description

Collection of Window types.

### 4.61.2 Function Documentation

#### 4.61.2.1 **rect()** float MHAWindow::rect ( float x )

Rectangular window function.

**4.61.2.2 bartlett()** float MHAWindow::bartlett ( float x )

Bartlett window function.

**4.61.2.3 hanning()** float MHAWindow::hanning ( float x )

Hanning window function.

**4.61.2.4 hamming()** float MHAWindow::hamming ( float x )

Hamming window function.

**4.61.2.5 blackman()** float MHAWindow::blackman ( float x )

Blackman window function.

## 4.62 multibandcompressor Namespace Reference

### Classes

- class **fftfb\_plug\_t**
- class **interface\_t**
- class **plugin\_signals\_t**

## 4.63 noise\_psd\_estimator Namespace Reference

### Classes

- class **noise\_psd\_estimator\_if\_t**
- class **noise\_psd\_estimator\_t**

## 4.64 overlapadd Namespace Reference

### Classes

- class **overlapadd\_if\_t**
- class **overlapadd\_t**

## 4.65 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)

#### 4.65.1 Detailed Description

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

#### 4.65.2 Function Documentation

##### 4.65.2.1 **drand()** double plingploing::drand (

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

## 4.66 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. 847) structures, and report differences as an error.*

#### 4.66.1 Function Documentation

**4.66.1.1 mhastrdomain()** const char \* PluginLoader::mhastrdomain ( **mha\_domain\_t** d )

**4.66.1.2 mhaconfig\_compare()** void PluginLoader::mhaconfig\_compare ( const **mhaconfig\_t** & req, const **mhaconfig\_t** & avail, const std::string & pref = "" )

Compare two **mhaconfig\_t** (p. 847) structures, and report differences as an error.

#### Parameters

<i>req</i>	Expected <b>mhaconfig_t</b> (p. 847) structure
<i>avail</i>	Available <b>mhaconfig_t</b> (p. 847) structure
<i>pref</i>	Prefix for error messages

## 4.67 plugins Namespace Reference

### Namespaces

- **hoertech**

## 4.68 plugins::hoertech Namespace Reference

### Namespaces

- **acrec**

## 4.69 plugins::hoertech::acrec Namespace Reference

### Classes

- class **acrec\_t**  
*Plugin interface class of plugin acrec.*
- class **acwriter\_t**  
*acwriter\_t* (p. 1378) decouples signal processing from writing to disk.

### Functions

- std::string **to\_iso8601** (time\_t tm)

### 4.69.1 Function Documentation

**4.69.1.1 to\_iso8601()** std::string plugins::hoertech::acrec::to\_iso8601 ( time\_t tm )

## 4.70 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 }

### 4.70.1 Enumeration Type Documentation

#### 4.70.1.1 **UNIT** enum **rmslevel::UNIT** [strong]

##### Enumerator

SPL	
HL	

## 4.71 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

### 4.71.1 Function Documentation

#### 4.71.1.1 **j0()** double rohBeam::j0 ( double x )

Cylindrical bessel function of the first kind of order 0.

**Parameters**

x	the argument of the function
---	------------------------------

**Returns** $j_0(x)$ **4.71.2 Variable Documentation****4.71.2.1 scalarify** auto rohBeam::scalarify = [ ](auto t){return t(0);}**4.71.2.2 CONST\_C** constexpr float rohBeam::CONST\_C = 343.0115f [constexpr]**4.71.2.3 refL** constexpr int rohBeam::refL = 0 [constexpr]**4.71.2.4 refR** constexpr int rohBeam::refR = 3 [constexpr]**4.72 route Namespace Reference****Classes**

- class **interface\_t**
- class **process\_t**

**4.73 shadowfilter\_begin Namespace Reference****Classes**

- class **cfg\_t**
- class **shadowfilter\_begin\_t**

## 4.74 shadowfilter\_end Namespace Reference

### Classes

- class `cfg_t`
- class `shadowfilter_end_t`

## 4.75 smooth\_cepstrum Namespace Reference

### Classes

- class `smooth_cepstrum_if_t`
- class `smooth_cepstrum_t`
- class `smooth_params`

## 4.76 smoothgains\_bridge Namespace Reference

### Classes

- class `overlapadd_if_t`
- class `smoothspec_wrap_t`

## 4.77 testplugin Namespace Reference

### Classes

- class `ac_parser_t`
- class `config_parser_t`
- class `if_t`
- class `signal_parser_t`

## 4.78 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*

#### 4.78.1 Detailed Description

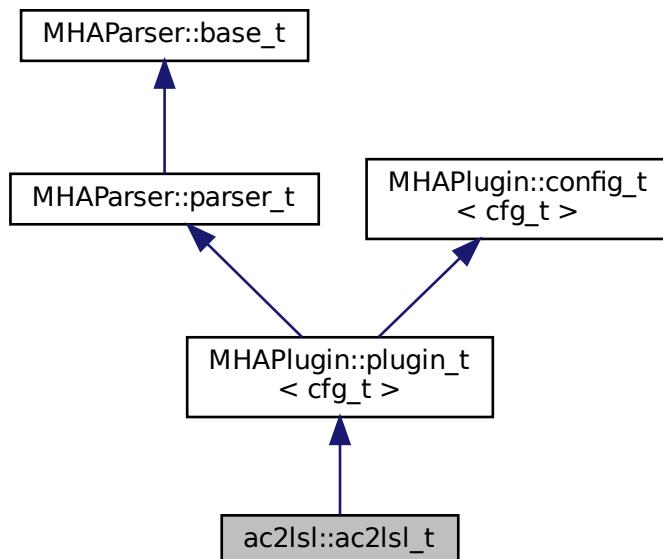
namespace for plugin windnoise which detects and cancels wind noise

## 5 Class Documentation

### 5.1 ac2lsl::ac2lsl\_t Class Reference

Plugin class of **ac2lsl** (p. 78).

Inheritance diagram for ac2lsl::ac2lsl\_t:



#### Public Member Functions

- **ac2lsl\_t ( algo\_comm\_t iac, const std::string &configured\_name)**
- **void prepare ( mhaconfig\_t &)**  
*Prepare constructs the vector of bridge variables and locks the configuration, then calls **update()** (p. 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

### 5.1.1 Detailed Description

Plugin class of **ac2lsl** (p. 78).

### 5.1.2 Constructor & Destructor Documentation

#### 5.1.2.1 ac2lsl\_t() ac2lsl::ac2lsl\_t::ac2lsl\_t (

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

### 5.1.3 Member Function Documentation

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

Prepare constructs the vector of bridge variables and locks the configuration, then calls **update()** (p. 165).

Implements **MHAPlugin::plugin\_t< cfg\_t >** (p. 1149).

**5.1.3.2 `process() [1/3]`** `mha_wave_t* ac2lsl::ac2lsl_t::process (`  
 `mha_wave_t * s ) [inline]`

Processing fct for waveforms.

Calls **process(void)** (p. 164).

**5.1.3.3 `process() [2/3]`** `mha_spec_t* ac2lsl::ac2lsl_t::process (`  
 `mha_spec_t * s ) [inline]`

Processing fct for spectra.

Calls **process(void)** (p. 164).

**5.1.3.4 `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).

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

Release fct.

Unlocks variable name list

Reimplemented from **MHAPlugin::plugin\_t< cfg\_t >** (p. 1150).

**5.1.3.6 `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

**5.1.3.7 update()** `void ac2lsl::ac2lsl_t::update ( ) [private]`

Construct new runtime configuration.

**5.1.4 Member Data Documentation****5.1.4.1 vars** `MHAParser::vstring_t ac2lsl::ac2lsl_t::vars [private]`**5.1.4.2 source\_id** `MHAParser::string_t ac2lsl::ac2lsl_t::source_id [private]`**5.1.4.3 rt\_strict** `MHAParser::bool_t ac2lsl::ac2lsl_t::rt_strict [private]`**5.1.4.4 activate** `MHAParser::bool_t ac2lsl::ac2lsl_t::activate [private]`**5.1.4.5 skip** `MHAParser::int_t ac2lsl::ac2lsl_t::skip [private]`

**5.1.4.6 patchbay** `MHAEEvents::patchbay_t< ac2lsl_t> ac2lsl::ac2lsl_t::patchbay`  
[private]

**5.1.4.7 is\_first\_run** `bool ac2lsl::ac2lsl_t::is_first_run` [private]

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

- `ac2lsl.cpp`

## 5.2 ac2lsl::cfg\_t Class Reference

Runtime configuration class of the **ac2lsl** (p. 78) plugin.

### Public Member Functions

- `cfg_t (const algo_comm_t &ac_, unsigned skip_, const std::string &source_id, const std::vector< std::string > &varnames_, double rate)`  
*C'tor of ac2lsl (p. 78) run time configuration.*
- `void process ()`

### Private Member Functions

- `void create_or_replace_var (const std::string &name, const comm_var_t &v)`
- `void check_vars ()`
- `void update_varlist ()`

### Private Attributes

- `std::map< std::string, std::unique_ptr< save_var_base_t > > varlist`  
*Maps variable name to unique ptr's of ac to 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.*

### 5.2.1 Detailed Description

Runtime configuration class of the **ac2lsl** (p. 78) plugin.

### 5.2.2 Constructor & Destructor Documentation

```
5.2.2.1 cfg_t() cfg_t::cfg_t (
    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>

### 5.2.3 Member Function Documentation

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

**5.2.3.2 `check_vars()`** void ac2lsl::cfg\_t::check\_vars ( ) [private]

**5.2.3.3 `update_varlist()`** void cfg\_t::update\_varlist ( ) [private]

**5.2.3.4 `process()`** void cfg\_t::process ( )

## 5.2.4 Member Data Documentation

**5.2.4.1 `varlist`** std::map<std::string, std::unique\_ptr< **save\_var\_base\_t**> > ac2lsl::cfg\_t::varlist [private]

Maps variable name to unique ptr's of ac to Isl bridges.

**5.2.4.2 `skipcnt`** unsigned ac2lsl::cfg\_t::skipcnt [private]

Counter of frames to skip.

**5.2.4.3 `skip`** const unsigned ac2lsl::cfg\_t::skip [private]

Number of frames to skip after each send.

**5.2.4.4 `srate`** const double ac2lsl::cfg\_t::srate [private]

Sampling rate of the stream.

#### 5.2.4.5 source\_id const std::string ac2lsl::cfg\_t::source\_id [private]

User configurable source id.

#### 5.2.4.6 ac const algo\_comm\_t& ac2lsl::cfg\_t::ac [private]

Handle to the ac space.

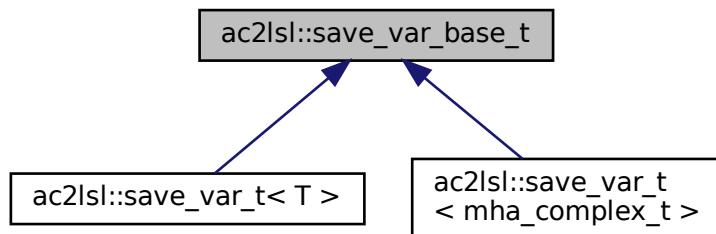
The documentation for this class was generated from the following file:

- [ac2lsl.cpp](#)

## 5.3 ac2lsl::save\_var\_base\_t Class Reference

Interface for ac to lsl bridge variable.

Inheritance diagram for ac2lsl::save\_var\_base\_t:



## Public Member Functions

- virtual void **send\_frame** ()=0
- virtual void \* **get\_buf\_address** () const noexcept=0
- virtual void **set\_buf\_address** (void \*data)=0
- virtual lsl::stream\_info **info** () const noexcept=0
- virtual unsigned **data\_type** () const noexcept=0
- virtual unsigned **num\_entries** () const noexcept=0
- virtual ~**save\_var\_base\_t** ()=default

### 5.3.1 Detailed Description

Interface for ac to Isl bridge variable.

### 5.3.2 Constructor & Destructor Documentation

**5.3.2.1 ~save\_var\_base\_t()** virtual ac2isl::save\_var\_base\_t::~save\_var\_base\_t ( )  
[virtual], [default]

### 5.3.3 Member Function Documentation

**5.3.3.1 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).

**5.3.3.2 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).

**5.3.3.3 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).

**5.3.3.4 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).

**5.3.3.5 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).

**5.3.3.6 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).

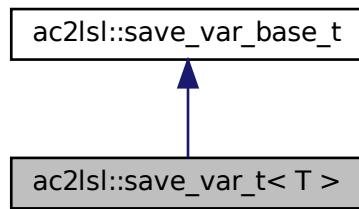
The documentation for this class was generated from the following file:

- **ac2isl.cpp**

## 5.4 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.*

### 5.4.1 Detailed Description

```
template<typename T>
class ac2isl::save_var_t< T >
```

Implementation for all ac to Isl bridges except complex types.

### 5.4.2 Constructor & Destructor Documentation

```
5.4.2.1 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.

```
5.4.2.2 ~save_var_t() template<typename T >
virtual ac2isl::save_var_t< T >::~ save_var_t ( ) [virtual], [default]
```

### 5.4.3 Member Function Documentation

```
5.4.3.1 get_buf_address() template<typename T >
virtual void* 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).

```
5.4.3.2 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 `ac2isl::save_var_base_t` (p. 170).

**5.4.3.3 info()** template<typename T >  
 virtual lsl::stream\_info `ac2isl::save_var_t< T >::info` ( ) const [inline], [override], [virtual], [noexcept]

Get stream info object from stream outlet.

Implements `ac2isl::save_var_base_t` (p. 170).

**5.4.3.4 num\_entries()** template<typename T >  
 virtual unsigned `ac2isl::save_var_t< T >::num_entries` ( ) const [inline], [override], [virtual], [noexcept]

Get number of entries in the stream object.

Implements `ac2isl::save_var_base_t` (p. 171).

**5.4.3.5 data\_type()** template<typename T >  
 virtual unsigned `ac2isl::save_var_t< T >::data_type` ( ) const [inline], [override], [virtual], [noexcept]

Get data type id according MHA convention.

Implements `ac2isl::save_var_base_t` (p. 171).

**5.4.3.6 send\_frame()** template<typename T >  
 virtual void `ac2isl::save_var_t< T >::send_frame` ( ) [inline], [override], [virtual]

Send a frame to Isl.

Implements `ac2isl::save_var_base_t` (p. 170).

#### 5.4.4 Member Data Documentation

**5.4.4.1 stream** template<typename T >  
`lsl::stream_outlet ac2lsl::save_var_t< T >::stream [private]`

LSL stream outlet.

Interface to lsl

**5.4.4.2 buf** template<typename T >  
`T* ac2lsl::save_var_t< T >::buf [private]`

Pointer to data buffer of the ac variable.

**5.4.4.3 data\_type\_** template<typename T >  
`const unsigned ac2lsl::save_var_t< T >::data_type_ [private]`

Data type id according to MHA convention.

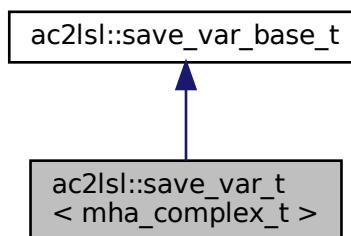
The documentation for this class was generated from the following file:

- `ac2lsl.cpp`

## 5.5 ac2lsl::save\_var\_t< mha\_complex\_t > Class Reference

Template specialization of the **ac2lsl** (p. 78) bridge to take care of complex numbers.

Inheritance diagram for ac2lsl::save\_var\_t< mha\_complex\_t >:



## Public Member Functions

- **save\_var\_t** (const std::string &name\_, const std::string &type\_, 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.*

### 5.5.1 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), ....]

### 5.5.2 Constructor & Destructor Documentation

```
5.5.2.1 save_var_t() ac2lsl::save_var_t< mha_complex_t >:: save_var_t (
    const std::string & name_,
    const std::string & type_,
    const unsigned num_entries_,
    const mha_real_t rate_,
    const lsl::channel_format_t format_,
    const std::string & source_id_,
    void * data_ ) [inline]
```

C'tor of specialization for complex types.

See generic c'tor for details.

```
5.5.2.2 ~save_var_t() virtual ac2lsl::save_var_t< mha_complex_t >::~ save_var_t
( ) [virtual], [default]
```

### 5.5.3 Member Function Documentation

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

Implements [ac2lsl::save\\_var\\_base\\_t](#) (p. 170).

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

Implements [ac2lsl::save\\_var\\_base\\_t](#) (p. 170).

```
5.5.3.3 info() virtual lsl::stream_info ac2lsl::save_var_t< mha_complex_t >::info
( ) const [inline], [override], [virtual], [noexcept]
```

Get buffer address as void pointer.

#### Returns

Adress of the data buffer

Implements [ac2lsl::save\\_var\\_base\\_t](#) (p. 170).

**5.5.3.4 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).

**5.5.3.5 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).

**5.5.3.6 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).

## 5.5.4 Member Data Documentation

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

LSL stream outlet.

Interface to lsl

**5.5.4.2 buf** `mha_complex_t* ac2lsl::save_var_t< mha_complex_t >::buf` [private]

Pointer to data buffer of the ac variable.

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

- `ac2lsl.cpp`

## 5.6 ac2lsl::type\_info Struct Reference

### Public Attributes

- `const std::string name`
- `const lsl::channel_format_t format`

### 5.6.1 Member Data Documentation

**5.6.1.1 name** `const std::string ac2lsl::type_info::name`

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

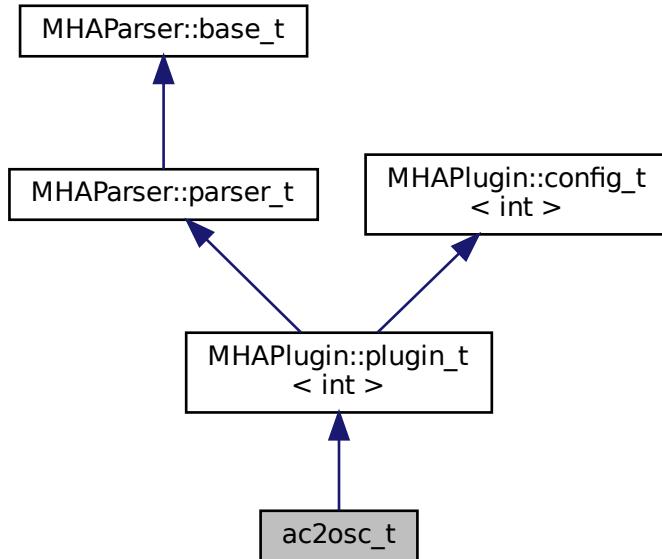
The documentation for this struct was generated from the following file:

- `ac2lsl.cpp`

## 5.7 ac2osc\_t Class Reference

Plugin class of the ac2osc plugin.

Inheritance diagram for ac2osc\_t:



### Public Member Functions

- **ac2osc\_t ( algo\_comm\_t iac, const std::string &configured\_name)**  
*C'tor of plugin class.*
- void **prepare ( mhaconfig\_t &)**
- **mha\_wave\_t \* process ( mha\_wave\_t \*s)**  
*Processing fct for waveforms.*
- **mha\_spec\_t \* process ( mha\_spec\_t \*s)**  
*Processing fct for spectra.*
- void **process ()**  
*Process function.*
- void **release ()**  
*Release frees osc related memory, does cleanup.*

### Private Member Functions

- void **send\_osc\_float ()**
- void **update\_mode ()**  
*Start/Stop sending of messages.*

## Private Attributes

- **MHAParser::string\_t host**  
*OSC server host name.*
- **MHAParser::string\_t port**  
*OSC server port.*
- **MHAParser::int\_t ttl**  
*Time-to-live of UDP packages.*
- **MHAParser::vstring\_t vars**  
*List of AC variables to be saved, empty for all.*
- **MHAParser::kw\_t mode**  
*Record mode.*
- **MHAParser::int\_t skip**  
*number of frames to skip after sending*
- **MHAParser::bool\_t rt\_strict**  
*abort if used in real-time thread?*
- std::unique\_ptr< **MHA\_AC::acspace2matrix\_t** > **acspace**
- **MHAEvents::patchbay\_t< ac2osc\_t > patchbay**
- bool **b\_record**
- uint8\_t \* **rtmem**
- float **framerate**
- int **skipcnt**
- lo\_address **lo\_addr**
- bool **is\_first\_run**

## Additional Inherited Members

### 5.7.1 Detailed Description

Plugin class of the ac2osc plugin.

### 5.7.2 Constructor & Destructor Documentation

#### 5.7.2.1 ac2osc\_t() ac2osc\_t::ac2osc\_t (

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

C'tor of plugin class.

### 5.7.3 Member Function Documentation

**5.7.3.1 `prepare()`** `void ac2osc_t::prepare ( mhaconfig_t & cf ) [virtual]`

Implements `MHAPlugin::plugin_t< int >` (p. [1149](#)).

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

Processing fct for waveforms.

Calls `process(void)` (p. [182](#)).

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

Processing fct for spectra.

Calls `process(void)` (p. [182](#)).

**5.7.3.4 `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.

**5.7.3.5 `release()`** `void ac2osc_t::release ( ) [virtual]`

Release frees osc related memory, does cleanup.

Reimplemented from `MHAPlugin::plugin_t< int >` (p. [1150](#)).

**5.7.3.6 send\_osc\_float()** void ac2osc\_t::send\_osc\_float ( ) [private]

**5.7.3.7 update\_mode()** void ac2osc\_t::update\_mode ( ) [private]

Start/Stop sending of messages.

#### 5.7.4 Member Data Documentation

**5.7.4.1 host** `MHAParser::string_t` ac2osc\_t::host [private]

OSC server host name.

**5.7.4.2 port** `MHAParser::string_t` ac2osc\_t::port [private]

OSC server port.

**5.7.4.3 ttl** `MHAParser::int_t` ac2osc\_t::ttl [private]

Time-to-live of UDP packages.

**5.7.4.4 vars** `MHAParser::vstring_t` ac2osc\_t::vars [private]

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

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

Record mode.

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

number of frames to skip after sending

**5.7.4.7 rt\_strict** `MHAParser::bool_t ac2osc_t::rt_strict [private]`

abort if used in real-time thread?

**5.7.4.8 acspace** `std::unique_ptr< MHA_AC::acspace2matrix_t> ac2osc_t::acspace [private]`**5.7.4.9 patchbay** `MHAEVENTS::patchbay_t< ac2osc_t> ac2osc_t::patchbay [private]`**5.7.4.10 b\_record** `bool ac2osc_t::b_record [private]`**5.7.4.11 rtmem** `uint8_t* ac2osc_t::rtmem [private]`**5.7.4.12 framerate** `float ac2osc_t::framerate [private]`

**5.7.4.13 skipcnt** int ac2osc\_t::skipcnt [private]

**5.7.4.14 lo\_addr** lo\_address ac2osc\_t::lo\_addr [private]

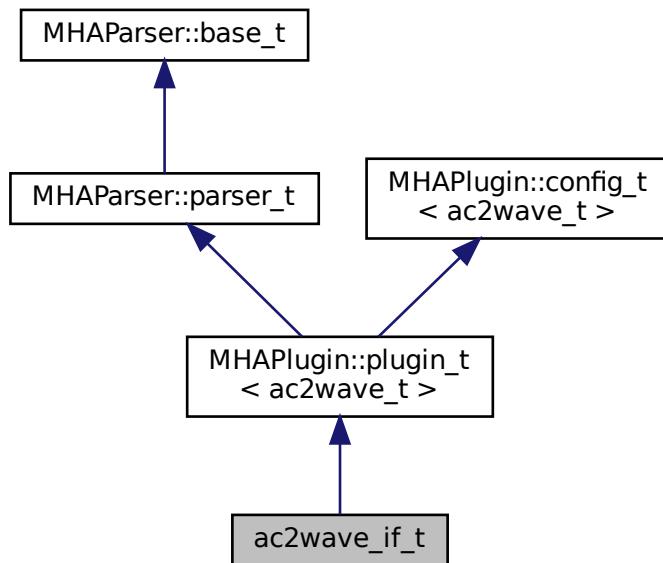
**5.7.4.15 is\_first\_run** bool ac2osc\_t::is\_first\_run [private]

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

- **ac2osc.cpp**

## 5.8 ac2wave\_if\_t Class Reference

Inheritance diagram for ac2wave\_if\_t:



## Public Member Functions

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

## Private Member Functions

- `void update ()`

## Private Attributes

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

## Additional Inherited Members

### 5.8.1 Constructor & Destructor Documentation

**5.8.1.1 `ac2wave_if_t()`** `ac2wave_if_t::ac2wave_if_t (`  
`algo_comm_t iac,`  
`const std::string & configured_name )`

### 5.8.2 Member Function Documentation

**5.8.2.1 process()** [1/2] `mha_wave_t * ac2wave_if_t::process ( mha_spec_t * )`

**5.8.2.2 process()** [2/2] `mha_wave_t * ac2wave_if_t::process ( mha_wave_t * s )`

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

Implements `MHAParser::plugin_t< ac2wave_t >` (p. 1149).

**5.8.2.4 release()** `void ac2wave_if_t::release ( ) [virtual]`

Reimplemented from `MHAParser::plugin_t< ac2wave_t >` (p. 1150).

**5.8.2.5 update()** `void ac2wave_if_t::update ( ) [private]`

### 5.8.3 Member Data Documentation

**5.8.3.1 name** `MHAParser::string_t ac2wave_if_t::name [private]`

**5.8.3.2 gain\_in** `MHAParser::float_t ac2wave_if_t::gain_in [private]`

**5.8.3.3 gain\_ac** `MHAParser::float_t ac2wave_if_t::gain_ac [private]`

**5.8.3.4 delay\_in** `MHAParser::int_t ac2wave_if_t::delay_in [private]`

**5.8.3.5 delay\_ac** `MHAParser::int_t ac2wave_if_t::delay_ac [private]`

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

**5.8.3.7 prepared** `bool ac2wave_if_t::prepared [private]`

**5.8.3.8 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**

## 5.9 ac2wave\_t Class Reference

### Public Member Functions

- **ac2wave\_t** (unsigned int frames\_, unsigned int channels\_, **algo\_comm\_t** ac\_, std::string name\_, float gain\_in\_, float gain\_ac\_, unsigned int delay\_in\_, unsigned int delay\_ac\_)
- **mha\_wave\_t \* process ( mha\_wave\_t \*)**

## Private Attributes

- unsigned int **frames**
- unsigned int **channels**
- **mha\_wave\_t w**
- **algo\_comm\_t ac**
- std::string **name**
- **MHASignal::delay\_wave\_t delay\_in**
- **MHASignal::delay\_wave\_t delay\_ac**
- **mha\_real\_t gain\_in**
- **mha\_real\_t gain\_ac**

### 5.9.1 Constructor & Destructor Documentation

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

### 5.9.2 Member Function Documentation

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

### 5.9.3 Member Data Documentation

```
5.9.3.1 frames unsigned int ac2wave_t::frames [private]
```

**5.9.3.2 channels** `unsigned int ac2wave_t::channels [private]`

**5.9.3.3 w** `mha_wave_t ac2wave_t::w [private]`

**5.9.3.4 ac** `algo_comm_t ac2wave_t::ac [private]`

**5.9.3.5 name** `std::string ac2wave_t::name [private]`

**5.9.3.6 delay\_in** `MHASignal::delay_wave_t ac2wave_t::delay_in [private]`

**5.9.3.7 delay\_ac** `MHASignal::delay_wave_t ac2wave_t::delay_ac [private]`

**5.9.3.8 gain\_in** `mha_real_t ac2wave_t::gain_in [private]`

**5.9.3.9 gain\_ac** `mha_real_t ac2wave_t::gain_ac [private]`

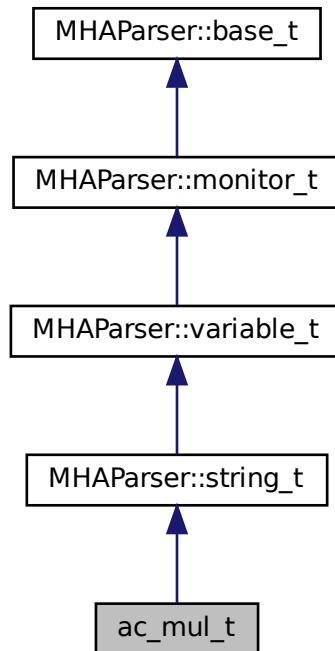
The documentation for this class was generated from the following file:

- **ac2wave.cpp**

## 5.10 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 iac, const std::string &configured\_name)**  
*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

### 5.10.1 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. 1111). 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.

### 5.10.2 Constructor & Destructor Documentation

```
5.10.2.1 ac_mul_t() ac_mul_t::ac_mul_t (
    algo_comm_t iac,
    const std::string & configured_name )
```

Plugin constructor.

### 5.10.3 Member Function Documentation

```
5.10.3.1 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.

```
5.10.3.2 release_() void ac_mul_t::release_ ( )
```

```
5.10.3.3 process() [1/3] mha_wave_t * ac_mul_t::process (
    mha_wave_t * s )
```

```
5.10.3.4 process() [2/3] mha_spec_t * ac_mul_t::process (
    mha_spec_t * s )
```

```
5.10.3.5 scan_syntax() void ac_mul_t::scan_syntax ( ) [private]
```

**5.10.3.6 `get_arg_type_and_dimension()` [1/2]** void ac\_mul\_t::get\_arg\_type\_and\_dimension ( ) [private]

**5.10.3.7 `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]

**5.10.3.8 `process()` [3/3]** void ac\_mul\_t::process ( ) [private]

**5.10.3.9 `process_rr()`** void ac\_mul\_t::process\_rr ( ) [private]

**5.10.3.10 `process_rc()`** void ac\_mul\_t::process\_rc ( ) [private]

**5.10.3.11 `process_cr()`** void ac\_mul\_t::process\_cr ( ) [private]

**5.10.3.12 `process_cc()`** void ac\_mul\_t::process\_cc ( ) [private]

## 5.10.4 Member Data Documentation

**5.10.4.1 `ac algo_comm_t`** ac\_mul\_t::ac [private]

**5.10.4.2 algo** std::string ac\_mul\_t::algo [private]

**5.10.4.3 argt** arg\_type\_t ac\_mul\_t::argt [private]

**5.10.4.4 str\_a** std::string ac\_mul\_t::str\_a [private]

**5.10.4.5 str\_b** std::string ac\_mul\_t::str\_b [private]

**5.10.4.6 res\_r** MHA\_AC::waveform\_t\* ac\_mul\_t::res\_r [private]

**5.10.4.7 res\_c** MHA\_AC::spectrum\_t\* ac\_mul\_t::res\_c [private]

**5.10.4.8 num\_frames** unsigned int ac\_mul\_t::num\_frames [private]

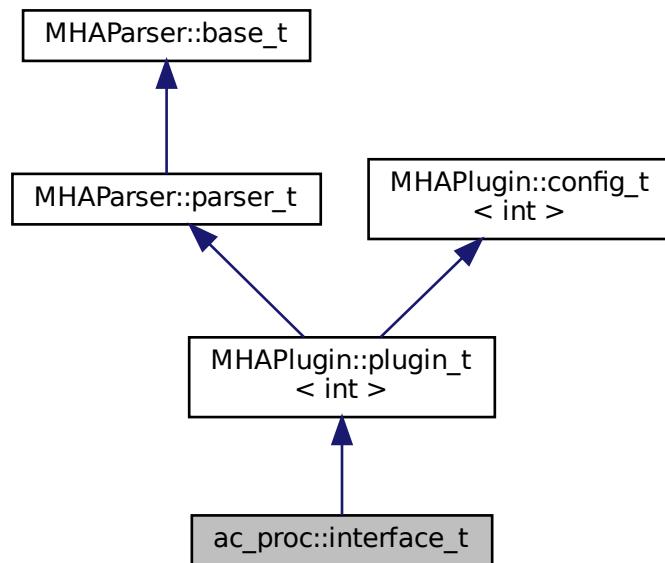
**5.10.4.9 num\_channels** unsigned int ac\_mul\_t::num\_channels [private]

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

- ac\_mul.hh
- ac\_mul.cpp

## 5.11 ac\_proc::interface\_t Class Reference

Inheritance diagram for ac\_proc::interface\_t:



### Public Member Functions

- `interface_t ( algo_comm_t iac, const std::string &configured_name)`
- `void prepare ( mhaconfig_t &)`
- `void release ()`
- `void process ()`
- `mha_spec_t * process ( mha_spec_t *)`
- `mha_wave_t * process ( mha_wave_t *)`

### Private Attributes

- `std::string algo`
- `MHAParser::mhapluginloader_t plug`
- `MHAParser::string_t input`
- `MHAParser::bool_t permute`
- `MHA_AC::waveform_t * s_out`
- `MHASignal::waveform_t * s_in_perm`
- `bool b_permute`
- `mha_wave_t s_in`

## Additional Inherited Members

### 5.11.1 Constructor & Destructor Documentation

```
5.11.1.1 interface_t() ac_proc::interface_t::interface_t (
    algo_comm_t iac,
    const std::string & configured_name )
```

Default values are set and MHA configuration variables registered into the parser.

#### Parameters

<i>iac</i>	algorithm communication handle
<i>configured_name</i>	algorithm name

### 5.11.2 Member Function Documentation

```
5.11.2.1 prepare() void ac_proc::interface_t::prepare (
    mhaconfig_t & tf ) [virtual]
```

Implements **MHAPlugIn::plugin\_t< int >** (p. [1149](#)).

```
5.11.2.2 release() void ac_proc::interface_t::release ( ) [virtual]
```

Reimplemented from **MHAPlugIn::plugin\_t< int >** (p. [1150](#)).

```
5.11.2.3 process() [1/3] void ac_proc::interface_t::process ( )
```

**5.11.2.4 process()** [2/3] `mha_spec_t * ac_proc::interface_t::process ( mha_spec_t * s )`

**5.11.2.5 process()** [3/3] `mha_wave_t * ac_proc::interface_t::process ( mha_wave_t * s )`

### 5.11.3 Member Data Documentation

**5.11.3.1 algo** `std::string ac_proc::interface_t::algo` [private]

**5.11.3.2 plug** `MHAParser::mhapluginloader_t ac_proc::interface_t::plug` [private]

**5.11.3.3 input** `MHAParser::string_t ac_proc::interface_t::input` [private]

**5.11.3.4 permute** `MHAParser::bool_t ac_proc::interface_t::permute` [private]

**5.11.3.5 s\_out** `MHA_AC::waveform_t* ac_proc::interface_t::s_out` [private]

**5.11.3.6 s\_in\_perm** `MHASignal::waveform_t* ac_proc::interface_t::s_in_perm` [private]

### 5.11.3.7 b\_permute bool ac\_proc::interface\_t::b\_permute [private]

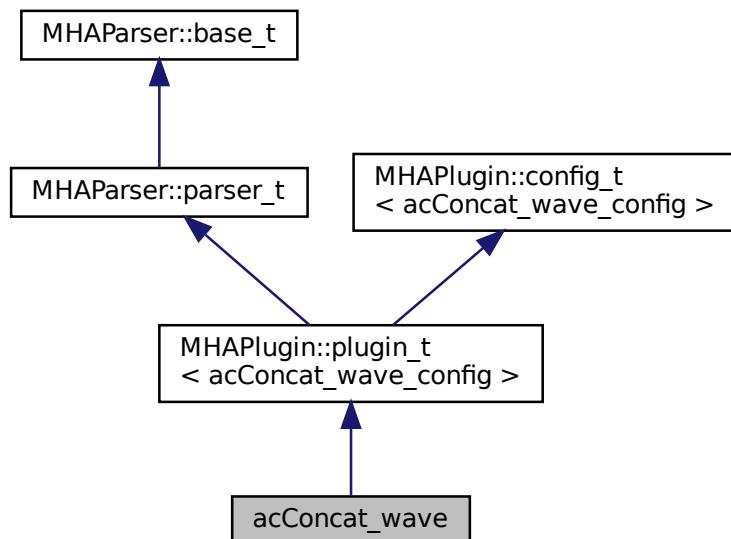
### 5.11.3.8 s\_in mha\_wave\_t ac\_proc::interface\_t::s\_in [private]

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

- `ac_proc.cpp`

## 5.12 acConcat\_wave Class Reference

Inheritance diagram for acConcat\_wave:



## Public Member Functions

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

**Public Attributes**

- `MHAParser::int_t num_AC`
- `MHAParser::string_t prefix_names_AC`
- `MHAParser::vint_t samples_AC`
- `MHAParser::string_t name_con_AC`
- `MHAParser::int_t numchannels`

**Private Member Functions**

- `void update_cfg ()`

**Private Attributes**

- `MHAEVENTS::patchbay_t< acConcat_wave > patchbay`

**Additional Inherited Members****5.12.1 Constructor & Destructor Documentation**

**5.12.1.1 acConcat\_wave()** `acConcat_wave::acConcat_wave ( algo_comm_t iac, const std::string & configured_name )`

Constructs our plugin.

**5.12.1.2 ~acConcat\_wave()** `acConcat_wave::~acConcat_wave ( )`

**5.12.2 Member Function Documentation**

**5.12.2.1 process()** `mha_wave_t * acConcat_wave::process ( mha_wave_t * signal )`

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

**5.12.2.2 prepare()** `void acConcat_wave::prepare ( mhaconfig_t & ) [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. [1149](#)).

**5.12.2.3 release()** void acConcat\_wave::release (  
void ) [inline], [virtual]

Reimplemented from **MHAPlugIn::plugin\_t< acConcat\_wave\_config >** (p. [1150](#)).

**5.12.2.4 update\_cfg()** void acConcat\_wave::update\_cfg ( ) [private]

### 5.12.3 Member Data Documentation

**5.12.3.1 num\_AC** MHAParser::int\_t acConcat\_wave::num\_AC

**5.12.3.2 prefix\_names\_AC** MHAParser::string\_t acConcat\_wave::prefix\_names\_AC

**5.12.3.3 samples\_AC** MHAParser::vint\_t acConcat\_wave::samples\_AC

**5.12.3.4 name\_con\_AC** MHAParser::string\_t acConcat\_wave::name\_con\_AC

**5.12.3.5 numchannels** `MHAParser::int_t acConcat_wave::numchannels`**5.12.3.6 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`

**5.13 acConcat\_wave\_config Class Reference****Public Member Functions**

- `acConcat_wave_config ( algo_comm_t & ac, acConcat_wave *_concat)`
- `~acConcat_wave_config ()`
- `mha_wave_t * process ( mha_wave_t *)`

**Public Attributes**

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

**5.13.1 Constructor & Destructor Documentation****5.13.1.1 acConcat\_wave\_config()** `acConcat_wave_config::acConcat_wave_config (`  
`algo_comm_t & ac,`  
`acConcat_wave * _concat )`**5.13.1.2 ~acConcat\_wave\_config()** `acConcat_wave_config::~acConcat_wave_config ( )`

### 5.13.2 Member Function Documentation

**5.13.2.1 process()** `mha_wave_t * acConcat_wave_config::process (`  
`mha_wave_t * wave )`

### 5.13.3 Member Data Documentation

**5.13.3.1 ac algo\_comm\_t& acConcat\_wave\_config::ac**

**5.13.3.2 strNames\_AC** `std::vector<std::string> acConcat_wave_config::strNames_AC`

**5.13.3.3 numSamples\_AC** `std::vector<int> acConcat_wave_config::numSamples_AC`

**5.13.3.4 vGCC mha\_wave\_t acConcat\_wave\_config::vGCC**

**5.13.3.5 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**

## 5.14 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*

### 5.14.1 Detailed Description

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

### 5.14.2 Constructor & Destructor Documentation

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

Converts AC variable to parser monitor.

**Parameters**

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

**5.14.3 Member Function Documentation****5.14.3.1 getvar()** void acmon::ac\_monitor\_t::getvar ( algo\_comm\_t ac )

Update values of monitor.

**Parameters**

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

**5.14.4 Member Data Documentation****5.14.4.1 name** std::string acmon::ac\_monitor\_t::name

name of AC variable and parser monitor

**5.14.4.2 dimstr** std::string acmon::ac\_monitor\_t::dimstr

columns x rows

**5.14.4.3 mon** `MHAParser::vfloat_mon_t` `acmon::ac_monitor_t::mon`

Monitor used for real vectors.

**5.14.4.4 mon\_mat** `MHAParser::mfloat_mon_t` `acmon::ac_monitor_t::mon_mat`

Monitor used for real matrices.

**5.14.4.5 mon\_complex** `MHAParser::vcomplex_mon_t` `acmon::ac_monitor_t::mon_complex`

monitor used for complex vectors

**5.14.4.6 mon\_mat\_complex** `MHAParser::mcomplex_mon_t` `acmon::ac_monitor_t::mon_←mat_complex`

monitor used for complex matrices

**5.14.4.7 p\_parser** `MHAParser::parser_t&` `acmon::ac_monitor_t::p_parser`

parent parser to insert monitor into

**5.14.4.8 use\_mat** `bool acmon::ac_monitor_t::use_mat [private]`

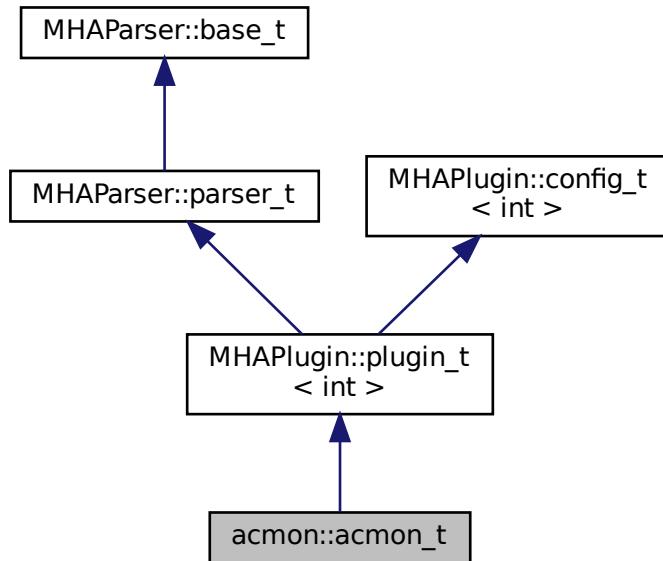
if true, use matrix monitor, else use vector monitor

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

- `ac_monitor_type.hh`
- `ac_monitor_type.cpp`

## 5.15 acmon::acmon\_t Class Reference

Inheritance diagram for acmon::acmon\_t:



### Public Member Functions

- `acmon_t ( algo_comm_t, const std::string &configured_name)`
- `~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 algo`
- `bool b_cont`
- `bool b_snapshot`

**Additional Inherited Members****5.15.1 Constructor & Destructor Documentation**

**5.15.1.1 `acmon_t()`** `acmon::acmon_t::acmon_t (`  
`algo_comm_t iac,`  
`const std::string & configured_name )`

**5.15.1.2 `~acmon_t()`** `acmon::acmon_t::~acmon_t ( )`

**5.15.2 Member Function Documentation**

**5.15.2.1 `prepare()`** `void acmon::acmon_t::prepare (`  
`mhaconfig_t & ) [virtual]`

Implements **MHAPlugin::plugin\_t< int >** (p. 1149).

**5.15.2.2 release()** void acmon::acmon\_t::release () [inline], [virtual]

Reimplemented from **MHAParser::plugin\_t< int >** (p. 1150).

**5.15.2.3 process() [1/2]** mha\_spec\_t \* acmon::acmon\_t::process (mha\_spec\_t \* s )

**5.15.2.4 process() [2/2]** mha\_wave\_t \* acmon::acmon\_t::process (mha\_wave\_t \* s )

**5.15.2.5 save\_vars()** void acmon::acmon\_t::save\_vars () [private]

**5.15.2.6 update\_recmode()** void acmon::acmon\_t::update\_recmode () [private]

### 5.15.3 Member Data Documentation

**5.15.3.1 ac algo\_comm\_t** acmon::acmon\_t::ac [private]

**5.15.3.2 varlist MHAParser::vstring\_mon\_t** acmon::acmon\_t::varlist [private]

**5.15.3.3 dimensions MHAParser::vstring\_mon\_t** acmon::acmon\_t::dimensions [private]

**5.15.3.4 dispmode** `MHAParser::kw_t acmon::acmon_t::dispmode [private]`

**5.15.3.5 recmode** `MHAParser::kw_t acmon::acmon_t::recmode [private]`

**5.15.3.6 vars** `std::vector< ac_monitor_t*> acmon::acmon_t::vars [private]`

**5.15.3.7 patchbay** `MHAEvents::patchbay_t< acmon_t> acmon::acmon_t::patchbay [private]`

**5.15.3.8 algo** `std::string acmon::acmon_t::algo [private]`

**5.15.3.9 b\_cont** `bool acmon::acmon_t::b_cont [private]`

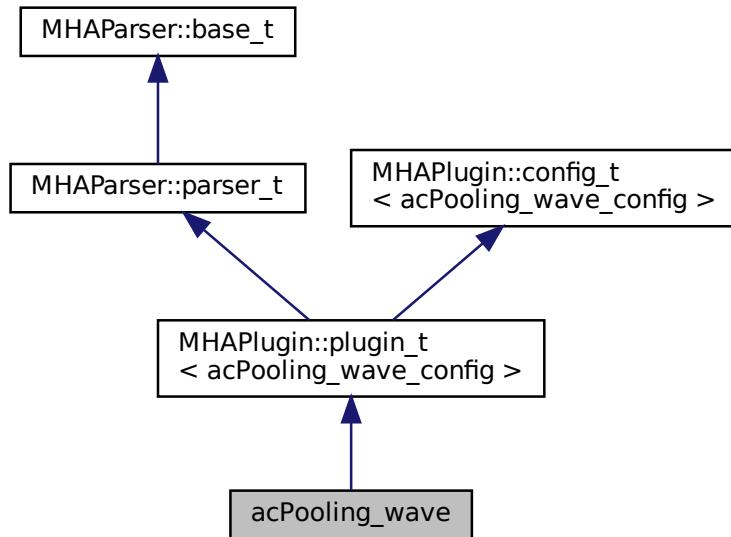
**5.15.3.10 b\_snapshot** `bool acmon::acmon_t::b_snapshot [private]`

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

- `acmon.cpp`

## 5.16 acPooling\_wave Class Reference

Inheritance diagram for acPooling\_wave:



### Public Member Functions

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

### Public Attributes

- `MHAParser::int_t numsamples`
- `MHAParser::int_t pooling_wndlen`
- `MHAParser::kw_t pooling_type`
- `MHAParser::float_t upper_threshold`
- `MHAParser::float_t lower_threshold`
- `MHAParser::int_t neighbourhood`

- `MHAParser::float_t alpha`
- `MHAParser::string_t p_name`
- `MHAParser::string_t p_biased_name`
- `MHAParser::string_t pool_name`
- `MHAParser::string_t max_pool_ind_name`
- `MHAParser::string_t like_ratio_name`
- `MHAParser::vfloat_t prob_bias`

### Private Member Functions

- `void update_cfg ()`

### Private Attributes

- `MHAEvents::patchbay_t< acPooling_wave > patchbay`

### Additional Inherited Members

#### 5.16.1 Constructor & Destructor Documentation

**5.16.1.1 acPooling\_wave()** `acPooling_wave::acPooling_wave ( algo_comm_t iac, const std::string & configured_name )`

Constructs our plugin.

**5.16.1.2 ~acPooling\_wave()** `acPooling_wave::~acPooling_wave ( )`

#### 5.16.2 Member Function Documentation

**5.16.2.1 process()** `mha_wave_t * acPooling_wave::process ( mha_wave_t * signal )`

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

**5.16.2.2 prepare()** `void acPooling_wave::prepare ( mhaconfig_t & signal_info ) [virtual]`

Plugin preparation.

An opportunity to validate configuration parameters before instantiating a configuration.

**Parameters**

<i>signal_info</i>	Structure containing a description of the form of the signal (domain, number of channels, frames per block, sampling rate).
--------------------	---

Implements **MHAParser::plugin\_t< acPooling\_wave\_config >** (p. 1149).

**5.16.2.3 release()** void acPooling\_wave::release (  
void ) [inline], [virtual]

Reimplemented from **MHAParser::plugin\_t< acPooling\_wave\_config >** (p. 1150).

**5.16.2.4 update\_cfg()** void acPooling\_wave::update\_cfg ( ) [private]

### 5.16.3 Member Data Documentation

**5.16.3.1 numsamples** MHAParser::int\_t acPooling\_wave::numsamples

**5.16.3.2 pooling\_wndlen** MHAParser::int\_t acPooling\_wave::pooling\_wndlen

**5.16.3.3 pooling\_type** MHAParser::kw\_t acPooling\_wave::pooling\_type

**5.16.3.4 upper\_threshold** MHAParser::float\_t acPooling\_wave::upper\_threshold

**5.16.3.5 lower\_threshold** `MHAParser::float_t acPooling_wave::lower_threshold`

**5.16.3.6 neighbourhood** `MHAParser::int_t acPooling_wave::neighbourhood`

**5.16.3.7 alpha** `MHAParser::float_t acPooling_wave::alpha`

**5.16.3.8 p\_name** `MHAParser::string_t acPooling_wave::p_name`

**5.16.3.9 p\_biased\_name** `MHAParser::string_t acPooling_wave::p_biased_name`

**5.16.3.10 pool\_name** `MHAParser::string_t acPooling_wave::pool_name`

**5.16.3.11 max\_pool\_ind\_name** `MHAParser::string_t acPooling_wave::max_pool_ind_→  
name`

**5.16.3.12 like\_ratio\_name** `MHAParser::string_t acPooling_wave::like_ratio_name`

**5.16.3.13 prob\_bias** `MHAParser::vfloat_t acPooling_wave::prob_bias`

**5.16.3.14 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`

## 5.17 acPooling\_wave\_config Class Reference

### Public Member Functions

- `acPooling_wave_config ( algo_comm_t & ac, const mhaconfig_t in_cfg, ac< Pooling_wave * _pooling)`
- `~acPooling_wave_config ()`
- `mha_wave_t * process ( mha_wave_t *)`
- `void insert ()`

### Public Attributes

- `algo_comm_t & ac`
- `std::string raw_p_name`
- `MHA_AC::waveform_t p`
- `MHA_AC::waveform_t p_biased`
- `MHA_AC::waveform_t p_max`
- `MHA_AC::waveform_t like_ratio`
- `mha_wave_t c`
- `unsigned int pooling_ind`
- `unsigned int pooling_option`
- `unsigned int pooling_size`
- `float up_thresh`
- `float low_thresh`
- `int neigh`
- `float alpha`
- `MHASignal::waveform_t pool`
- `MHASignal::waveform_t prob_bias_func`

### 5.17.1 Constructor & Destructor Documentation

**5.17.1.1 acPooling\_wave\_config()** `acPooling_wave_config::acPooling_wave_config ( algo_comm_t & ac, const mhaconfig_t in_cfg, acPooling_wave * _pooling )`

**5.17.1.2 ~acPooling\_wave\_config()** `acPooling_wave_config::~acPooling_wave_config ( )`

## 5.17.2 Member Function Documentation

**5.17.2.1 process()** `mha_wave_t * acPooling_wave_config::process ( mha_wave_t * wave )`

**5.17.2.2 insert()** `void acPooling_wave_config::insert ( )`

## 5.17.3 Member Data Documentation

**5.17.3.1 ac** `algo_comm_t& acPooling_wave_config::ac`

**5.17.3.2 raw\_p\_name** `std::string acPooling_wave_config::raw_p_name`

**5.17.3.3 p** `MHA_AC::waveform_t acPooling_wave_config::p`

**5.17.3.4 p\_biased** `MHA_AC::waveform_t` acPooling\_wave\_config::p\_biased

**5.17.3.5 p\_max** `MHA_AC::waveform_t` acPooling\_wave\_config::p\_max

**5.17.3.6 like\_ratio** `MHA_AC::waveform_t` acPooling\_wave\_config::like\_ratio

**5.17.3.7 c** `mha_wave_t` acPooling\_wave\_config::c

**5.17.3.8 pooling\_ind** `unsigned int` acPooling\_wave\_config::pooling\_ind

**5.17.3.9 pooling\_option** `unsigned int` acPooling\_wave\_config::pooling\_option

**5.17.3.10 pooling\_size** `unsigned int` acPooling\_wave\_config::pooling\_size

**5.17.3.11 up\_thresh** `float` acPooling\_wave\_config::up\_thresh

**5.17.3.12 low\_thresh** `float` acPooling\_wave\_config::low\_thresh

**5.17.3.13 `neigh`** `int acPooling_wave_config::neigh`

**5.17.3.14 `alpha`** `float acPooling_wave_config::alpha`

**5.17.3.15 `pool`** `MHASignal::waveform_t acPooling_wave_config::pool`

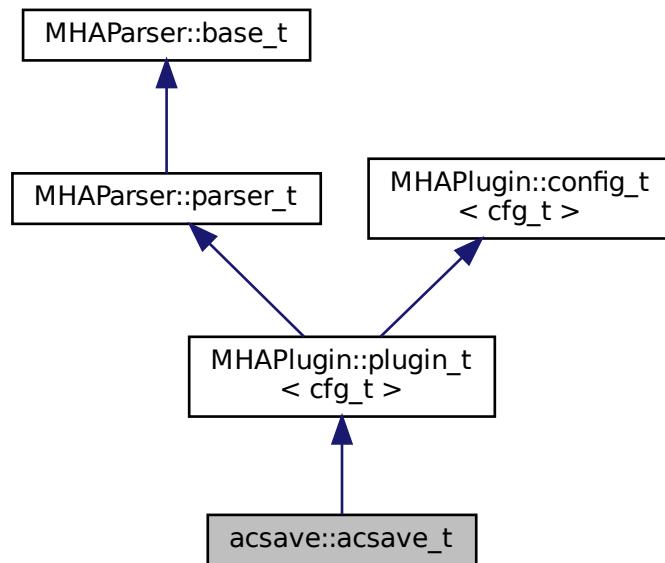
**5.17.3.16 `prob_bias_func`** `MHASignal::waveform_t acPooling_wave_config::prob_bias_func`

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

- `acPooling_wave.h`
- `acPooling_wave.cpp`

## 5.18 `acsave::acsave_t` Class Reference

Inheritance diagram for `acsave::acsave_t`:



## Public Member Functions

- `acsave_t ( algo_comm_t iac, const std::string &configured_name)`
- `void prepare ( mhaconfig_t &)`
- `void release ()`
- `mha_spec_t * process ( mha_spec_t *)`
- `mha_wave_t * process ( mha_wave_t *)`
- `void event_start_recording ()`
- `void event_stop_and_flush ()`

## Private Types

- `typedef std::vector< save_var_t * > varlist_t`

## Private Member Functions

- `void process ()`

## Private Attributes

- `MHAParser::bool_t bflush`
- `MHAParser::kw_t fileformat`
- `MHAParser::string_t fname`
- `MHAParser::float_t reclen`
- `MHAParser::vstring_t variables`
- `varlist_t varlist`
- `std::string algo`
- `bool b_prepared`
- `bool b_flushed`
- `MHAEvents::patchbay_t< acsave_t > patchbay`

## Additional Inherited Members

### 5.18.1 Member Typedef Documentation

**5.18.1.1 varlist\_t** `typedef std::vector< save_var_t *> acsave::acsave_t::varlist_t [private]`

## 5.18.2 Constructor & Destructor Documentation

**5.18.2.1 `acsave_t()`** `acsave::acsave_t::acsave_t ( algo_comm_t iac, const std::string & configured_name )`

## 5.18.3 Member Function Documentation

**5.18.3.1 `prepare()`** `void acsave::acsave_t::prepare ( mhaconfig_t & tf ) [virtual]`

Implements **MHAPlugin::plugin\_t< cfg\_t >** (p. [1149](#)).

**5.18.3.2 `release()`** `void acsave::acsave_t::release ( ) [virtual]`

Reimplemented from **MHAPlugin::plugin\_t< cfg\_t >** (p. [1150](#)).

**5.18.3.3 `process() [1/3]`** `mha_spec_t * acsave::acsave_t::process ( mha_spec_t * s )`

**5.18.3.4 `process() [2/3]`** `mha_wave_t * acsave::acsave_t::process ( mha_wave_t * s )`

**5.18.3.5 `event_start_recording()`** `void acsave::acsave_t::event_start_recording ( )`

**5.18.3.6 event\_stop\_and\_flush()** void acsave::acsave\_t::event\_stop\_and\_flush ( )

**5.18.3.7 process() [3/3]** void acsave::acsave\_t::process ( ) [private]

#### 5.18.4 Member Data Documentation

**5.18.4.1 bflush** MHAParser::bool\_t acsave::acsave\_t::bflush [private]

**5.18.4.2 fileformat** MHAParser::kw\_t acsave::acsave\_t::fileformat [private]

**5.18.4.3 fname** MHAParser::string\_t acsave::acsave\_t::fname [private]

**5.18.4.4 reclen** MHAParser::float\_t acsave::acsave\_t::reclen [private]

**5.18.4.5 variables** MHAParser::vstring\_t acsave::acsave\_t::variables [private]

**5.18.4.6 varlist** varlist\_t acsave::acsave\_t::varlist [private]

**5.18.4.7 algo** std::string acsave::acsave\_t::algo [private]

**5.18.4.8 `b_prepared`** `bool acsave::acsave_t::b_prepared [private]`

**5.18.4.9 `b_flushed`** `bool acsave::acsave_t::b_flushed [private]`

**5.18.4.10 `patchbay`** `MHAEEvents::patchbay_t< acsave_t> acsave::acsave_t::patchbay [private]`

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

- `acsave.cpp`

## 5.19 `acsave::cfg_t` Class Reference

### Public Member Functions

- `cfg_t` (`const algo_comm_t &ac, unsigned int imax_frames, std::vector< std::string > &var_names`)
- `~cfg_t ()`
- `void store_frame ()`
- `void flush_data (const std::string &, unsigned int)`

### Private Attributes

- `algo_comm_t ac`
- `unsigned int nvars`
- `save_var_t ** varlist`
- `unsigned int rec_frames`
- `unsigned int max_frames`

### 5.19.1 Constructor & Destructor Documentation

```
5.19.1.1 cfg_t() cfg_t::cfg_t (
    const algo_comm_t & iac,
    unsigned int imax_frames,
    std::vector< std::string > & var_names )
```

```
5.19.1.2 ~cfg_t() cfg_t::~cfg_t ( )
```

## 5.19.2 Member Function Documentation

```
5.19.2.1 store_frame() void cfg_t::store_frame ( )
```

This function is called in the processing thread.

```
5.19.2.2 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

## 5.19.3 Member Data Documentation

```
5.19.3.1 ac algo_comm_t acsave::cfg_t::ac [private]
```

```
5.19.3.2 nvars unsigned int acsave::cfg_t::nvars [private]
```

**5.19.3.3 varlist** `save_var_t** acsave::cfg_t::varlist` [private]

**5.19.3.4 rec\_frames** `unsigned int acsave::cfg_t::rec_frames` [private]

**5.19.3.5 max\_frames** `unsigned int acsave::cfg_t::max_frames` [private]

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

- `acsave.cpp`

## 5.20 `acsave::mat4head_t` Struct Reference

### Public Attributes

- `int32_t t`
- `int32_t rows`
- `int32_t cols`
- `int32_t imag`
- `int32_t namelen`

### 5.20.1 Member Data Documentation

**5.20.1.1 t** `int32_t acsave::mat4head_t::t`

**5.20.1.2 rows** `int32_t acsave::mat4head_t::rows`

**5.20.1.3 cols** int32\_t acsave::mat4head\_t::cols

**5.20.1.4 imag** int32\_t acsave::mat4head\_t::imag

**5.20.1.5 namelen** int32\_t acsave::mat4head\_t::namelen

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

- **acsave.cpp**

## 5.21 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**

## 5.21.1 Constructor & Destructor Documentation

**5.21.1.1 `save_var_t()`** acsave::save\_var\_t::save\_var\_t (

```
const std::string & nm,
int n,
const algo_comm_t & iac )
```

**5.21.1.2 `~save_var_t()`** acsave::save\_var\_t::~save\_var\_t ( )

## 5.21.2 Member Function Documentation

**5.21.2.1 `store_frame()`** void acsave::save\_var\_t::store\_frame ( )

**5.21.2.2 `save_txt()`** void acsave::save\_var\_t::save\_txt (

```
FILE * fh,
unsigned int writeframes )
```

**5.21.2.3 `save_mat4()`** void acsave::save\_var\_t::save\_mat4 (

```
FILE * fh,
unsigned int writeframes )
```

**5.21.2.4 `save_m()`** void acsave::save\_var\_t::save\_m (

```
FILE * fh,
unsigned int writeframes )
```

### 5.21.3 Member Data Documentation

**5.21.3.1 data** double\* acsave::save\_var\_t::data

**5.21.3.2 name** std::string acsave::save\_var\_t::name [private]

**5.21.3.3 nframes** unsigned int acsave::save\_var\_t::nframes [private]

**5.21.3.4 ndim** unsigned int acsave::save\_var\_t::ndim [private]

**5.21.3.5 maxframe** unsigned int acsave::save\_var\_t::maxframe [private]

**5.21.3.6 ac** algo\_comm\_t acsave::save\_var\_t::ac [private]

**5.21.3.7 framecnt** unsigned int acsave::save\_var\_t::framecnt [private]

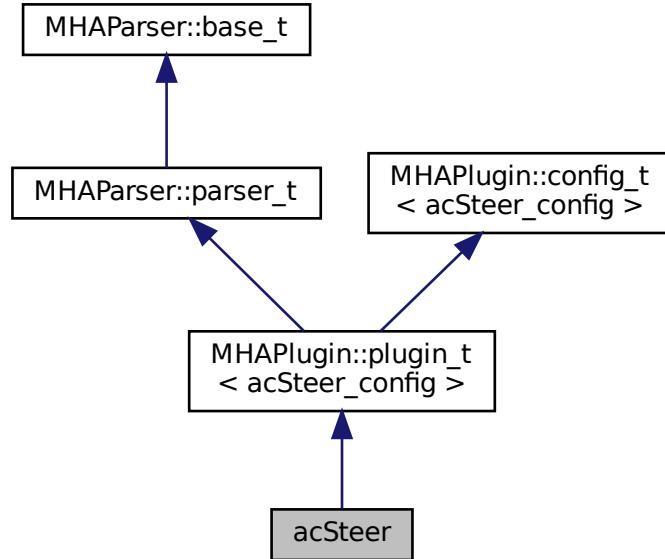
**5.21.3.8 b\_complex** bool acsave::save\_var\_t::b\_complex [private]

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

- **acsave.cpp**

## 5.22 acSteer Class Reference

Inheritance diagram for acSteer:



### Public Member Functions

- **acSteer ( algo\_comm\_t iac, const std::string &configured\_name)**  
*Constructs our plugin.*
- **~acSteer ()**
- **mha\_spec\_t \* process ( mha\_spec\_t \*)**  
*This method is a NOOP.*
- **void prepare ( mhaconfig\_t &)**  
*Plugin preparation.*
- **void release (void)**

### Public Attributes

- `MHAParser::string_t steerFile`
- `MHAParser::string_t acSteerName1`
- `MHAParser::string_t acSteerName2`
- `MHAParser::int_t nsteerchan`
- `MHAParser::int_t nrefmic`

### Private Member Functions

- void **update\_cfg ()**

### Private Attributes

- **MHAEvents::patchbay\_t< acSteer > patchbay**

### Additional Inherited Members

#### 5.22.1 Constructor & Destructor Documentation

**5.22.1.1 acSteer()** acSteer::acSteer (

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

Constructs our plugin.

**5.22.1.2 ~acSteer()** acSteer::~acSteer ( )

#### 5.22.2 Member Function Documentation

**5.22.2.1 process()** mha\_spec\_t \* acSteer::process (

```
    mha_spec_t * signal )
```

This method is a NOOP.

**5.22.2.2 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 **MHAPlugIn::plugin\_t< acSteer\_config >** (p. 1149).

**5.22.2.3 release()** `void acSteer::release ( void ) [inline], [virtual]`

Reimplemented from **MHAPlugIn::plugin\_t< acSteer\_config >** (p. 1150).

**5.22.2.4 update\_cfg()** `void acSteer::update_cfg ( ) [private]`

## 5.22.3 Member Data Documentation

**5.22.3.1 steerFile** `MHAParser::string_t acSteer::steerFile`

**5.22.3.2 acSteerName1** `MHAParser::string_t acSteer::acSteerName1`

**5.22.3.3 acSteerName2** `MHAParser::string_t acSteer::acSteerName2`

**5.22.3.4 nsteerchan** `MHAParser::int_t acSteer::nsteerchan`

**5.22.3.5 nrefmic** `MHAParser::int_t acSteer::nrefmic`**5.22.3.6 patchbay** `MHAEvents::patchbay_t< acSteer> acSteer::patchbay [private]`

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

- `acSteer.h`
- `acSteer.cpp`

## 5.23 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`

### 5.23.1 Constructor & Destructor Documentation

**5.23.1.1 acSteer\_config()** `acSteer_config::acSteer_config (`  
`algo_comm_t & ac,`  
`const mhaconfig_t in_cfg,`  
`acSteer * acSteer )`

**5.23.1.2 ~acSteer\_config()** acSteer\_config::~acSteer\_config ( )

## 5.23.2 Member Function Documentation

**5.23.2.1 insert()** void acSteer\_config::insert ( )

## 5.23.3 Member Data Documentation

**5.23.3.1 nchan** unsigned int acSteer\_config::nchan

**5.23.3.2 nfreq** unsigned int acSteer\_config::nfreq

**5.23.3.3 nsteerchan** unsigned int acSteer\_config::nsteerchan

**5.23.3.4 nrefmic** unsigned int acSteer\_config::nrefmic

**5.23.3.5 nangle** unsigned int acSteer\_config::nangle

**5.23.3.6 specSteer1** MHA\_AC::spectrum\_t acSteer\_config::specSteer1

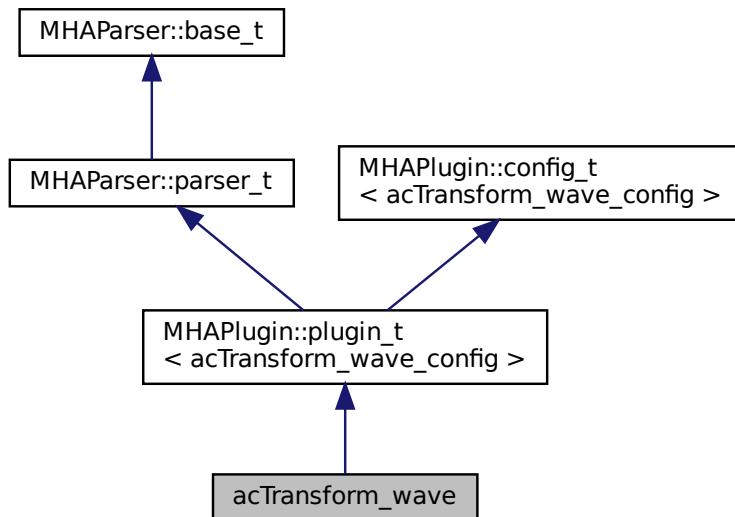
### 5.23.3.7 specSteer2 `MHA_AC::spectrum_t acSteer_config::specSteer2`

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

- `acSteer.h`
- `acSteer.cpp`

## 5.24 acTransform\_wave Class Reference

Inheritance diagram for acTransform\_wave:



### Public Member Functions

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

## Public Attributes

- `MHAParser::string_t ang_name`
- `MHAParser::string_t raw_p_name`
- `MHAParser::string_t raw_p_max_name`
- `MHAParser::string_t rotated_p_name`
- `MHAParser::string_t rotated_p_max_name`
- `MHAParser::int_t numsamples`
- `MHAParser::bool_t to_from`

## Private Member Functions

- `void update_cfg ()`

## Private Attributes

- `MHAEvents::patchbay_t< acTransform_wave > patchbay`

## Additional Inherited Members

### 5.24.1 Constructor & Destructor Documentation

**5.24.1.1 `acTransform_wave()`** `acTransform_wave::acTransform_wave (`  
`algo_comm_t iac,`  
`const std::string & configured_name )`

Constructs our plugin.

**5.24.1.2 `~acTransform_wave()`** `acTransform_wave::~acTransform_wave ( )`

### 5.24.2 Member Function Documentation

```
5.24.2.1 process() mha_wave_t * acTransform_wave::process (
    mha_wave_t * signal )
```

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

```
5.24.2.2 prepare() void acTransform_wave::prepare (
    mhaconfig_t & signal_info ) [virtual]
```

Plugin preparation.

An opportunity to validate configuration parameters before instantiating a configuration.

## Parameters

<i>signal_info</i>	Structure containing a description of the form of the signal (domain, number of channels, frames per block, sampling rate).
--------------------	---

Implements **MHAPlugIn::plugin\_t< acTransform\_wave\_config >** (p. 1149).

**5.24.2.3 release()** void acTransform\_wave::release ( void ) [inline], [virtual]

Reimplemented from **MHAPlugIn::plugin\_t< acTransform\_wave\_config >** (p. 1150).

**5.24.2.4 update\_cfg()** void acTransform\_wave::update\_cfg ( ) [private]

## 5.24.3 Member Data Documentation

**5.24.3.1 ang\_name** MHAParser::string\_t acTransform\_wave::ang\_name

**5.24.3.2 raw\_p\_name** MHAParser::string\_t acTransform\_wave::raw\_p\_name

**5.24.3.3 raw\_p\_max\_name** MHAParser::string\_t acTransform\_wave::raw\_p\_max\_name

**5.24.3.4 rotated\_p\_name** MHAParser::string\_t acTransform\_wave::rotated\_p\_name

**5.24.3.5 rotated\_p\_max\_name** `MHAParser::string_t acTransform_wave::rotated_p_<→ max_name`

**5.24.3.6 numsamples** `MHAParser::int_t acTransform_wave::numsamples`

**5.24.3.7 to\_from** `MHAParser::bool_t acTransform_wave::to_from`

**5.24.3.8 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`

## 5.25 acTransform\_wave\_config Class Reference

### Public Member Functions

- `acTransform_wave_config ( algo_comm_t & ac, 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`

### 5.25.1 Constructor & Destructor Documentation

**5.25.1.1 acTransform\_wave\_config()** acTransform\_wave\_config::acTransform\_wave\_config ( algo\_comm\_t & ac, acTransform\_wave \* \_transform )

**5.25.1.2 ~acTransform\_wave\_config()** acTransform\_wave\_config::~acTransform\_wave\_config ( )

### 5.25.2 Member Function Documentation

**5.25.2.1 process()** mha\_wave\_t \* acTransform\_wave\_config::process ( mha\_wave\_t \* wave )

### 5.25.3 Member Data Documentation

**5.25.3.1 ac** algo\_comm\_t& acTransform\_wave\_config::ac

**5.25.3.2 ang\_name** std::string acTransform\_wave\_config::ang\_name

**5.25.3.3 raw\_p\_name** std::string acTransform\_wave\_config::raw\_p\_name

**5.25.3.4 raw\_p\_max\_name** std::string acTransform\_wave\_config::raw\_p\_max\_name

**5.25.3.5 rotated\_p** MHA\_AC::waveform\_t acTransform\_wave\_config::rotated\_p

**5.25.3.6 rotated\_i** MHA\_AC::int\_t acTransform\_wave\_config::rotated\_i

**5.25.3.7 offset** unsigned int acTransform\_wave\_config::offset

**5.25.3.8 resolution** unsigned int acTransform\_wave\_config::resolution

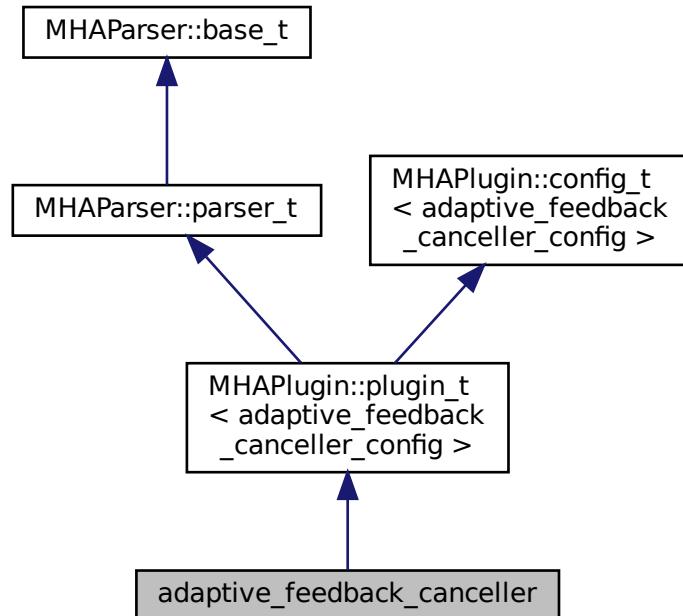
**5.25.3.9 to\_from** unsigned int acTransform\_wave\_config::to\_from

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

- **acTransform\_wave.h**
- **acTransform\_wave.cpp**

## 5.26 adaptive\_feedback\_canceller Class Reference

Inheritance diagram for adaptive\_feedback\_canceller:



### Public Member Functions

- **`adaptive_feedback_canceller ( algo_comm_t ac, const std::string &configured_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`**

- MHParse::vfloat\_t gains
- MHParse::string\_t name\_e
- MHParse::string\_t name\_f
- MHParse::string\_t name\_lpc
- MHParse::int\_t lpc\_order
- MHParse::vint\_t afc\_delay
- MHParse::vint\_t delay\_w
- MHParse::vint\_t delay\_d
- MHParse::int\_t n\_no\_update

### Private Member Functions

- void update\_cfg ()

### Private Attributes

- MHAEvents::patchbay\_t< adaptive\_feedback\_canceller > patchbay

### Additional Inherited Members

#### 5.26.1 Constructor & Destructor Documentation

```
5.26.1.1 adaptive_feedback_canceller() adaptive_feedback_canceller::adaptive_<→
feedback_canceller (
    algo_comm_t ac,
    const std::string & configured_name )
```

Constructs our plugin.

```
5.26.1.2 ~adaptive_feedback_canceller() adaptive_feedback_canceller::~adaptive_<→
feedback_canceller ( )
```

#### 5.26.2 Member Function Documentation

---

**5.26.2.1 process()** `mha_wave_t * adaptive_feedback_canceller::process (`  
`mha_wave_t * signal )`

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

**5.26.2.2 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. [1149](#)).

**5.26.2.3 release()** `void adaptive_feedback_canceller::release (`  
`void ) [inline], [virtual]`

Reimplemented from **MHAPlugIn::plugin\_t< adaptive\_feedback\_canceller\_config >** (p. [1150](#)).

**5.26.2.4 update\_cfg()** `void adaptive_feedback_canceller::update_cfg ( ) [private]`

### 5.26.3 Member Data Documentation

**5.26.3.1 rho** `MHAParser::float_t adaptive_feedback_canceller::rho`

**5.26.3.2 c** `MHAParser::float_t` `adaptive_feedback_canceller::c`

**5.26.3.3 ntaps** `MHAParser::int_t` `adaptive_feedback_canceller::ntaps`

**5.26.3.4 gains** `MHAParser::vfloat_t` `adaptive_feedback_canceller::gains`

**5.26.3.5 name\_e** `MHAParser::string_t` `adaptive_feedback_canceller::name_e`

**5.26.3.6 name\_f** `MHAParser::string_t` `adaptive_feedback_canceller::name_f`

**5.26.3.7 name\_lpc** `MHAParser::string_t` `adaptive_feedback_canceller::name_lpc`

**5.26.3.8 lpc\_order** `MHAParser::int_t` `adaptive_feedback_canceller::lpc_order`

**5.26.3.9 afc\_delay** `MHAParser::vint_t` `adaptive_feedback_canceller::afc_delay`

**5.26.3.10 delay\_w** `MHAParser::vint_t` `adaptive_feedback_canceller::delay_w`

---

**5.26.3.11 `delay_d`** `MHAParser::vint_t adaptive_feedback_canceller::delay_d`

**5.26.3.12 `n_no_update`** `MHAParser::int_t adaptive_feedback_canceller::n_no_update`

**5.26.3.13 `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`

## 5.27 `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`

### 5.27.1 Constructor & Destructor Documentation

**5.27.1.1 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 )`

**5.27.1.2 ~adaptive\_feedback\_canceller\_config()** `adaptive_feedback_canceller_config::~adaptive_feedback_canceller_config ( )`

### 5.27.2 Member Function Documentation

**5.27.2.1 process()** `mha_wave_t * adaptive_feedback_canceller_config::process (`  
`mha_wave_t * s_Y,`  
`mha_real_t rho,`  
`mha_real_t c )`

**5.27.2.2 insert()** `void adaptive_feedback_canceller_config::insert ( )`

### 5.27.3 Member Data Documentation

**5.27.3.1 ac** `algo_comm_t adaptive_feedback_canceller_config::ac [private]`

**5.27.3.2 ntaps** `unsigned int adaptive_feedback_canceller_config::ntaps [private]`

**5.27.3.3 frames** `unsigned int adaptive_feedback_canceller_config::frames [private]`

**5.27.3.4 channels** `unsigned int adaptive_feedback_canceller_config::channels [private]`

**5.27.3.5 s\_E** `MHA_AC::waveform_t adaptive_feedback_canceller_config::s_E [private]`

**5.27.3.6 F** `MHA_AC::waveform_t adaptive_feedback_canceller_config::F [private]`

**5.27.3.7 Pu** `MHASignal::waveform_t` `adaptive_feedback_canceller_config::Pu` [private]

Power of input signal delayline.

**5.27.3.8 name\_d\_** `std::string` `adaptive_feedback_canceller_config::name_d_` [private]

**5.27.3.9 name\_lpc\_** `std::string` `adaptive_feedback_canceller_config::name_lpc_` [private]

**5.27.3.10 n\_no\_update\_** `int` `adaptive_feedback_canceller_config::n_no_update_` [private]

**5.27.3.11 no\_iter** `int` `adaptive_feedback_canceller_config::no_iter` [private]

**5.27.3.12 iter** `int` `adaptive_feedback_canceller_config::iter` [private]

**5.27.3.13 PSD\_val** `double` `adaptive_feedback_canceller_config::PSD_val` [private]

**5.27.3.14 v\_G** `MHASignal::waveform_t` `adaptive_feedback_canceller_config::v_G` [private]

**5.27.3.15 s\_U** `MHASignal::waveform_t` `adaptive_feedback_canceller_config::s_U` [private]

**5.27.3.16 s\_E\_afc\_delay** `MHASignal::delay_t` adaptive\_feedback\_canceller\_config←  
  ::s\_E\_afc\_delay [private]

**5.27.3.17 s\_W** `MHASignal::delay_t` adaptive\_feedback\_canceller\_config::s\_W [private]

**5.27.3.18 s\_Wfilt** `MHASignal::ringbuffer_t` adaptive\_feedback\_canceller\_config::s\_Wfilt←  
  [private]

**5.27.3.19 s\_U\_delay** `MHASignal::delay_t` adaptive\_feedback\_canceller\_config::s\_U\_delay←  
  [private]

**5.27.3.20 s\_U\_delayfilt** `MHASignal::ringbuffer_t` adaptive\_feedback\_canceller\_config←  
  ::s\_U\_delayfilt [private]

**5.27.3.21 F\_Ufilt** `MHASignal::waveform_t` adaptive\_feedback\_canceller\_config::F\_Ufilt  
  [private]

**5.27.3.22 s\_Y\_delay** `MHASignal::delay_t` adaptive\_feedback\_canceller\_config::s\_Y\_delay←  
  [private]

**5.27.3.23 s\_Y\_delayfilt** `MHASignal::ringbuffer_t` adaptive\_feedback\_canceller\_config←  
  ::s\_Y\_delayfilt [private]

**5.27.3.24 UbufferPrew** `MHASignal::ringbuffer_t adaptive_feedback_canceller_<config>::UbufferPrew` [private]

**5.27.3.25 s\_LPC** `mha_wave_t adaptive_feedback_canceller_config::s_LPC` [private]

**5.27.3.26 UPrew** `mha_wave_t adaptive_feedback_canceller_config::UPrew` [private]

**5.27.3.27 YPrew** `mha_wave_t adaptive_feedback_canceller_config::YPrew` [private]

**5.27.3.28 EPrew** `mha_wave_t adaptive_feedback_canceller_config::EPrew` [private]

**5.27.3.29 UPrewW** `mha_wave_t adaptive_feedback_canceller_config::UPrewW` [private]

**5.27.3.30 smpl** `mha_wave_t adaptive_feedback_canceller_config::smpl` [private]

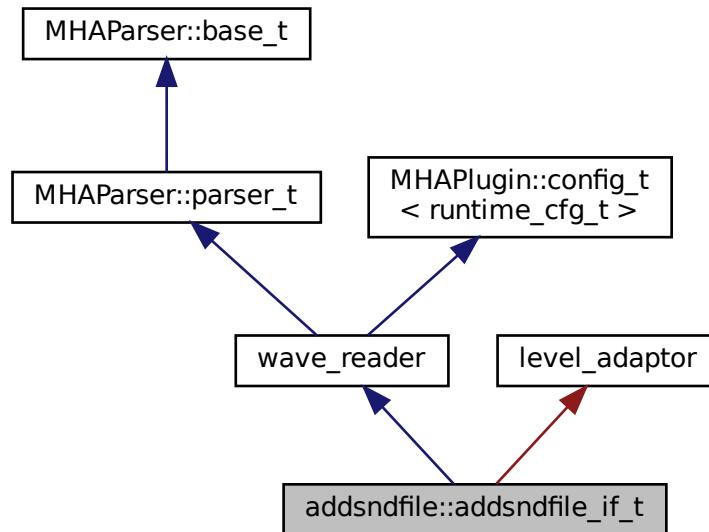
**5.27.3.31 s\_Usmpl** `mha_wave_t* adaptive_feedback_canceller_config::s_Usmpl` [private]

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

- `adaptive_feedback_canceller.h`
- `adaptive_feedback_canceller.cpp`

## 5.28 addsndfile::addsndfile\_if\_t Class Reference

Inheritance diagram for addsndfile::addsndfile\_if\_t:



### Public Member Functions

- **addsndfile\_if\_t ( algo\_comm\_t iac, const std::string &configured\_name)**
- **mha\_wave\_t \* process ( mha\_wave\_t \*)**
- **void prepare ( mhaconfig\_t &)**
- **void release ()**

### Private Member Functions

- **void update ()**
- **void change\_mode ()**
- **void set\_level ()**
- **void scan\_dir ()**

## Private Attributes

- `MHAParser::string_t filename`
- `MHAParser::string_t path`
- `MHAParser::bool_t loop`
- `MHAParser::float_t level`
- `MHAParser::kw_t levelmode`
- `MHAParser::kw_t resamplingmode`
- `MHAParser::vint_t channels`
- `MHAParser::kw_t mode`
- `MHAParser::float_t ramplen`
- `MHAParser::int_t startpos`
- `MHAParser::vint_mon_t mapping`
- `MHAParser::int_mon_t numchannels`
- `MHAParser::int_mon_t mhachannels`
- `MHAParser::int_mon_t active`
- `MHAParser::string_t search_pattern`
- `MHAParser::vstring_mon_t search_result`
- `unsigned int uint_mode`
- `MHAEvents::patchbay_t< addsndfile_if_t > patchbay`

## Additional Inherited Members

### 5.28.1 Constructor & Destructor Documentation

```
5.28.1.1 addsndfile_if_t() addsndfile::addsndfile_if_t::addsndfile_if_t (
    algo_comm_t iac,
    const std::string & configured_name )
```

### 5.28.2 Member Function Documentation

```
5.28.2.1 process() mha_wave_t * addsndfile::addsndfile_if_t::process (
    mha_wave_t * s )
```

**5.28.2.2 `prepare()`** void addsndfile::addsndfile\_if\_t::prepare ( mhaconfig\_t & tf ) [virtual]

Implements **MHAPlugIn::plugin\_t< runtime\_cfg\_t >** (p. [1149](#)).

**5.28.2.3 `release()`** void addsndfile::addsndfile\_if\_t::release ( ) [virtual]

Reimplemented from **MHAPlugIn::plugin\_t< runtime\_cfg\_t >** (p. [1150](#)).

**5.28.2.4 `update()`** void addsndfile::addsndfile\_if\_t::update ( ) [private]

**5.28.2.5 `change_mode()`** void addsndfile::addsndfile\_if\_t::change\_mode ( ) [private]

**5.28.2.6 `set_level()`** void addsndfile::addsndfile\_if\_t::set\_level ( ) [private]

**5.28.2.7 `scan_dir()`** void addsndfile::addsndfile\_if\_t::scan\_dir ( ) [private]

### 5.28.3 Member Data Documentation

**5.28.3.1 `filename`** MHAParser::string\_t addsndfile::addsndfile\_if\_t::filename [private]

**5.28.3.2 `path`** MHAParser::string\_t addsndfile::addsndfile\_if\_t::path [private]

**5.28.3.3 `loop`** `MHAParser::bool_t` `addsndfile::addsndfile_if_t::loop` [private]

**5.28.3.4 `level`** `MHAParser::float_t` `addsndfile::addsndfile_if_t::level` [private]

**5.28.3.5 `levelmode`** `MHAParser::kw_t` `addsndfile::addsndfile_if_t::levelmode` [private]

**5.28.3.6 `resamplingmode`** `MHAParser::kw_t` `addsndfile::addsndfile_if_t::resamplingmode` [private]

**5.28.3.7 `channels`** `MHAParser::vint_t` `addsndfile::addsndfile_if_t::channels` [private]

**5.28.3.8 `mode`** `MHAParser::kw_t` `addsndfile::addsndfile_if_t::mode` [private]

**5.28.3.9 `ramplen`** `MHAParser::float_t` `addsndfile::addsndfile_if_t::ramplen` [private]

**5.28.3.10 `startpos`** `MHAParser::int_t` `addsndfile::addsndfile_if_t::startpos` [private]

**5.28.3.11 `mapping`** `MHAParser::vint_mon_t` `addsndfile::addsndfile_if_t::mapping` [private]

**5.28.3.12 numchannels** `MHAParser::int_mon_t addsndfile::addsndfile_if_t::numchannels` [private]

**5.28.3.13 mhachannels** `MHAParser::int_mon_t addsndfile::addsndfile_if_t::mhachannels` [private]

**5.28.3.14 active** `MHAParser::int_mon_t addsndfile::addsndfile_if_t::active` [private]

**5.28.3.15 search\_pattern** `MHAParser::string_t addsndfile::addsndfile_if_t::search←_pattern` [private]

**5.28.3.16 search\_result** `MHAParser::vstring_mon_t addsndfile::addsndfile_if_t←::search_result` [private]

**5.28.3.17 uint\_mode** `unsigned int addsndfile::addsndfile_if_t::uint_mode` [private]

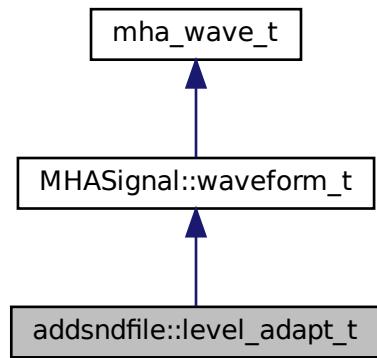
**5.28.3.18 patchbay** `MHAEEvents::patchbay_t< addsndfile_if_t> addsndfile::addsndfile_if_t::patchbay` [private]

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

- `addsndfile.cpp`

## 5.29 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

#### 5.29.1 Constructor & Destructor Documentation

**5.29.1.1 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_ )`

## 5.29.2 Member Function Documentation

**5.29.2.1 update\_frame()** `void addsndfile::level_adapt_t::update_frame ( )`

**5.29.2.2 get\_level()** `mha_real_t addsndfile::level_adapt_t::get_level ( ) const [inline]`

**5.29.2.3 can\_update()** `bool addsndfile::level_adapt_t::can_update ( ) const [inline]`

## 5.29.3 Member Data Documentation

**5.29.3.1 ilen** `unsigned int addsndfile::level_adapt_t::ilen [private]`

**5.29.3.2 pos** `unsigned int addsndfile::level_adapt_t::pos [private]`

**5.29.3.3 wnd** `MHAWindow::fun_t addsndfile::level_adapt_t::wnd [private]`

### 5.29.3.4 l\_new mha\_real\_t addsndfile::level\_adapt\_t::l\_new [private]

### 5.29.3.5 l\_old mha\_real\_t addsndfile::level\_adapt\_t::l\_old [private]

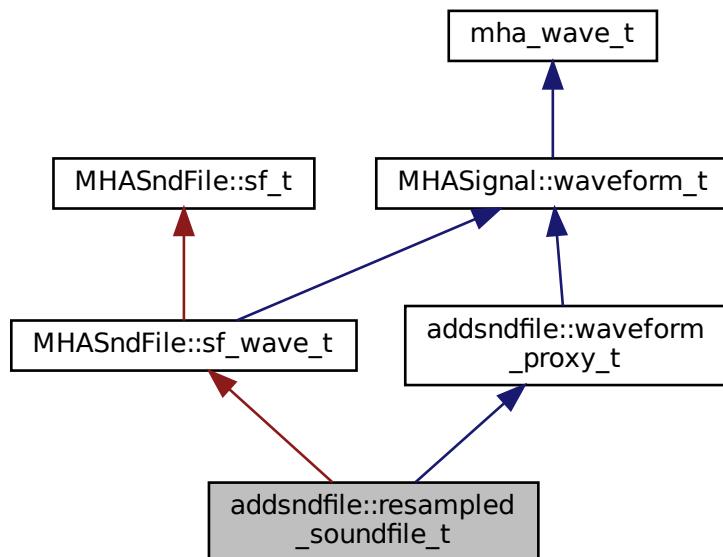
The documentation for this class was generated from the following file:

- `addsndfile.cpp`

## 5.30 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

### 5.30.1 Detailed Description

Reads sound from file and resamples it if necessary and wanted.

Sound data can then be used by addsndfile.

### 5.30.2 Constructor & Destructor Documentation

```
5.30.2.1 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

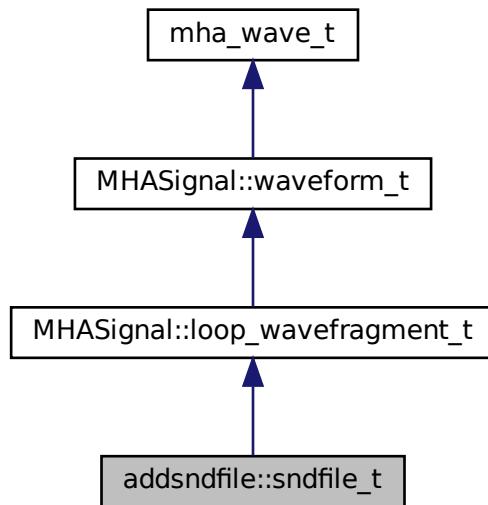
<b>MHA_Error</b> (p. <a href="#">760</a> )	If the sampling rate of the file does not match the sampling rate of the MHA and DONT_RESAMPLE_STRICT was requested. If resampling failed (e.g. due to non-rational quotient of MHA sampling rate and sound file sampling rate).
--	--

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

- [addsndfile.cpp](#)

## 5.31 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

#### 5.31.1 Constructor & Destructor Documentation

```
5.31.1.1 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 )
```

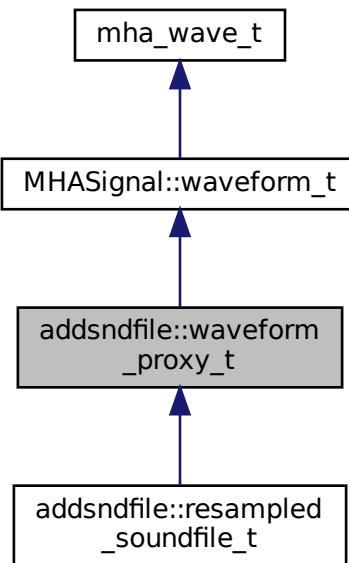
The documentation for this class was generated from the following file:

- **addsndfile.cpp**

## 5.32 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

### 5.32.1 Detailed Description

Class helps to specify which instance of MHASignal\_waveform\_t parent instance is meant in **resampled\_soundfile\_t** (p. [257](#)).

### 5.32.2 Constructor & Destructor Documentation

#### 5.32.2.1 **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**

## 5.33 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**

### 5.33.1 Detailed Description

```
template<class F>
class ADM::ADM< F >
```

Adaptive differential microphone, working for speech frequency range.

### 5.33.2 Constructor & Destructor Documentation

```
5.33.2.1 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

<i>fs</i>	Sampling rate / Hz
<i>dist</i>	Distance between physical microphones / m
<i>lp_order</i>	Filter order of FIR lowpass filter used for adaptation

### 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> $\leq 1$ 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

### 5.33.3 Member Function Documentation

**5.33.3.1 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 $\geq 0$ , 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

**5.33.3.2 `beta()`** template<class F >  
F **ADM::ADM**< F >::beta ( ) const [inline]

## 5.33.4 Member Data Documentation

**5.33.4.1 `m_delay_front`** template<class F >  
**Delay**<F> **ADM::ADM**< F >::m\_delay\_front [private]

**5.33.4.2 `m_delay_back`** template<class F >  
**Delay**<F> **ADM::ADM**< F >::m\_delay\_back [private]

**5.33.4.3 `m_lp_bf`** template<class F >  
**Linearphase\_FIR**<F> **ADM::ADM**< F >::m\_lp\_bf [private]

**5.33.4.4 `m_lp_result`** template<class F >  
**Linearphase\_FIR**<F> **ADM::ADM**< F >::m\_lp\_result [private]

**5.33.4.5 `m_decomb`** template<class F >  
**Linearphase\_FIR**<F> **ADM::ADM**< F >::m\_decomb [private]

**5.33.4.6 `m_beta`** template<class F >  
F **ADM::ADM**< F >::m\_beta [private]

**5.33.4.7 m\_mu\_beta** template<class F >  
F ADM::ADM< F >::m\_mu\_beta [private]

**5.33.4.8 m\_powerfilter\_coeff** template<class F >  
F ADM::ADM< F >::m\_powerfilter\_coeff [private]

**5.33.4.9 m\_powerfilter\_norm** template<class F >  
F ADM::ADM< F >::m\_powerfilter\_norm [private]

**5.33.4.10 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

## 5.34 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*

### 5.34.1 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.

### 5.34.2 Constructor & Destructor Documentation

```
5.34.2.1 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).

**5.34.2.2 ~Delay()** template<class F >  
ADM::Delay< F >::~ Delay

### 5.34.3 Member Function Documentation

**5.34.3.1 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.

### 5.34.4 Member Data Documentation

**5.34.4.1 m\_fullsamples** template<class F >  
unsigned ADM::Delay< F >::m\_fullsamples [private]

Integer part of delay.

---

**5.34.4.2 m\_coeff** template<class F >  
F **ADM::Delay**< F >::m\_coeff [private]

coefficient for 1st order IIR lowpass filter which does the subsample delay

**5.34.4.3 m\_norm** template<class F >  
F **ADM::Delay**< F >::m\_norm [private]

normalization for the IIR subsample delay filter

**5.34.4.4 m\_state** template<class F >  
F\* **ADM::Delay**< F >::m\_state [private]

Ringbuffer: Delayline.

**5.34.4.5 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

## 5.35 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*

### 5.35.1 Detailed Description

```
template<class F>
class ADM::Linearphase_FIR< F >
```

An efficient linear-phase fir filter implementation.

### 5.35.2 Constructor & Destructor Documentation

```
5.35.2.1 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.)

```
5.35.2.2 ~Linearphase_FIR() template<class F >
ADM::Linearphase_FIR< F >::~ Linearphase_FIR
```

### 5.35.3 Member Function Documentation

**5.35.3.1 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

### 5.35.4 Member Data Documentation

**5.35.4.1 m\_order** template<class F >  
unsigned ADM::Linearphase\_FIR< F >::m\_order [private]

The filter order of this linear-phase FIR filter.

**5.35.4.2 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

**5.35.4.3 m\_output** template<class F >  
F\* ADM::Linearphase\_FIR< F >::m\_output [private]

Ringbuffer for building future output.

```
5.35.4.4 m_now template<class F >
unsigned ADM::Linearphase_FIR< F >::m_now [private]
```

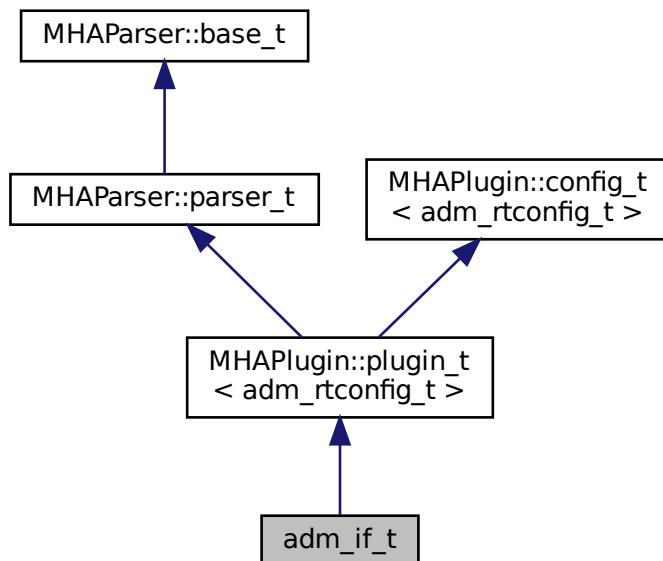
current start of ringbuffer

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

- **adm.hh**

## 5.36 adm\_if\_t Class Reference

Inheritance diagram for adm\_if\_t:



### Public Member Functions

- `adm_if_t ( algo_comm_t ac, const std::string &configured_name)`
- `mha_wave_t * process ( mha_wave_t *in)`
- `virtual void prepare ( mhaconfig_t &)`
- `virtual void release ()`

### Private Member Functions

- `void update ()`
- `bool is_prepared ()`

## Private Attributes

- `MHASignal::waveform_t * out`
- `MHAParser::vint_t front_channels`
- `MHAParser::vint_t rear_channels`
- `MHAParser::vfloat_t distances`
- `MHAParser::int_t lp_order`
- `MHAParser::int_t decomb_order`
- `MHAParser::int_t bypass`
- `MHAParser::float_t beta`
- `MHAParser::vfloat_t mu_beta`
- `MHAParser::vfloat_t tau_beta`
- `MHAParser::vfloat_mon_t coeff_lp`
- `MHAParser::vfloat_mon_t coeff_decomb`
- `MHAParser::int_t adaptation_ratio`
- `unsigned input_channels`
- `int framecnt`
- `mha_real_t srate`
- `MHAEvents::patchbay_t< adm_if_t > patchbay`

## Additional Inherited Members

### 5.36.1 Constructor & Destructor Documentation

```
5.36.1.1 adm_if_t() adm_if_t::adm_if_t (
    algo_comm_t ac,
    const std::string & configured_name )
```

### 5.36.2 Member Function Documentation

```
5.36.2.1 process() mha_wave_t * adm_if_t::process (
    mha_wave_t * in )
```

```
5.36.2.2 prepare() void adm_if_t::prepare (
    mhaconfig_t & cfg ) [virtual]
```

Implements **MHAPlugIn::plugin\_t<adm\_rtconfig\_t>** (p. 1149).

```
5.36.2.3 release() void adm_if_t::release () [virtual]
```

Reimplemented from **MHAPlugIn::plugin\_t<adm\_rtconfig\_t>** (p. 1150).

```
5.36.2.4 update() void adm_if_t::update () [private]
```

```
5.36.2.5 is_prepared() bool adm_if_t::is_prepared () [inline], [private]
```

### 5.36.3 Member Data Documentation

```
5.36.3.1 out MHASignal::waveform_t* adm_if_t::out [private]
```

```
5.36.3.2 front_channels MHAParser::vint_t adm_if_t::front_channels [private]
```

```
5.36.3.3 rear_channels MHAParser::vint_t adm_if_t::rear_channels [private]
```

```
5.36.3.4 distances MHAParser::vfloat_t adm_if_t::distances [private]
```

**5.36.3.5 ip\_order** `MHAParser::int_t adm_if_t::lp_order [private]`

**5.36.3.6 decomb\_order** `MHAParser::int_t adm_if_t::decomb_order [private]`

**5.36.3.7 bypass** `MHAParser::int_t adm_if_t::bypass [private]`

**5.36.3.8 beta** `MHAParser::float_t adm_if_t::beta [private]`

**5.36.3.9 mu\_beta** `MHAParser::vfloat_t adm_if_t::mu_beta [private]`

**5.36.3.10 tau\_beta** `MHAParser::vfloat_t adm_if_t::tau_beta [private]`

**5.36.3.11 coeff\_lp** `MHAParser::vfloat_mon_t adm_if_t::coeff_lp [private]`

**5.36.3.12 coeff\_decomb** `MHAParser::vfloat_mon_t adm_if_t::coeff_decomb [private]`

**5.36.3.13 adaptation\_ratio** `MHAParser::int_t adm_if_t::adaptation_ratio [private]`

**5.36.3.14 input\_channels** unsigned adm\_if\_t::input\_channels [private]

**5.36.3.15 framecnt** int adm\_if\_t::framecnt [private]

**5.36.3.16 srate** mha\_real\_t adm\_if\_t::srate [private]

**5.36.3.17 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

## 5.37 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.*

### 5.37.1 Member Typedef Documentation

#### 5.37.1.1 **adm\_t** `typedef ADM::ADM< mha_real_t> adm_rtconfig_t::adm_t`

### 5.37.2 Constructor & Destructor Documentation

#### 5.37.2.1 **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 adaptation_ratio frames
<i>front_channels</i>	Parser's front_channels setting
<i>rear_channels</i>	Parser's front_channels setting
<i>fs</i>	Sampling rate / Hz
<i>distances</i>	Distances between microphones / m
<i>lp_order</i>	Filter order of FIR lowpass filter for adaptation
<i>decomb_order</i>	Filter order of FIR compensation filter (compensates for comb filter characteristic)
<i>tau_beta</i>	Time constants of the lowpass filter used for averaging the power of the output signal used for adaptation
<i>mu_beta</i>	Adaptation step sizes

**5.37.2.2 ~adm\_rtconfig\_t()** `adm_rtconfig_t::~adm_rtconfig_t ( ) [virtual]`

### 5.37.3 Member Function Documentation

**5.37.3.1 check\_index()** `void adm_rtconfig_t::check_index ( unsigned index ) const [inline], [private]`

Index checking for all internal arrays.

#### Exceptions

<b>MHA_Error</b> (p. <a href="#">760</a> )	if index out of range.
--	------------------------

**5.37.3.2 num\_adms()** `size_t adm_rtconfig_t::num_adms ( ) const [inline]`

**5.37.3.3 `adm()`** `adm_t& adm_rtconfig_t::adm (`  
`unsigned index ) [inline]`

Returns adm object number index.

**5.37.3.4 `front_channel()`** `int adm_rtconfig_t::front_channel (`  
`unsigned index ) const [inline]`

Returns index of front channel for adm number index.

**5.37.3.5 `rear_channel()`** `int adm_rtconfig_t::rear_channel (`  
`unsigned index ) const [inline]`

Returns index of rear channel for adm number index.

**5.37.3.6 `get_adaptation_ratio()`** `int adm_rtconfig_t::get_adaptation_ratio ( ) const`  
[inline]

## 5.37.4 Member Data Documentation

**5.37.4.1 `front_channels`** `std::vector<int> adm_rtconfig_t::front_channels [private]`

Indices of channels containing the signals from the front microphones.

**5.37.4.2 `rear_channels`** `std::vector<int> adm_rtconfig_t::rear_channels [private]`

Indices of channels containing the signals from the rear microphones.

**5.37.4.3 adaptation\_ratio** int adm\_rtconfig\_t::adaptation\_ratio [private]

Prescale.

**5.37.4.4 lp\_coeffs** MHASignal::waveform\_t\* adm\_rtconfig\_t::lp\_coeffs [private]

Lowpass filter coefficients.

**5.37.4.5 decomb\_coeffs** std::vector< MHASignal::waveform\_t\*> adm\_rtconfig\_t::decomb\_coeffs [private]

Decomb-Filter coefficients.

**5.37.4.6 adms** std::vector< adm\_t \*> adm\_rtconfig\_t::adms [private]

ADMs.

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

- adm.cpp

## 5.38 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.*

### 5.38.1 Detailed Description

A reference handle for algorithm communication variables.

This structure contains a coontrol handle and a set of function pointers for sharing variables within one processing chain. See also section **Communication between algorithms** (p. 23).

### 5.38.2 Member Data Documentation

**5.38.2.1 handle algo\_comm\_t::handle**

AC variable control handle.

**5.38.2.2 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 <b>comm_var_t</b> (p. 359)

**Returns**

Error code or zero on success

**5.38.2.3 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

#### **5.38.2.4 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

#### **5.38.2.5 insert\_var\_double int(\* algo\_comm\_t::insert\_var\_double) (void \*, const char \*, double \*)**

#### **5.38.2.6 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

**5.38.2.7 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

**5.38.2.8 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

**5.38.2.9 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. 359) of a variable of the given name. If no variable of that name exists, an error code is returned.

**Parameters**

<i>h</i>	AC handle
<i>n</i>	name of variable
<i>v</i>	pointer to a AC variable object

**Returns**

Error code or zero on success

**5.38.2.10 `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

**5.38.2.11 `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

**5.38.2.12 get\_var\_double** `int(* algo_comm_t::get_var_double) (void *, const char *, double *)`

**5.38.2.13 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.

### 5.38.2.14 `get_error` `algo_comm_t::get_error`

Convert AC error codes into human readable error messages.

#### Parameters

<code>e</code>	Error code
----------------	------------

#### Returns

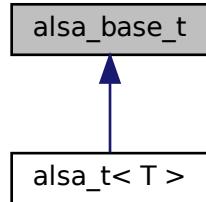
Error message

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

- `mha.hh`
- `mha_algo_comm.cpp`

## 5.39 `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. 836) buffer, then update the pointer given as parameter to point to the internal structure.*
- virtual bool `write(mha_wave_t *)=0`  
*write audio samples from the given waveform buffer to the sound device.*

## Public Attributes

- **snd\_pcm\_t \* pcm**  
*The underlying alsa handle to this sound card.*

## 5.39.1 Constructor & Destructor Documentation

**5.39.1.1 `alsa_base_t()`** `alsa_base_t::alsa_base_t ( ) [inline]`

**5.39.1.2 `~alsa_base_t()`** `virtual alsa_base_t::~alsa_base_t ( ) [virtual], [default]`

## 5.39.2 Member Function Documentation

**5.39.2.1 `start()`** `virtual void alsa_base_t::start ( ) [pure virtual]`

start puts alsa device in usable state

Implemented in **alsa\_t< T >** (p. [293](#)).

**5.39.2.2 `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](#)).

**5.39.2.3 `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. 836) 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).

**5.39.2.4 `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).

### 5.39.3 Member Data Documentation

**5.39.3.1 `pcm`** `snd_pcm_t* alsa_base_t::pcm`

The underlying alsa handle to this sound card.

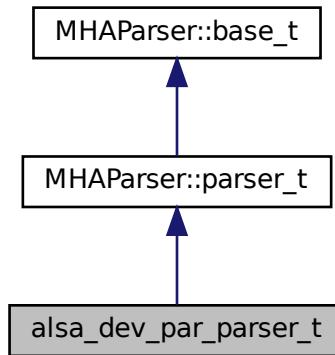
The documentation for this class was generated from the following file:

- **MHAIoalsa.cpp**

## 5.40 `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

#### 5.40.1 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.

## 5.40.2 Constructor & Destructor Documentation

**5.40.2.1 `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
-------------------------	---------------------

## 5.40.3 Member Data Documentation

**5.40.3.1 `device`** `MHAParser::string_t alsa_dev_par_parser_t::device`

Name of the device in the alsal world, like "hw:0.0", "default", etc.

**5.40.3.2 `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.

**5.40.3.3 `stream_dir`** `snd_pcm_stream_t alsa_dev_par_parser_t::stream_dir`

Remember the direction (capture/playback) of this device.

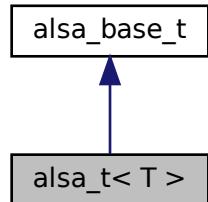
The documentation for this class was generated from the following file:

- **MHAIOalsa.cpp**

## 5.41 `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. 836) buffer, then update the pointer given as parameter to point to the internal structure.*
- `bool write ( mha_wave_t *) override`  
*write audio samples from the given waveform buffer to the sound device.*

### Private Attributes

- `unsigned int channels`
- `unsigned int fragsize`
- `T * buffer`
- `std::vector< mha_real_t > frame_data`
- `MHASignal::waveform_t wave`  
*internal buffer to store sound samples coming from the sound card.*
- `const mha_real_t gain`
- `const mha_real_t invgain`
- `snd_pcm_format_t pcm_format`

## Additional Inherited Members

### 5.41.1 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.

### 5.41.2 Constructor & Destructor Documentation

```
5.41.2.1 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

```
5.41.2.2 ~alsa_t() template<typename T >
alsa_t< T >::~ alsa_t
```

Destructor closes the sound device.

### 5.41.3 Member Function Documentation

#### 5.41.3.1 **start()** template<typename T >

```
void alsa_t< T >::start [override], [virtual]
```

start puts alsa device in usable state

Implements **alsa\_base\_t** (p. [287](#)).

#### 5.41.3.2 **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](#)).

#### 5.41.3.3 **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. [836](#)) 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. [287](#)).

#### 5.41.3.4 **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](#)).

## 5.41.4 Member Data Documentation

**5.41.4.1 channels** template<typename T >  
unsigned int **alsa\_t**< T >::channels [private]

**5.41.4.2 fragsize** template<typename T >  
unsigned int **alsa\_t**< T >::fragsize [private]

**5.41.4.3 buffer** template<typename T >  
T\* **alsa\_t**< T >::buffer [private]

**5.41.4.4 frame\_data** template<typename T >  
std::vector< **mha\_real\_t**> **alsa\_t**< T >::frame\_data [private]

**5.41.4.5 wave** template<typename T >  
**MHASignal::waveform\_t** **alsa\_t**< T >::wave [private]

internal buffer to store sound samples coming from the sound card.

**5.41.4.6 gain** template<typename T >  
const **mha\_real\_t** **alsa\_t**< T >::gain [private]

**5.41.4.7 invgain** template<typename T >  
const **mha\_real\_t** **alsa\_t**< T >::invgain [private]

```
5.41.4.8 pcm_format template<typename T >
snd_pcm_format_t alsa_t< T >::pcm_format [private]
```

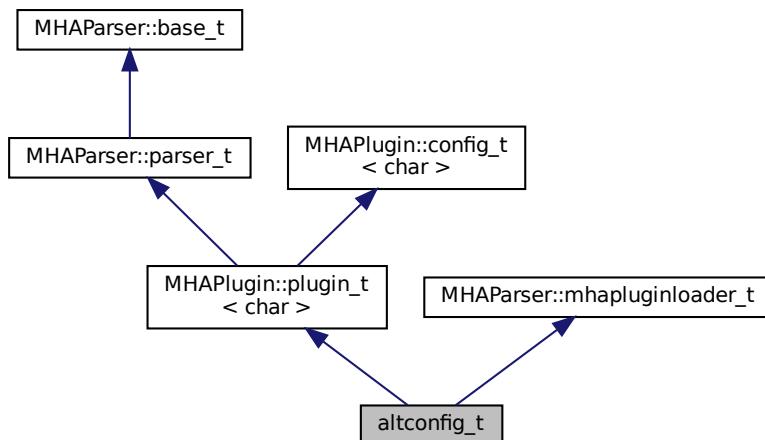
The documentation for this class was generated from the following file:

- **MHAIoalsa.cpp**

## 5.42 altconfig\_t Class Reference

Single class implementing plugin altconfig.

Inheritance diagram for altconfig\_t:



### Public Member Functions

- **altconfig\_t ( algo\_comm\_t iac, const std::string &configured\_name)**  
*Constructor initializes an instance of the altconfig plugin.*
- **void prepare ( mhaconfig\_t &cf)**  
*Invoked by MHA when this plugin should prepare for signal processing.*
- **void release ()**  
*Invoked by MHA when this plugin should call its release function.*

## Private Member Functions

- void **on\_set\_algos ()**  
*Callback executed when the configuration variable "algos" is written to at run time.*
- void **on\_set\_select ()**  
*Callback executed when the configuration variable "select" is successfully written to at run time.*
- void **event\_select\_all ()**  
*Callback executed when the configuration variable "selectall" is written to at run time.*
- std::map< std::string, 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

### 5.42.1 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 "algos" and "select" even though the alternatives that can be selected in this plugin are not algorithms or plugins but configuration commands in order to be interface-compatible with plugin altplugs. mhacontrol has a UI area that automatically fills with the alternatives found in either altplugs or altconfig if the complete MHA loads exactly one plugin with this interface.

### 5.42.2 Constructor & Destructor Documentation

#### 5.42.2.1 altconfig\_t() altconfig\_t::altconfig\_t (

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

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>configured_name</i>	Configured name of this plugin.

**5.42.3 Member Function Documentation**

**5.42.3.1 `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. [1149](#)).

**5.42.3.2 `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. [1150](#)).

**5.42.3.3 `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"

**5.42.3.4 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.

**5.42.3.5 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.

**5.42.3.6 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

**5.42.3.7 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
--------------	------------------

#### 5.42.4 Member Data Documentation

**5.42.4.1 parser\_algos** `MHAParser::vstring_t altconfig_t::parser_algos [private]`

**5.42.4.2 select\_plug** `MHAParser::kw_t altconfig_t::select_plug [private]`

**5.42.4.3 selectall** `MHAParser::bool_t altconfig_t::selectall [private]`

**5.42.4.4 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.

**5.42.4.5 patchbay** `MHAEvents::patchbay_t< altconfig_t> altconfig_t::patchbay [private]`

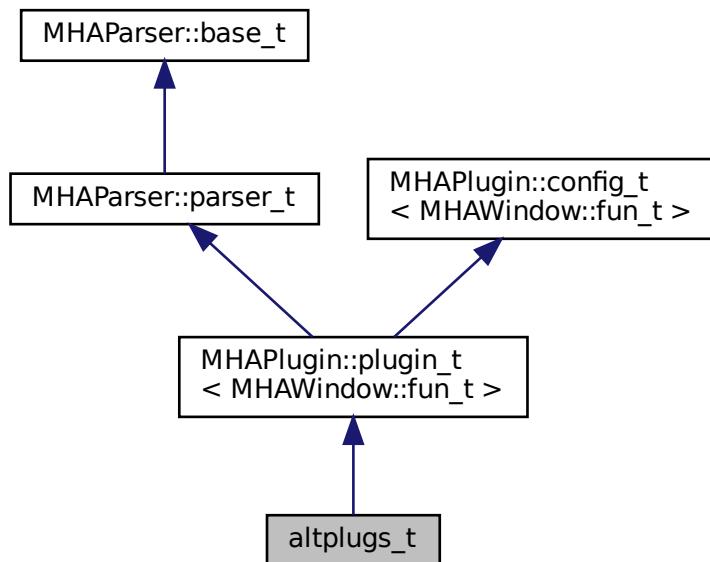
Configuration event broker.

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

- **altconfig.hh**
- **altconfig.cpp**

## 5.43 altplugs\_t Class Reference

Inheritance diagram for altplugs\_t:



### Public Member Functions

- `altplugs_t ( algo_comm_t iac, const std::string &configured_name)`
- `void prepare ( mhaconfig_t &)`
- `void release ()`
- `void process ( mha_wave_t *, mha_wave_t **)`
- `void process ( mha_spec_t *, mha_wave_t **)`
- `void process ( mha_wave_t *, mha_spec_t **)`
- `void process ( mha_spec_t *, mha_spec_t **)`
- `virtual std::string parse (const std::string &arg)`
- `virtual void parse (const char *a1, char *a2, unsigned int a3)`

### Private Member Functions

- `void event_set_plugs ()`
- `void event_add_plug ()`
- `void event_delete_plug ()`
- `void event_select_plug ()`
- `void update_selector_list ()`
- `void update_ramplen ()`
- `void proc_ramp ( mha_wave_t *s)`

## Private Attributes

- `MHAParser::bool_t use_own_ac`
- `MHAParser::vstring_t parser_plugs`
- `MHAParser::string_t add_plug`
- `MHAParser::string_t delete_plug`
- `MHAParser::float_t ramplen`
- `MHAParser::kw_t select_plug`
- `MHAParser::parser_t current`
- `MHAParser::vstring_mon_t nondefault_labels`
- `std::vector< mhaplug_cfg_t * > plugs`
- `mhaplug_cfg_t * selected_plug`
- `MHAEvents::patchbay_t< altplugs_t > patchbay`
- `MHASignal::waveform_t * fallback_wave`
- `MHASignal::spectrum_t * fallback_spec`
- `mhaconfig_t cfin`
- `mhaconfig_t cout`
- `bool prepared`
- `bool added_via_plugs`
- `unsigned int ramp_counter`
- `unsigned int ramp_len`

## Additional Inherited Members

### 5.43.1 Constructor & Destructor Documentation

```
5.43.1.1 altplugs_t() altplugs_t::altplugs_t (
    algo_comm_t iac,
    const std::string & configured_name )
```

### 5.43.2 Member Function Documentation

```
5.43.2.1 prepare() void altplugs_t::prepare (
    mhaconfig_t & cf ) [virtual]
```

Implements `MHAPlugin::plugin_t< MHAWindow::fun_t >` (p. [1149](#)).

**5.43.2.2 `release()`** void altplugs\_t::release ( ) [virtual]

Reimplemented from **MHAPlugIn::plugin\_t**<**MHAWindow::fun\_t**> (p. 1150).

**5.43.2.3 `process() [1/4]`** void altplugs\_t::process (

```
mha_wave_t * sIn,
mha_wave_t ** sOut )
```

**5.43.2.4 `process() [2/4]`** void altplugs\_t::process (

```
mha_spec_t * sIn,
mha_wave_t ** sOut )
```

**5.43.2.5 `process() [3/4]`** void altplugs\_t::process (

```
mha_wave_t * sIn,
mha_spec_t ** sOut )
```

**5.43.2.6 `process() [4/4]`** void altplugs\_t::process (

```
mha_spec_t * sIn,
mha_spec_t ** sOut )
```

**5.43.2.7 `parse() [1/2]`** std::string altplugs\_t::parse (

```
const std::string & arg ) [virtual]
```

Reimplemented from **MHAParser::base\_t** (p. 1031).

**5.43.2.8 `parse() [2/2]`** virtual void altplugs\_t::parse (

```
const char * a1,
char * a2,
unsigned int a3 ) [inline], [virtual]
```

Reimplemented from **MHAParser::base\_t** (p. 1032).

**5.43.2.9 event\_set\_plugs()** void altplugs\_t::event\_set\_plugs ( ) [private]

**5.43.2.10 event\_add\_plug()** void altplugs\_t::event\_add\_plug ( ) [private]

**5.43.2.11 event\_delete\_plug()** void altplugs\_t::event\_delete\_plug ( ) [private]

**5.43.2.12 event\_select\_plug()** void altplugs\_t::event\_select\_plug ( ) [private]

**5.43.2.13 update\_selector\_list()** void altplugs\_t::update\_selector\_list ( ) [private]

**5.43.2.14 update\_ramplen()** void altplugs\_t::update\_ramplen ( ) [private]

**5.43.2.15 proc\_ramp()** void altplugs\_t::proc\_ramp (   
      mha\_wave\_t \* s ) [private]

### 5.43.3 Member Data Documentation

**5.43.3.1 use\_own\_ac** MHAParser::bool\_t altplugs\_t::use\_own\_ac [private]

**5.43.3.2 parser\_plugs** `MHAParser::vstring_t` `altplugs_t::parser_plugs` [private]

**5.43.3.3 add\_plug** `MHAParser::string_t` `altplugs_t::add_plug` [private]

**5.43.3.4 delete\_plug** `MHAParser::string_t` `altplugs_t::delete_plug` [private]

**5.43.3.5 ramplen** `MHAParser::float_t` `altplugs_t::ramplen` [private]

**5.43.3.6 select\_plug** `MHAParser::kw_t` `altplugs_t::select_plug` [private]

**5.43.3.7 current** `MHAParser::parser_t` `altplugs_t::current` [private]

**5.43.3.8 nondefault\_labels** `MHAParser::vstring_mon_t` `altplugs_t::nondefault_labels` [private]

**5.43.3.9 plugs** `std::vector< mhaplug_cfg_t*>` `altplugs_t::plugs` [private]

**5.43.3.10 selected\_plug** `mhaplug_cfg_t*` `altplugs_t::selected_plug` [private]

**5.43.3.11 patchbay** `MHAEEvents::patchbay_t< altplugs_t> altplugs_t::patchbay [private]`

**5.43.3.12 fallback\_wave** `MHASignal::waveform_t* altplugs_t::fallback_wave [private]`

**5.43.3.13 fallback\_spec** `MHASignal::spectrum_t* altplugs_t::fallback_spec [private]`

**5.43.3.14 cfin** `mhaconfig_t altplugs_t::cfin [private]`

**5.43.3.15 cfout** `mhaconfig_t altplugs_t::cfout [private]`

**5.43.3.16 prepared** `bool altplugs_t::prepared [private]`

**5.43.3.17 added\_via\_plugs** `bool altplugs_t::added_via_plugs [private]`

**5.43.3.18 ramp\_counter** `unsigned int altplugs_t::ramp_counter [private]`

**5.43.3.19 ramp\_len** `unsigned int altplugs_t::ramp_len [private]`

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

- **altplugs.cpp**

## 5.44 analysepath\_t Class Reference

### Public Member Functions

- **analysepath\_t** (unsigned int nchannels\_in, unsigned int inner\_fragsize, int **priority**, **MHAProc\_wave2wave\_t** inner\_proc\_wave2wave, **MHAProc\_wave2spec\_t** inner\_proc\_wave2spec, void \*ilibdata, **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

### 5.44.1 Constructor & Destructor Documentation

```
5.44.1.1 analysepath_t() analysepath_t::analysepath_t (
    unsigned int nchannels_in,
    unsigned int inner_fragsize,
    int priority,
    MHAProc_wave2wave_t inner_proc_wave2wave,
    MHAProc_wave2spec_t inner_proc_wave2spec,
    void * ilibdata,
    algo_comm_t outer_ac,
    const MHA_AC::acspace2matrix_t & acspace_template,
    mha_domain_t inner_out_domain,
    unsigned int fifo_len_blocks )
```

5.44.1.2 ~analysepath\_t() analysepath\_t::~analysepath\_t ( ) [virtual]

## 5.44.2 Member Function Documentation

5.44.2.1 rt\_process() void analysepath\_t::rt\_process (
 mha\_wave\_t \* outer\_input )

5.44.2.2 svc() int analysepath\_t::svc ( ) [virtual]

## 5.44.3 Member Data Documentation

5.44.3.1 inner\_process\_wave2wave MHAProc\_wave2wave\_t analysepath\_t::inner\_←
process\_wave2wave [private]

5.44.3.2 inner\_process\_wave2spec MHAProc\_wave2spec\_t analysepath\_t::inner\_←
process\_wave2spec [private]

**5.44.3.3 inner\_input** `MHASignal::waveform_t` `analysepath_t::inner_input` [private]

**5.44.3.4 libdata** `void*` `analysepath_t::libdata` [private]

**5.44.3.5 wave\_fifo** `mha_fifo_lf_t< mha_real_t>` `analysepath_t::wave_fifo` [private]

**5.44.3.6 ac\_fifo** `mha_fifo_lf_t< MHA_AC::acspace2matrix_t>` `analysepath_t::ac_fifo` [private]

**5.44.3.7 inner\_ac\_copy** `MHA_AC::acspace2matrix_t` `analysepath_t::inner_ac_copy` [private]

**5.44.3.8 outer\_ac\_copy** `MHA_AC::acspace2matrix_t` `analysepath_t::outer_ac_copy` [private]

**5.44.3.9 outer\_ac** `algo_comm_t` `analysepath_t::outer_ac` [private]

**5.44.3.10 inner\_out\_domain** `mha_domain_t` `analysepath_t::inner_out_domain` [private]

**5.44.3.11 inner\_error** `MHA_Error` `analysepath_t::inner_error` [private]

**5.44.3.12 has\_inner\_error** bool analysepath\_t::has\_inner\_error [private]

**5.44.3.13 flag\_terminate\_inner\_thread** bool analysepath\_t::flag\_terminate\_inner\_thread [private]

**5.44.3.14 input\_to\_process** int analysepath\_t::input\_to\_process [private]

**5.44.3.15 ProcessMutex** pthread\_mutex\_t analysepath\_t::ProcessMutex [private]

**5.44.3.16 attr** pthread\_attr\_t analysepath\_t::attr [private]

**5.44.3.17 priority** struct sched\_param analysepath\_t::priority [private]

**5.44.3.18 scheduler** int analysepath\_t::scheduler [private]

**5.44.3.19 thread** pthread\_t analysepath\_t::thread [private]

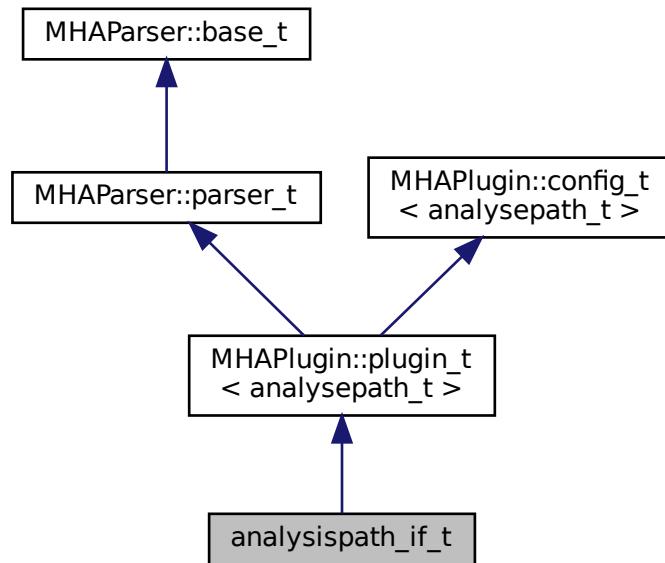
**5.44.3.20 cond\_to\_process** pthread\_cond\_t analysepath\_t::cond\_to\_process [private]

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

- **analysispath.cpp**

## 5.45 analysispath\_if\_t Class Reference

Inheritance diagram for analysispath\_if\_t:



### Public Member Functions

- `analysispath_if_t ( algo_comm_t iac, const std::string &configured_name)`
- `mha_wave_t * process ( mha_wave_t *)`
- `void prepare ( mhaconfig_t &)`
- `void release ()`
- `~analysispath_if_t ()`

### Private Member Functions

- `void loadlib ()`

### Private Attributes

- `MHAEvents::patchbay_t< analysispath_if_t > patchbay`
- `MHAParser::string_t libname`
- `MHAParser::int_t fragsize`
- `MHAParser::int_t fifolen`
- `MHAParser::int_t priority`
- `MHAParser::vstring_t vars`
- `plug_t * plug`
- `std::string algo`
- `MHA_AC::acspace2matrix_t * acspace_template`

## Additional Inherited Members

### 5.45.1 Constructor & Destructor Documentation

**5.45.1.1 analysispath\_if\_t()** analysispath\_if\_t::analysispath\_if\_t (   
     algo\_comm\_t iac,  
     const std::string & configured\_name )

**5.45.1.2 ~analysispath\_if\_t()** analysispath\_if\_t::~analysispath\_if\_t ( )

### 5.45.2 Member Function Documentation

**5.45.2.1 process()** mha\_wave\_t \* analysispath\_if\_t::process (   
     mha\_wave\_t \* s )

**5.45.2.2 prepare()** void analysispath\_if\_t::prepare (   
     mhaconfig\_t & conf ) [virtual]

Implements **MHAPlugIn::plugin\_t< analysepath\_t >** (p. [1149](#)).

**5.45.2.3 release()** void analysispath\_if\_t::release ( ) [virtual]

Reimplemented from **MHAPlugIn::plugin\_t< analysepath\_t >** (p. [1150](#)).

**5.45.2.4 loadlib()** void analysispath\_if\_t::loadlib ( ) [private]

### 5.45.3 Member Data Documentation

**5.45.3.1 patchbay** `MHAEVENTS::patchbay_t< analysispath_if_t > analysispath_if_t::patchbay` [private]

**5.45.3.2 libname** `MHAPARSER::string_t analysispath_if_t::libname` [private]

**5.45.3.3 fragsize** `MHAPARSER::int_t analysispath_if_t::fragsize` [private]

**5.45.3.4 fifolen** `MHAPARSER::int_t analysispath_if_t::fifolen` [private]

**5.45.3.5 priority** `MHAPARSER::int_t analysispath_if_t::priority` [private]

**5.45.3.6 vars** `MHAPARSER::vstring_t analysispath_if_t::vars` [private]

**5.45.3.7 plug** `plug_t* analysispath_if_t::plug` [private]

**5.45.3.8 algo** `std::string analysispath_if_t::algo` [private]

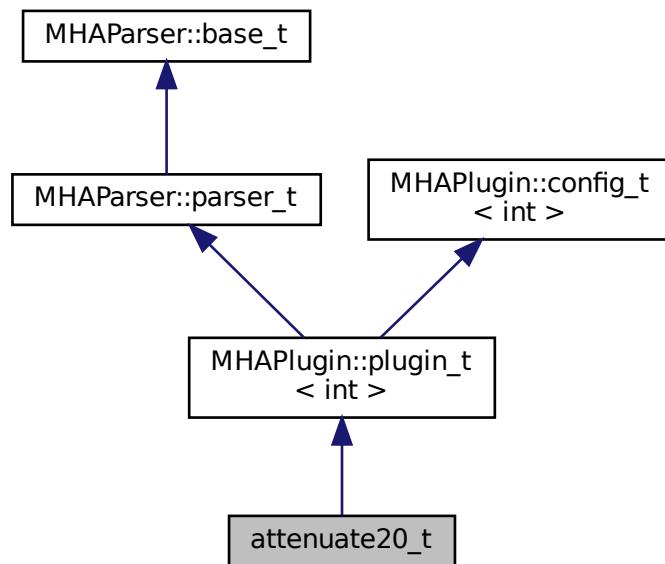
**5.45.3.9 acspace\_template** `MHA_AC::acspace2matrix_t*` `analysispath_if_t::acspace_`↔  
template [private]

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

- `analysispath.cpp`

## 5.46 attenuate20\_t Class Reference

Inheritance diagram for attenuate20\_t:



### Public Member Functions

- `attenuate20_t ( algo_comm_t iac, const std::string &configured_name)`
- `void release (void) override`
- `void prepare ( mhaconfig_t &signal_info) override`
- `mha_wave_t * process ( mha_wave_t *signal)`

### Additional Inherited Members

#### 5.46.1 Constructor & Destructor Documentation

```
5.46.1.1 attenuate20_t() attenuate20_t::attenuate20_t (
    algo_comm_t iac,
    const std::string & configured_name ) [inline]
```

## 5.46.2 Member Function Documentation

```
5.46.2.1 release() void attenuate20_t::release (
    void ) [inline], [override], [virtual]
```

Reimplemented from **MHAPlugIn::plugin\_t< int >** (p. 1150).

```
5.46.2.2 prepare() void attenuate20_t::prepare (
    mhaconfig_t & signal_info ) [inline], [override], [virtual]
```

Implements **MHAPlugIn::plugin\_t< int >** (p. 1149).

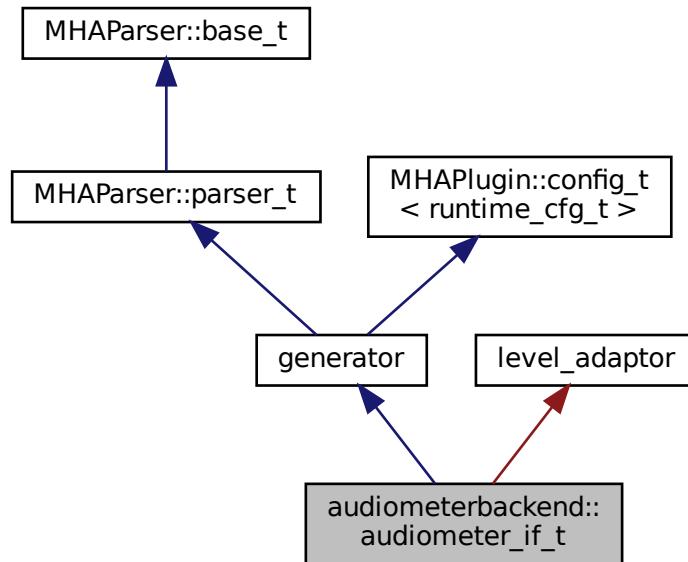
```
5.46.2.3 process() mha_wave_t* attenuate20_t::process (
    mha_wave_t * signal ) [inline]
```

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

- **attenuate20.cpp**

## 5.47 audiometerbackend::audiometer\_if\_t Class Reference

Inheritance diagram for audiometerbackend::audiometer\_if\_t:



### Public Member Functions

- `audiometer_if_t ( algo_comm_t iac, const std::string &configured_name)`
- `mha_wave_t * process ( mha_wave_t *)`
- `void prepare ( mhaconfig_t &)`

### Private Member Functions

- `void update ()`
- `void change_mode ()`
- `void set_level ()`

### Private Attributes

- `MHParse::int_t freq`
- `MHParse::kw_t sigtype`
- `MHParse::float_t level`
- `MHParse::kw_t mode`
- `MHParse::float_t ramplen`
- `MHASignal::loop_wavefragment_t::playback_mode_t pmode`
- `std::vector< int > outchannel`
- `MHAEvents::patchbay_t< audiometer_if_t > patchbay`

## Additional Inherited Members

### 5.47.1 Constructor & Destructor Documentation

**5.47.1.1 audiometer\_if\_t()** audiometerbackend::audiometer\_if\_t::audiometer\_if\_t (

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

### 5.47.2 Member Function Documentation

**5.47.2.1 process()** mha\_wave\_t \* audiometerbackend::audiometer\_if\_t::process (

```
    mha_wave_t * s )
```

**5.47.2.2 prepare()** void audiometerbackend::audiometer\_if\_t::prepare (

```
    mhaconfig_t & tf ) [virtual]
```

Implements **MHAPlugin::plugin\_t< runtime\_cfg\_t >** (p. [1149](#)).

**5.47.2.3 update()** void audiometerbackend::audiometer\_if\_t::update ( ) [private]

**5.47.2.4 change\_mode()** void audiometerbackend::audiometer\_if\_t::change\_mode ( ) [private]

**5.47.2.5 set\_level()** void audiometerbackend::audiometer\_if\_t::set\_level ( ) [private]

### 5.47.3 Member Data Documentation

**5.47.3.1 freq** `MHAParser::int_t` audiometerbackend::audiometer\_if\_t::freq [private]

**5.47.3.2 sigtype** `MHAParser::kw_t` audiometerbackend::audiometer\_if\_t::sigtype [private]

**5.47.3.3 level** `MHAParser::float_t` audiometerbackend::audiometer\_if\_t::level [private]

**5.47.3.4 mode** `MHAParser::kw_t` audiometerbackend::audiometer\_if\_t::mode [private]

**5.47.3.5 ramplen** `MHAParser::float_t` audiometerbackend::audiometer\_if\_t::ramplen [private]

**5.47.3.6 pmode** `MHASignal::loop_wavefragment_t::playback_mode_t` audiometerbackend::audiometer\_if\_t::pmode [private]

**5.47.3.7 outchannel** `std::vector<int>` audiometerbackend::audiometer\_if\_t::outchannel [private]

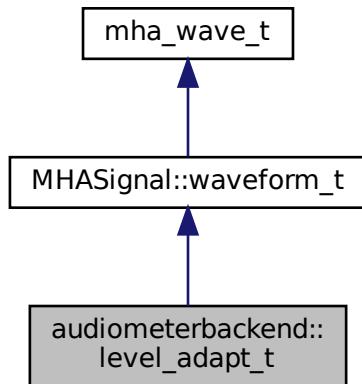
**5.47.3.8 patchbay** `MHAEVENTS::patchbay_t< audiometer_if_t> audiometerbackend::audiometer_if_t::patchbay [private]`

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

- `audiometerbackend.cpp`

## 5.48 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

### 5.48.1 Constructor & Destructor Documentation

```
5.48.1.1 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_ )
```

### 5.48.2 Member Function Documentation

```
5.48.2.1 update_frame() void audiometerbackend::level_adapt_t::update_frame ( )
```

```
5.48.2.2 get_level() mha_real_t audiometerbackend::level_adapt_t::get_level ( )
const [inline]
```

```
5.48.2.3 can_update() bool audiometerbackend::level_adapt_t::can_update ( ) const
[inline]
```

### 5.48.3 Member Data Documentation

```
5.48.3.1 ilen unsigned int audiometerbackend::level_adapt_t::ilen [private]
```

**5.48.3.2 pos** `unsigned int audiometerbackend::level_adapt_t::pos` [private]

**5.48.3.3 wnd** `MHAWindow::fun_t audiometerbackend::level_adapt_t::wnd` [private]

**5.48.3.4 l\_new** `mha_real_t audiometerbackend::level_adapt_t::l_new` [private]

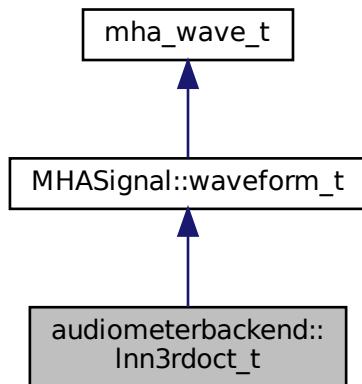
**5.48.3.5 l\_old** `mha_real_t audiometerbackend::level_adapt_t::l_old` [private]

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

- **audiometerbackend.cpp**

## 5.49 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

### 5.49.1 Constructor & Destructor Documentation

```
5.49.1.1 Inn3rdoct_t() audiometerbackend::Inn3rdoct_t::Inn3rdoct_t (
    unsigned int fs,
    unsigned int f,
    float bw,
    unsigned int niter )
```

### 5.49.2 Member Function Documentation

```
5.49.2.1 iterate_Inn() void audiometerbackend::Inn3rdoct_t::iterate_Inn ( ) [private]
```

```
5.49.2.2 bandpass() void audiometerbackend::Inn3rdoct_t::bandpass ( ) [private]
```

### 5.49.3 Member Data Documentation

**5.49.3.1 `_fmin`** `unsigned int audiometerbackend::lnn3rdoct_t::_fmin [private]`

**5.49.3.2 `_fmax`** `unsigned int audiometerbackend::lnn3rdoct_t::_fmax [private]`

**5.49.3.3 `dev`** `std::random_device audiometerbackend::lnn3rdoct_t::dev [private]`

**5.49.3.4 `rng`** `std::default_random_engine audiometerbackend::lnn3rdoct_t::rng [private]`

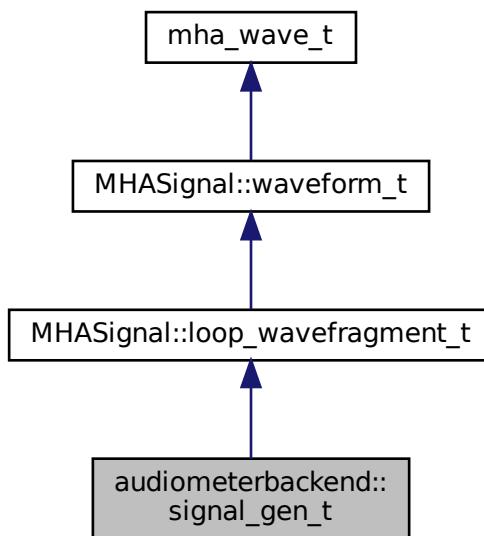
**5.49.3.5 `random`** `std::uniform_real_distribution<float> audiometerbackend::lnn3rdoct_t::random [private]`

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

- **audiometerbackend.cpp**

## 5.50 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

### 5.50.1 Constructor & Destructor Documentation

#### 5.50.1.1 signal\_gen\_t() audiometerbackend::signal\_gen\_t::signal\_gen\_t (

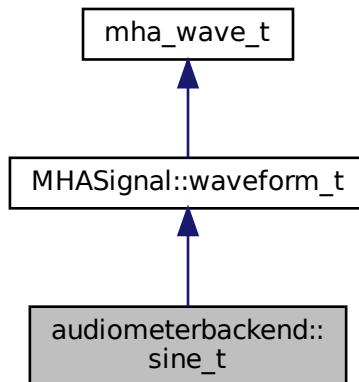
```
    int f,  
    int fs,  
    unsigned int sigtype )
```

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

- [audiometerbackend.cpp](#)

## 5.51 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

### 5.51.1 Constructor & Destructor Documentation

**5.51.1.1 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**

## 5.52 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

### 5.52.1 Detailed Description

A class to store frequency dependent data (e.g., HTL and UCL).

### 5.52.2 Member Function Documentation

**5.52.2.1 get\_frequencies()** std::vector< mha\_real\_t > AuditoryProfile::fmap\_t::get\_frequencies ( ) const

Return configured frequencies.

**5.52.2.2 get\_values()** std::vector< mha\_real\_t > AuditoryProfile::fmap\_t::get\_values ( ) const

Return stored values corresponding to the frequencies.

**5.52.2.3 isempty()** bool AuditoryProfile::fmap\_t::isempty ( ) const [inline]

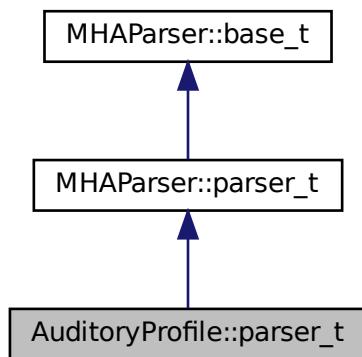
The documentation for this class was generated from the following files:

- **auditory\_profile.h**
- **auditory\_profile.cpp**

## 5.53 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

### 5.53.1 Detailed Description

Class to make the auditory profile accessible through the parser interface.

### 5.53.2 Constructor & Destructor Documentation

#### 5.53.2.1 `parser_t()` `AuditoryProfile::parser_t::parser_t( )`

### 5.53.3 Member Function Documentation

#### 5.53.3.1 `get_current_profile()` `AuditoryProfile::profile_t AuditoryProfile::parser_t::get_current_profile( )`

### 5.53.4 Member Data Documentation

**5.53.4.1 L** `AuditoryProfile::parser_t::ear_t` `AuditoryProfile::parser_t::L` [private]

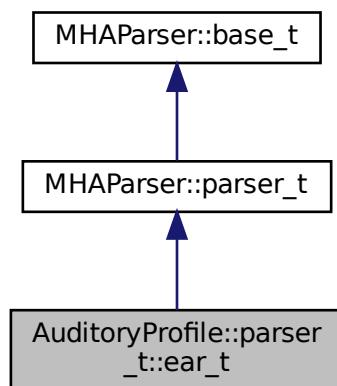
**5.53.4.2 R** `AuditoryProfile::parser_t::ear_t` `AuditoryProfile::parser_t::R` [private]

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

- `auditory_profile.h`
- `auditory_profile.cpp`

## 5.54 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** **HTML**
- **AuditoryProfile::parser\_t::fmap\_t** **UCL**

## Additional Inherited Members

### 5.54.1 Constructor & Destructor Documentation

**5.54.1.1 ear\_t()** `AuditoryProfile::parser_t::ear_t::ear_t ( )`

### 5.54.2 Member Function Documentation

**5.54.2.1 get\_ear()** `AuditoryProfile::profile_t::ear_t AuditoryProfile::parser_t<::ear_t>::get_ear ( ) const`

### 5.54.3 Member Data Documentation

**5.54.3.1 HTML** `AuditoryProfile::parser_t::fmap_t` `AuditoryProfile::parser_t::ear_t<::HTML>` [private]

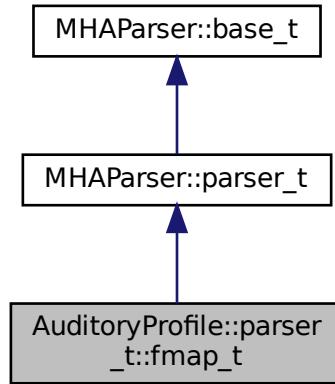
**5.54.3.2 UCL** `AuditoryProfile::parser_t::fmap_t` `AuditoryProfile::parser_t::ear_t<::UCL>` [private]

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

- **auditory\_profile.h**
- **auditory\_profile.cpp**

## 5.55 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**
- **MHParse::vfloat\_t f**
- **MHParse::vfloat\_t value**
- std::string **name\_**

### Additional Inherited Members

#### 5.55.1 Constructor & Destructor Documentation

**5.55.1.1 `fmap_t()`** `AuditoryProfile::parser_t::fmap_t::fmap_t (`  
    `const std::string & name,`  
    `const std::string & help )`

## 5.55.2 Member Function Documentation

**5.55.2.1 `get_fmap()`** `AuditoryProfile::fmap_t` `AuditoryProfile::parser_t::fmap_t::get_fmap ( ) const`

**5.55.2.2 `validate()`** `void AuditoryProfile::parser_t::fmap_t::validate ( ) [private]`

## 5.55.3 Member Data Documentation

**5.55.3.1 `patchbay`** `MHAEEvents::patchbay_t< AuditoryProfile::parser_t::fmap_t>` `AuditoryProfile::parser_t::fmap_t::patchbay [private]`

**5.55.3.2 `f`** `MHAParser::vfloat_t` `AuditoryProfile::parser_t::fmap_t::f [private]`

**5.55.3.3 `value`** `MHAParser::vfloat_t` `AuditoryProfile::parser_t::fmap_t::value [private]`

**5.55.3.4 `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`

## 5.56 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.*

### **5.56.1 Detailed Description**

## The Auditory Profile class.

See definition of auditory profile

Currently only the audiogram data is stored.

## 5.56.2 Member Function Documentation

Return ear information of channel number.

### 5.56.3 Member Data Documentation

#### 5.56.3.1 L `AuditoryProfile::profile_t::ear_t` `AuditoryProfile::profile_t::L`

Left ear data.

#### 5.56.3.2 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`

## 5.57 `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`

### 5.57.1 Detailed Description

Class for ear-dependent parameters, e.g., audiograms or unilateral loudness scaling.

## 5.57.2 Member Function Documentation

**5.57.2.1 convert\_empty2normal()** void AuditoryProfile::profile\_t::ear\_t::convert←  
\_empty2normal ( )

## 5.57.3 Member Data Documentation

**5.57.3.1 HTL** AuditoryProfile::fmap\_t AuditoryProfile::profile\_t::ear\_t::HTL

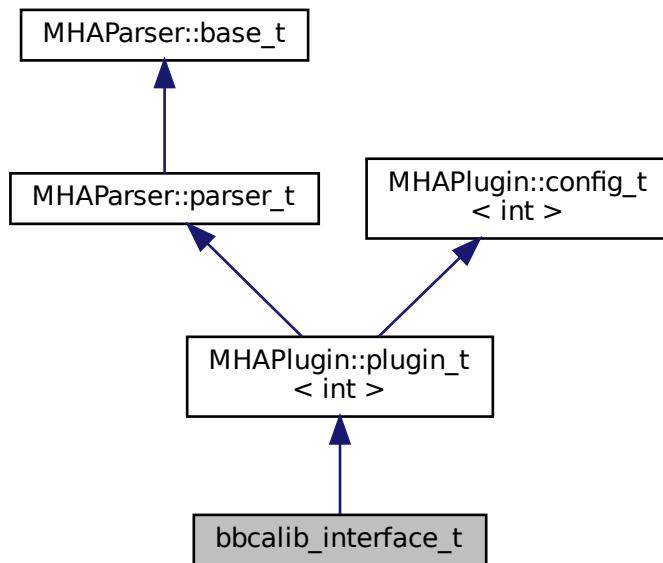
**5.57.3.2 UCL** AuditoryProfile::fmap\_t AuditoryProfile::profile\_t::ear\_t::UCL

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

- [auditory\\_profile.h](#)
- [auditory\\_profile.cpp](#)

## 5.58 bbcalib\_interface\_t Class Reference

Inheritance diagram for bbcalib\_interface\_t:



## Public Member Functions

- `bbcilib_interface_t ( algo_comm_t iac, const std::string &configured_name)`
- `~bbcilib_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::mhaplugloader_t plugloader`

## Additional Inherited Members

### 5.58.1 Constructor & Destructor Documentation

**5.58.1.1 `bbcilib_interface_t()`** `bbcilib_interface_t::bbcilib_interface_t (`  
`algo_comm_t iac,`  
`const std::string & configured_name )`

**5.58.1.2 `~bbcilib_interface_t()`** `bbcilib_interface_t::~bbcilib_interface_t ( )`

### 5.58.2 Member Function Documentation

**5.58.2.1 `process()`** `mha_wave_t * bbcilib_interface_t::process (`  
`mha_wave_t * s )`

```
5.58.2.2 prepare() void bbcalib_interface_t::prepare (
    mhaconfig_t & tf ) [virtual]
```

Implements **MHAPlugIn::plugin\_t< int >** (p. 1149).

```
5.58.2.3 release() void bbcalib_interface_t::release () [virtual]
```

Reimplemented from **MHAPlugIn::plugin\_t< int >** (p. 1150).

### 5.58.3 Member Data Documentation

```
5.58.3.1 calib_in calibrator_t bbcalib_interface_t::calib_in [private]
```

```
5.58.3.2 calib_out calibrator_t bbcalib_interface_t::calib_out [private]
```

```
5.58.3.3 plugloader MHAParser::mhapluginloader_t bbcalib_interface_t::plugloader
[private]
```

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

- **transducers.cpp**

## 5.59 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**

### 5.59.1 Constructor & Destructor Documentation

**5.59.1.1 calibrator\_runtime\_layer\_t()** `calibrator_runtime_layer_t::calibrator_runtime->`  
`_layer_t (`  
`bool is_input,`  
`const mhaconfig_t & tf,`  
`calibrator_variables_t & vars )`

### 5.59.2 Member Function Documentation

**5.59.2.1 process()** `mha_real_t calibrator_runtime_layer_t::process (`  
`mha_wave_t ** s )`

**5.59.2.2 firfirlen()** `unsigned int calibrator_runtime_layer_t::firfirlen (`  
`const std::vector< std::vector< float > > & fir ) [static], [private]`

**5.59.2.3 firfir2ffflen()** `unsigned int calibrator_runtime_layer_t::firfir2ffflen ( unsigned int fragsize, const std::vector< std::vector< float > > & fir ) [static], [private]`

### 5.59.3 Member Data Documentation

**5.59.3.1 fir** `MHAFilter::fftfilter_t calibrator_runtime_layer_t::fir [private]`

**5.59.3.2 quant** `MHASignal::quantizer_t calibrator_runtime_layer_t::quant [private]`

**5.59.3.3 gain** `MHASignal::waveform_t calibrator_runtime_layer_t::gain [private]`

**5.59.3.4 softclip** `softclipper_t calibrator_runtime_layer_t::softclip [private]`

**5.59.3.5 b\_is\_input** `bool calibrator_runtime_layer_t::b_is_input [private]`

**5.59.3.6 b\_use\_fir** `bool calibrator_runtime_layer_t::b_use_fir [private]`

**5.59.3.7 b\_use\_clipping** `bool calibrator_runtime_layer_t::b_use_clipping [private]`

**5.59.3.8 speechnoise** `MHASignal::loop_wavefragment_t calibrator_runtime_layer_t::speechnoise [private]`

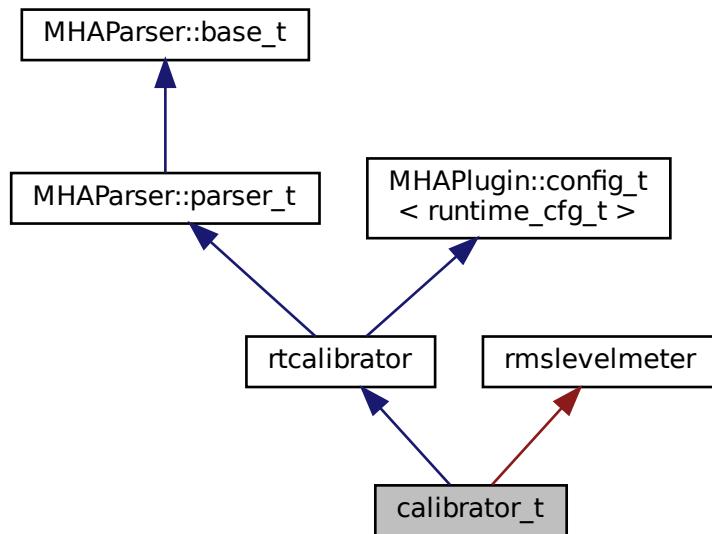
**5.59.3.9 pmode** `MHASignal::loop_wavefragment_t::playback_mode_t calibrator_runtime_layer_t::pmode [private]`

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

- `transducers.cpp`

## 5.60 calibrator\_t Class Reference

Inheritance diagram for `calibrator_t`:



### Public Member Functions

- `calibrator_t ( algo_comm_t, bool is_input)`
- `void prepare ( mhaconfig_t &tf)`
- `void release ()`
- `mha_wave_t * process ( mha_wave_t *s)`

## Private Member Functions

- void **update ()**
- void **update\_tau\_level ()**
- void **read\_levels ()**

## Private Attributes

- bool **b\_is\_input**
- **MHAEvents::patchbay\_t< calibrator\_t > patchbay**
- **calibrator\_variables\_t vars**
- bool **prepared**

## Additional Inherited Members

### 5.60.1 Constructor & Destructor Documentation

```
5.60.1.1 calibrator_t() calibrator_t::calibrator_t (
    algo_comm_t iac,
    bool is_input )
```

### 5.60.2 Member Function Documentation

```
5.60.2.1 prepare() void calibrator_t::prepare (
    mhaconfig_t & tf ) [inline], [virtual]
```

Implements **MHAPlugin::plugin\_t< runtime\_cfg\_t >** (p. [1149](#)).

```
5.60.2.2 release() void calibrator_t::release ( ) [inline], [virtual]
```

Reimplemented from **MHAPlugin::plugin\_t< runtime\_cfg\_t >** (p. [1150](#)).

**5.60.2.3 process()** `mha_wave_t * calibrator_t::process ( mha_wave_t * s )`

**5.60.2.4 update()** `void calibrator_t::update ( ) [private]`

**5.60.2.5 update\_tau\_level()** `void calibrator_t::update_tau_level ( ) [private]`

**5.60.2.6 read\_levels()** `void calibrator_t::read_levels ( ) [private]`

### 5.60.3 Member Data Documentation

**5.60.3.1 b\_is\_input** `bool calibrator_t::b_is_input [private]`

**5.60.3.2 patchbay** `MHAEvents::patchbay_t< calibrator_t > calibrator_t::patchbay [private]`

**5.60.3.3 vars** `calibrator_variables_t calibrator_t::vars [private]`

**5.60.3.4 prepared** `bool calibrator_t::prepared [private]`

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

- `transducers.cpp`

## 5.61 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`

### 5.61.1 Constructor & Destructor Documentation

```
5.61.1.1 calibrator_variables_t() calibrator_variables_t::calibrator_variables_t (  
    bool is_input,  
    MHAParser::parser_t & parent )
```

### 5.61.2 Member Data Documentation

```
5.61.2.1 peaklevel MHAParser::vfloat_t calibrator_variables_t::peaklevel
```

**5.61.2.2 fir** `MHAParser::mfloat_t` calibrator\_variables\_t::fir

**5.61.2.3 nbits** `MHAParser::int_t` calibrator\_variables\_t::nbits

**5.61.2.4 tau\_level** `MHAParser::float_t` calibrator\_variables\_t::tau\_level

**5.61.2.5 spnoise\_mode** `MHAParser::kw_t` calibrator\_variables\_t::spnoise\_mode

**5.61.2.6 spnoise\_channels** `MHAParser::vint_t` calibrator\_variables\_t::spnoise\_↔  
channels

**5.61.2.7 spnoise\_level** `MHAParser::float_t` calibrator\_variables\_t::spnoise\_level

**5.61.2.8 rmslevel** `MHAParser::vffloat_mon_t` calibrator\_variables\_t::rmslevel

**5.61.2.9 spnoise\_parser** `MHAParser::parser_t` calibrator\_variables\_t::spnoise\_↔  
parser

**5.61.2.10 srate** `MHAParser::float_mon_t` calibrator\_variables\_t::srate

**5.61.2.11 fragsize** `MHAParser::int_mon_t calibrator_variables_t::fragsize`

**5.61.2.12 num\_channels** `MHAParser::int_mon_t calibrator_variables_t::num_channels`

**5.61.2.13 config\_parser** `MHAParser::parser_t calibrator_variables_t::config_parser`

**5.61.2.14 softclip** `softclipper_variables_t calibrator_variables_t::softclip`

**5.61.2.15 do\_clipping** `MHAParser::bool_t calibrator_variables_t::do_clipping`

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

- `transducers.cpp`

## 5.62 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::o1filt_lowpass_t` `attack`
- `MHAFilter::o1filt_maxtrack_t` `decay`

## Private Attributes

- `mha_real_t gain_wave_`
- `mha_real_t gain_spec_`
- `bool replace_`
- `bool use_frozen_`
- `MHASignal::waveform_t frozen_noise_`
- `unsigned int pos`
- `std::default_random_engine rng`
- `std::uniform_real_distribution< mha_real_t > rand_dist`

### 5.62.1 Constructor & Destructor Documentation

**5.62.1.1 `cfg_t()` [1/4]** `cfg_t::cfg_t (`  
`unsigned int ichannel,`  
`unsigned int numchannels,`  
`const mha_complex_t & iscale )`

**5.62.1.2 `cfg_t()` [2/4]** `cfg_t::cfg_t (`  
`unsigned int ichannel,`  
`unsigned int numchannels )`

**5.62.1.3 `cfg_t()` [3/4]** `cfg_t::cfg_t (`  
`mhaconfig_t chcfg,`  
`mha_real_t newlev,`  
`bool replace,`  
`mha_real_t len,`  
`int seed )`

**5.62.1.4 `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 )`

## 5.62.2 Member Function Documentation

**5.62.2.1 process()** [1/2] void cfg\_t::process (

```
mha_wave_t * s ) [inline]
```

**5.62.2.2 process()** [2/2] void cfg\_t::process (

```
mha_spec_t * s ) [inline]
```

## 5.62.3 Member Data Documentation

**5.62.3.1 channel** unsigned int cfg\_t::channel

**5.62.3.2 scale** mha\_complex\_t cfg\_t::scale

**5.62.3.3 gain\_wave\_** mha\_real\_t cfg\_t::gain\_wave\_ [private]

**5.62.3.4 gain\_spec\_** mha\_real\_t cfg\_t::gain\_spec\_ [private]

**5.62.3.5 replace\_** bool cfg\_t::replace\_ [private]

**5.62.3.6 `use_frozen_`** `bool cfg_t::use_frozen_ [private]`

**5.62.3.7 `frozen_noise_`** `MHASignal::waveform_t cfg_t::frozen_noise_ [private]`

**5.62.3.8 `pos`** `unsigned int cfg_t::pos [private]`

**5.62.3.9 `rng`** `std::default_random_engine cfg_t::rng [private]`

**5.62.3.10 `rand_dist`** `std::uniform_real_distribution< mha_real_t> cfg_t::rand_dist [private]`

**5.62.3.11 `start_lin`** `mha_real_t cfg_t::start_lin`

**5.62.3.12 `alpha`** `mha_real_t cfg_t::alpha`

**5.62.3.13 `attack`** `MHAFilter::olflt_lowpass_t cfg_t::attack`

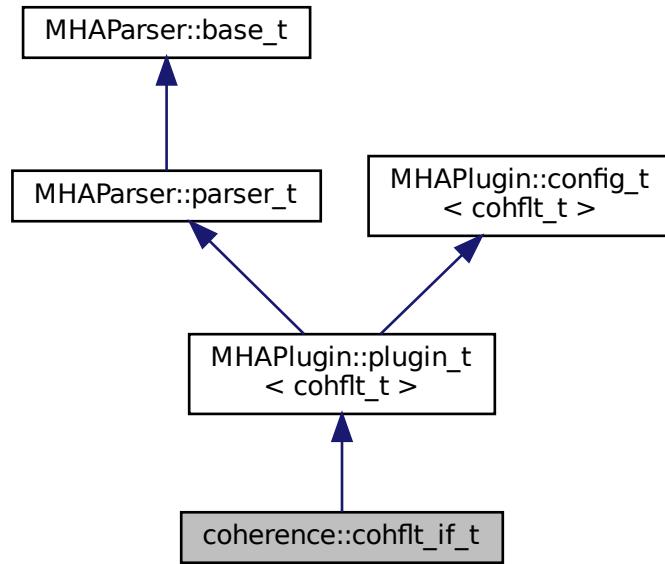
**5.62.3.14 `decay`** `MHAFilter::olflt_maxtrack_t cfg_t::decay`

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

- `complex_scale_channel.cpp`
- `example6.cpp`
- `noise.cpp`
- `softclip.cpp`

## 5.63 coherence::cohflt\_if\_t Class Reference

Inheritance diagram for coherence::cohflt\_if\_t:



### Public Member Functions

- **cohflt\_if\_t ( algo\_comm\_t iac, const std::string &configured\_name)**
- void **prepare ( mhaconfig\_t &)**
- void **release ()**
- **mha\_spec\_t \* process ( mha\_spec\_t \*)**

### Private Member Functions

- void **update ()**

### Private Attributes

- **MHAEvents::patchbay\_t< cohflt\_if\_t > patchbay**
- **vars\_t vars**
- **const std::string algo**

## Additional Inherited Members

### 5.63.1 Constructor & Destructor Documentation

```
5.63.1.1 cohflt_if_t() coherence::cohflt_if_t::cohflt_if_t (
    algo_comm_t iac,
    const std::string & configured_name )
```

### 5.63.2 Member Function Documentation

```
5.63.2.1 prepare() void coherence::cohflt_if_t::prepare (
    mhaconfig_t & tf ) [virtual]
```

Implements **MHAPlugin::plugin\_t< cohflt\_t >** (p. [1149](#)).

```
5.63.2.2 release() void coherence::cohflt_if_t::release () [virtual]
```

Reimplemented from **MHAPlugin::plugin\_t< cohflt\_t >** (p. [1150](#)).

```
5.63.2.3 process() mha_spec_t * coherence::cohflt_if_t::process (
    mha_spec_t * s )
```

```
5.63.2.4 update() void coherence::cohflt_if_t::update () [private]
```

### 5.63.3 Member Data Documentation

**5.63.3.1 patchbay** MHAEvents::patchbay\_t< cohflt\_if\_t> coherence::cohflt\_if\_t::patchbay [private]

**5.63.3.2 vars** vars\_t coherence::cohflt\_if\_t::vars [private]

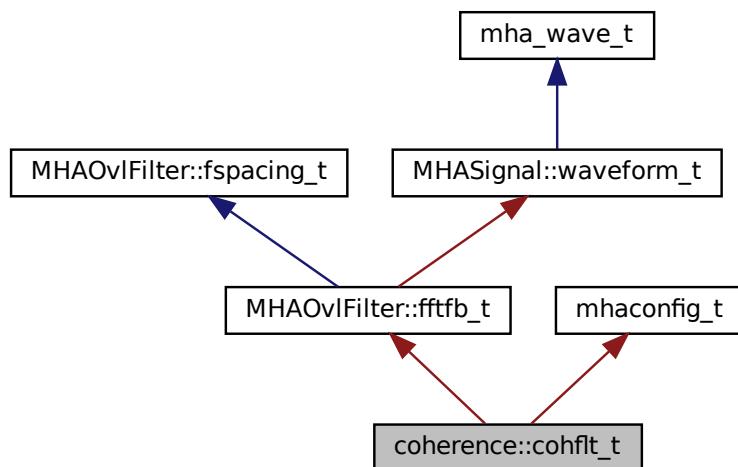
**5.63.3.3 algo** const std::string coherence::cohflt\_if\_t::algo [private]

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

- **coherence.cpp**

## 5.64 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

### 5.64.1 Constructor & Destructor Documentation

```
5.64.1.1 cohflt_t() coherence::cohflt_t::cohflt_t (
    vars_t & v,
    const mhaconfig_t & icf,
    algo_comm_t iac,
    const std::string & name )
```

### 5.64.2 Member Function Documentation

```
5.64.2.1 process() mha_spec_t * coherence::cohflt_t::process (
    mha_spec_t * s )
```

**5.64.2.2 insert()** void coherence::cohflt\_t::insert ( )

### 5.64.3 Member Data Documentation

**5.64.3.1 nbands** unsigned int coherence::cohflt\_t::nbands [private]

**5.64.3.2 avg\_ipd** bool coherence::cohflt\_t::avg\_ipd [private]

**5.64.3.3 cg** mha\_complex\_t coherence::cohflt\_t::cg [private]

**5.64.3.4 g** float coherence::cohflt\_t::g [private]

**5.64.3.5 c\_scale** float coherence::cohflt\_t::c\_scale [private]

**5.64.3.6 c\_min** float coherence::cohflt\_t::c\_min [private]

**5.64.3.7 alpha** MHASignal::waveform\_t coherence::cohflt\_t::alpha [private]

**5.64.3.8 limit** float coherence::cohflt\_t::limit [private]

**5.64.3.9 lp1r** `MHAFilter::olflt_lowpass_t` coherence::cohflt\_t::lp1r [private]

**5.64.3.10 lp1i** `MHAFilter::olflt_lowpass_t` coherence::cohflt\_t::lp1i [private]

**5.64.3.11 coh\_c** `MHA_AC::spectrum_t` coherence::cohflt\_t::coh\_c [private]

**5.64.3.12 coh\_rlp** `MHA_AC::waveform_t` coherence::cohflt\_t::coh\_rlp [private]

**5.64.3.13 gain** `MHASignal::waveform_t` coherence::cohflt\_t::gain [private]

**5.64.3.14 gain\_delay** `MHASignal::delay_wave_t` coherence::cohflt\_t::gain\_delay [private]

**5.64.3.15 s\_out** `MHASignal::spectrum_t` coherence::cohflt\_t::s\_out [private]

**5.64.3.16 bInvert** bool coherence::cohflt\_t::bInvert [private]

**5.64.3.17 lp1ltg** `MHAFilter::olflt_lowpass_t` coherence::cohflt\_t::lp1ltg [private]

**5.64.3.18 b\_ltg** bool coherence::cohflt\_t::b\_ltg [private]

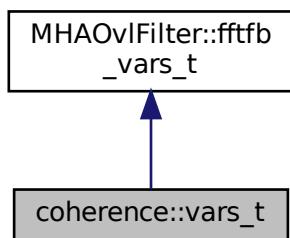
**5.64.3.19 staticgain** std::vector<float> coherence::cohflt\_t::staticgain [private]

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

- **coherence.cpp**

## 5.65 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**

## 5.65.1 Constructor & Destructor Documentation

**5.65.1.1 vars\_t()** coherence::vars\_t::vars\_t (

```
MHAParser::parser_t * p )
```

## 5.65.2 Member Data Documentation

**5.65.2.1 tau\_unit** MHAParser::kw\_t coherence::vars\_t::tau\_unit

**5.65.2.2 tau** MHAParser::vfloat\_t coherence::vars\_t::tau

**5.65.2.3 alpha** MHAParser::vfloat\_t coherence::vars\_t::alpha

**5.65.2.4 limit** MHAParser::float\_t coherence::vars\_t::limit

**5.65.2.5 mapping** MHAParser::vfloat\_t coherence::vars\_t::mapping

**5.65.2.6 average** MHAParser::kw\_t coherence::vars\_t::average

**5.65.2.7 invert** `MHAParser::bool_t coherence::vars_t::invert`

**5.65.2.8 ltgcomp** `MHAParser::bool_t coherence::vars_t::ltgcomp`

**5.65.2.9 ltgtau** `MHAParser::vfloat_t coherence::vars_t::ltgtau`

**5.65.2.10 staticgain** `MHAParser::vfloat_t coherence::vars_t::staticgain`

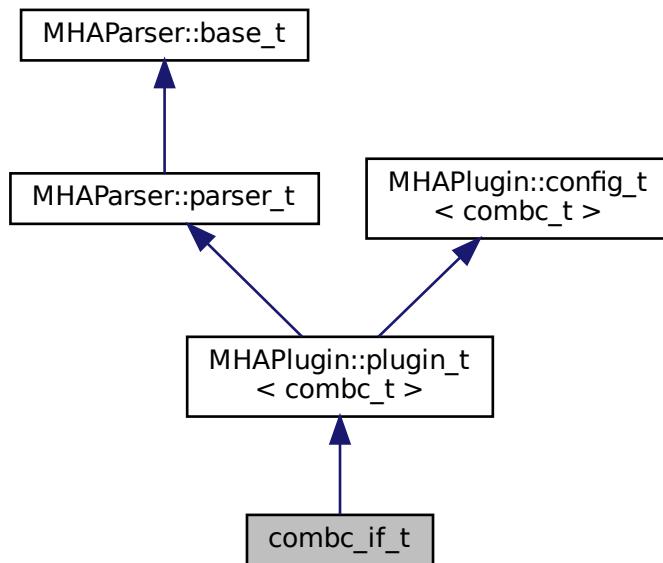
**5.65.2.11 delay** `MHAParser::int_t coherence::vars_t::delay`

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

- `coherence.cpp`

## 5.66 combc\_if\_t Class Reference

Inheritance diagram for combc\_if\_t:



## Public Member Functions

- `combc_if_t ( algo_comm_t iac, const std::string &configured_name)`
- `void prepare ( mhaconfig_t &)`
- `mha_wave_t * process ( mha_wave_t *)`
- `mha_spec_t * process ( mha_spec_t *)`

## Private Attributes

- `MHAParser::int_t outchannels`
- `MHAParser::bool_t interleaved`
- `MHAParser::string_t channel_gain_name`
- `MHAParser::string_t element_gain_name`

## Additional Inherited Members

### 5.66.1 Constructor & Destructor Documentation

**5.66.1.1 `combc_if_t()`** `combc_if_t::combc_if_t (`  
`algo_comm_t iac,`  
`const std::string & configured_name )`

### 5.66.2 Member Function Documentation

**5.66.2.1 `prepare()`** `void combc_if_t::prepare (`  
`mhaconfig_t & chcfg ) [virtual]`

Implements `MHAPlugin::plugin_t<combc_t>` (p. [1149](#)).

**5.66.2.2 `process() [1/2]`** `mha_wave_t * combc_if_t::process (`  
`mha_wave_t * s )`

**5.66.2.3 process() [2/2]** `mha_spec_t * combc_if_t::process (`  
`mha_spec_t * s )`

### 5.66.3 Member Data Documentation

**5.66.3.1 outchannels** `MHAParser::int_t combc_if_t::outchannels [private]`

**5.66.3.2 interleaved** `MHAParser::bool_t combc_if_t::interleaved [private]`

**5.66.3.3 channel\_gain\_name** `MHAParser::string_t combc_if_t::channel_gain_name [private]`

**5.66.3.4 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**

## 5.67 **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_`

### 5.67.1 Constructor & Destructor Documentation

#### 5.67.1.1 `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 )
```

### 5.67.2 Member Function Documentation

#### 5.67.2.1 `process() [1/2]` `mha_wave_t * combc_t::process (`

```
    mha_wave_t * s )
```

#### 5.67.2.2 `process() [2/2]` `mha_spec_t * combc_t::process (`

```
    mha_spec_t * s )
```

### 5.67.3 Member Data Documentation

**5.67.3.1 ac\_ algo\_comm\_t** combc\_t::ac\_ [private]

**5.67.3.2 interleaved\_** bool combc\_t::interleaved\_ [private]

**5.67.3.3 nbands** unsigned int combc\_t::nbands [private]

**5.67.3.4 w\_out** MHASignal::waveform\_t combc\_t::w\_out [private]

**5.67.3.5 s\_out** MHASignal::spectrum\_t combc\_t::s\_out [private]

**5.67.3.6 channel\_gains\_** std::vector< mha\_real\_t> combc\_t::channel\_gains\_ [private]

**5.67.3.7 element\_gain\_name\_** std::string combc\_t::element\_gain\_name\_ [private]

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

- **combinechannels.cpp**

## 5.68 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.*

**5.68.1 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.

**5.68.2 Member Data Documentation****5.68.2.1 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`

**5.68.2.2 num\_entries** comm\_var\_t::num\_entries

Number of entries.

**5.68.2.3 stride** comm\_var\_t::stride

length of one row (C interpretation) or of one column (Fortran interpretation)

**5.68.2.4 data** comm\_var\_t::data

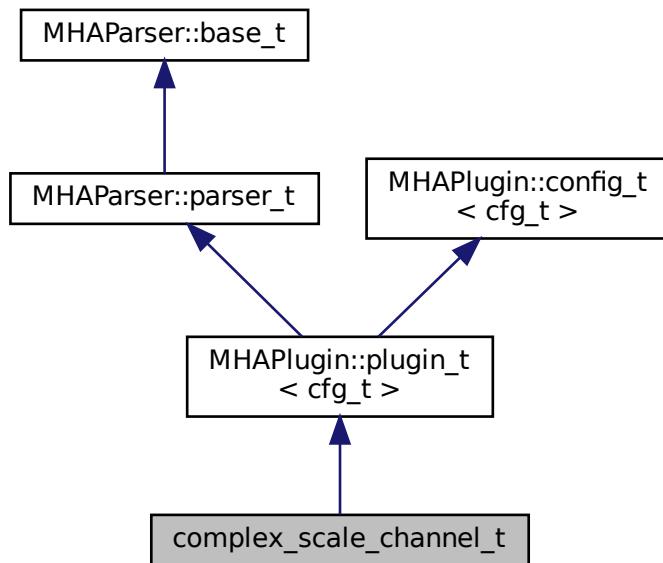
Pointer to variable data.

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

- [mha.hh](#)
- [mha\\_algo\\_comm.cpp](#)

**5.69 complex\_scale\_channel\_t Class Reference**

Inheritance diagram for complex\_scale\_channel\_t:



## Public Member Functions

- `complex_scale_channel_t ( algo_comm_t iac, const std::string &configured_name)`
- `mha_spec_t * process ( mha_spec_t *)`
- `void prepare ( mhaconfig_t &)`

## Private Member Functions

- `void update_cfg ()`

## Private Attributes

- `MHAEvents::patchbay_t< complex_scale_channel_t > patchbay`
- `MHAParser::int_t scale_ch`
- `MHAParser::complex_t factor`

## Additional Inherited Members

### 5.69.1 Constructor & Destructor Documentation

**5.69.1.1 `complex_scale_channel_t()`** `complex_scale_channel_t::complex_scale_channel_t (`  
`algo_comm_t iac,`  
`const std::string & configured_name )`

### 5.69.2 Member Function Documentation

**5.69.2.1 `process()`** `mha_spec_t * complex_scale_channel_t::process (`  
`mha_spec_t * spec )`

```
5.69.2.2 prepare() void complex_scale_channel_t::prepare (  
    mhaconfig_t & cfg ) [virtual]
```

Implements **MHAPlugIn::plugin\_t< cfg\_t >** (p. 1149).

```
5.69.2.3 update_cfg() void complex_scale_channel_t::update_cfg ( ) [private]
```

### 5.69.3 Member Data Documentation

```
5.69.3.1 patchbay MHAEvents::patchbay_t< complex_scale_channel_t > complex_scale←  
_channel_t::patchbay [private]
```

```
5.69.3.2 scale_ch MHAParser::int_t complex_scale_channel_t::scale_ch [private]
```

```
5.69.3.3 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**

## 5.70 cpuload::cpuload\_cfg\_t Class Reference

### Public Member Functions

- **cpuload\_cfg\_t ( mha\_real\_t factor\_, size\_t table\_size\_, bool use\_sine\_ )**  
*Ctor of the runtime configuration class.*
- **void process (unsigned fac\_)**  
*Process callback. Does not actually change signal.*

## Private Member Functions

- void **calc\_sine** ()
- void **write\_to\_table** ()

## Private Attributes

- float **phase** =0  
*Phase of the sine.*
- volatile float **result** =0  
*Result of sin(phase). Volatile to prevent compiler from optimizing away the calculation.*
- bool **use\_sine** =false  
*Use sine or do table operation.*
- float **factor** =0  
*cpu load factor. Values > 1 increase cpu load, values < 1 decrease it*
- std::vector< float > **table**  
*Table with arbitrary values to operate on. Unused if use\_sine=true.*

## 5.70.1 Constructor & Destructor Documentation

```
5.70.1.1 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>	

## 5.70.2 Member Function Documentation

**5.70.2.1 process()** void cpupload::cpupload\_cfg\_t::process ( unsigned *fac\_* )

Process callback. Does not actually change signal.

**5.70.2.2 calc\_sine()** void cpupload::cpupload\_cfg\_t::calc\_sine ( ) [private]

**5.70.2.3 write\_to\_table()** void cpupload::cpupload\_cfg\_t::write\_to\_table ( ) [private]

### 5.70.3 Member Data Documentation

**5.70.3.1 phase** float cpupload::cpupload\_cfg\_t::phase =0 [private]

Phase of the sine.

**5.70.3.2 result** volatile float cpupload::cpupload\_cfg\_t::result =0 [private]

Result of sin(phase). Volatile to prevent compiler from optimizing away the calculation.

**5.70.3.3 use\_sine** bool cpupload::cpupload\_cfg\_t::use\_sine =false [private]

Use sine or do table operation.

**5.70.3.4 factor** float cpupload::cpupload\_cfg\_t::factor =0 [private]

cpu load factor. Values > 1 increase cpu load, values < 1 decrease it

### 5.70.3.5 **table** std::vector<float> cpupload::cpupload\_cfg\_t::table [private]

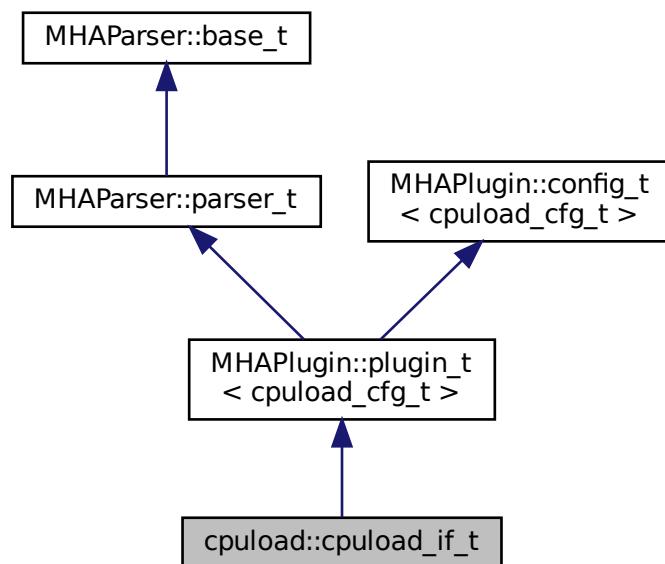
Table with arbitrary values to operate on. Unused if use\_sine=true.

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

- **cpupload.cpp**

## 5.71 cpupload::cpupload\_if\_t Class Reference

Inheritance diagram for cpupload::cpupload\_if\_t:



### Public Member Functions

- **cpupload\_if\_t ( algo\_comm\_t iac, const std::string &configured\_name)**
- **mha\_spec\_t \* process ( mha\_spec\_t \*)**
- **mha\_wave\_t \* process ( mha\_wave\_t \*)**
- **void prepare ( mhaconfig\_t &)**

### Private Member Functions

- **void update ()**

## Private Attributes

- MHAParser::float\_t factor
- MHAParser::int\_t table\_size
- MHAParser::bool\_t use\_sine
- MHAEvents::patchbay\_t< cpupload\_if\_t > patchbay

## Additional Inherited Members

### 5.71.1 Constructor & Destructor Documentation

**5.71.1.1 cpupload\_if\_t()** cpupload::cpupload\_if\_t::cpupload\_if\_t (

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

### 5.71.2 Member Function Documentation

**5.71.2.1 process() [1/2]** mha\_spec\_t \* cpupload::cpupload\_if\_t::process (

```
mha_spec_t * s )
```

**5.71.2.2 process() [2/2]** mha\_wave\_t \* cpupload::cpupload\_if\_t::process (

```
mha_wave_t * s )
```

**5.71.2.3 prepare()** void cpupload::cpupload\_if\_t::prepare (

```
mhaconfig_t & ) [virtual]
```

Implements **MHAPlugin::plugin\_t< cpupload\_cfg\_t >** (p. [1149](#)).

**5.71.2.4 update()** void cpupload::cpupload\_if\_t::update ( ) [private]

### 5.71.3 Member Data Documentation

**5.71.3.1 factor** **MHAParser::float\_t** cpupload::cpupload\_if\_t::factor [private]

**5.71.3.2 table\_size** **MHAParser::int\_t** cpupload::cpupload\_if\_t::table\_size [private]

**5.71.3.3 use\_sine** **MHAParser::bool\_t** cpupload::cpupload\_if\_t::use\_sine [private]

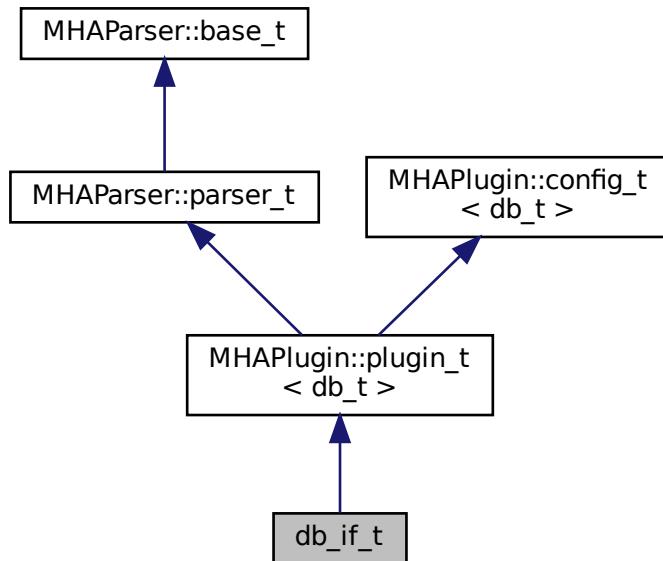
**5.71.3.4 patchbay** **MHAEvents::patchbay\_t< cpupload\_if\_t>** cpupload::cpupload\_if\_t::patchbay [private]

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

- **cpupload.cpp**

## 5.72 db\_if\_t Class Reference

Inheritance diagram for db\_if\_t:



### Public Member Functions

- `db_if_t ( algo_comm_t iac, const std::string &configured_name)`
- `mha_wave_t * process ( mha_wave_t *)`
- `void prepare ( mhaconfig_t &)`
- `void release ()`
- `~db_if_t ()`

### Private Attributes

- `MHAEvents::patchbay_t< db_if_t > patchbay`
- `MHAParser::int_t fragsize`
- `MHAParser::mhapluginloader_t plugloader`
- `std::string algo`
- `bool bypass`

## Additional Inherited Members

### 5.72.1 Constructor & Destructor Documentation

**5.72.1.1 `db_if_t()`** `db_if_t::db_if_t (`  
    `algo_comm_t iac,`  
    `const std::string & configured_name )`

**5.72.1.2 `~db_if_t()`** `db_if_t::~db_if_t ( )`

### 5.72.2 Member Function Documentation

**5.72.2.1 `process()`** `mha_wave_t * db_if_t::process (`  
    `mha_wave_t * s )`

**5.72.2.2 `prepare()`** `void db_if_t::prepare (`  
    `mhaconfig_t & conf ) [virtual]`

Implements `MHAPlugin::plugin_t< db_t >` (p. 1149).

**5.72.2.3 `release()`** `void db_if_t::release ( ) [virtual]`

Reimplemented from `MHAPlugin::plugin_t< db_t >` (p. 1150).

### 5.72.3 Member Data Documentation

**5.72.3.1 patchbay** `MHAEVENTS::patchbay_t< db_if_t > db_if_t::patchbay [private]`

**5.72.3.2 fragsize** `MHAPARSER::int_t db_if_t::fragsize [private]`

**5.72.3.3 plugloader** `MHAPARSER::mhaplugloader_t db_if_t::plugloader [private]`

**5.72.3.4 algo** `std::string db_if_t::algo [private]`

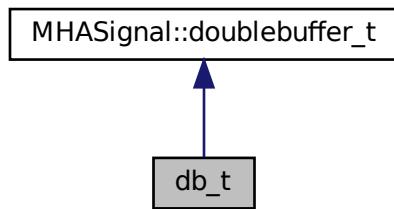
**5.72.3.5 bypass** `bool db_if_t::bypass [private]`

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

- `db.cpp`

## 5.73 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

### 5.73.1 Constructor & Destructor Documentation

```
5.73.1.1 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 )
```

### 5.73.2 Member Function Documentation

```
5.73.2.1 inner_process() mha_wave_t * db_t::inner_process (
    mha_wave_t * s ) [virtual]
```

Implements **MHASignal::doublebuffer\_t** (p. 1205).

### 5.73.3 Member Data Documentation

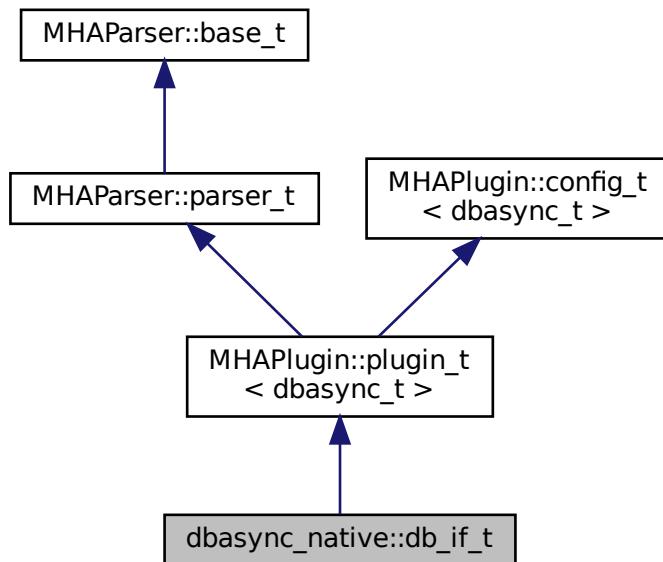
### 5.73.3.1 plugloader MHPARSER::MHAPLUGINLOADER\_T& db\_t::plugloader [private]

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

- db.cpp

## 5.74 dbasync\_native::db\_if\_t Class Reference

Inheritance diagram for dbasync\_native::db\_if\_t:



### Public Member Functions

- `db_if_t ( algo_comm_t iac, const std::string &configured_name )`
- `mha_wave_t * process ( mha_wave_t * )`
- `void prepare ( mhaconfig_t & )`
- `void release ()`
- `~db_if_t ()=default`

## Private Attributes

- **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 **algo**

## Additional Inherited Members

### 5.74.1 Constructor & Destructor Documentation

**5.74.1.1 db\_if\_t()** db\_if\_t::db\_if\_t (

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

**5.74.1.2 ~db\_if\_t()** dbasync\_native::db\_if\_t::~db\_if\_t ( ) [default]

### 5.74.2 Member Function Documentation

**5.74.2.1 process()** mha\_wave\_t \* db\_if\_t::process (

```
mha_wave_t * s )
```

```
5.74.2.2 prepare() void db_if_t::prepare (
    mhaconfig_t & conf ) [virtual]
```

Implements **MHAPlugIn::plugin\_t< dbasync\_t >** (p. 1149).

```
5.74.2.3 release() void db_if_t::release () [virtual]
```

Reimplemented from **MHAPlugIn::plugin\_t< dbasync\_t >** (p. 1150).

### 5.74.3 Member Data Documentation

```
5.74.3.1 sub_ac MHAKernel::algo_comm_class_t dbasync_native::db_if_t::sub_ac
[private]
```

```
5.74.3.2 plugloader MHAParser::mhapluginloader_t dbasync_native::db_if_t::plugloader
[private]
```

```
5.74.3.3 fragsize MHAParser::int_t dbasync_native::db_if_t::fragsize [private]
```

```
5.74.3.4 delay MHAParser::int_t dbasync_native::db_if_t::delay [private]
```

```
5.74.3.5 worker_thread_scheduler MHAParser::kw_t dbasync_native::db_if_t::worker←
_thread_scheduler [private]
```

Scheduler used for worker thread.

**5.74.3.6 worker\_thread\_priority** `MHAParser::int_t dbasync_native::db_if_t::worker_thread_priority [private]`

Priority of worker thread.

**5.74.3.7 framework\_thread\_scheduler** `MHAParser::string_mon_t dbasync_native::db_if_t::framework_thread_scheduler [private]`

Scheduler of the signal processing thread.

**5.74.3.8 framework\_thread\_priority** `MHAParser::int_mon_t dbasync_native::db_if_t::framework_thread_priority [private]`

Priority of signal processing thread.

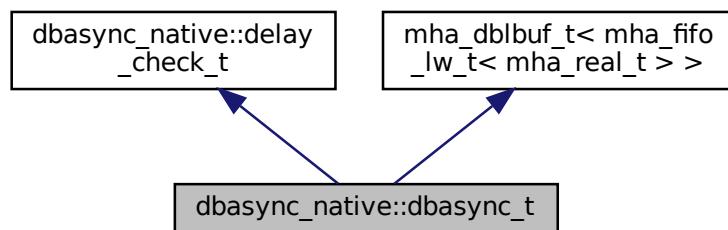
**5.74.3.9 algo** `std::string dbasync_native::db_if_t::algo [private]`

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

- `dbasync.cpp`

## 5.75 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

### 5.75.1 Constructor & Destructor Documentation

**5.75.1.1 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 )
```

**5.75.1.2 ~dbasync\_t()** dbasync\_native::dbasync\_t::~dbasync\_t ( )

### 5.75.2 Member Function Documentation

**5.75.2.1 outer\_process()** `mha_wave_t * dbasync_native::dbasync_t::outer_process ( mha_wave_t * outer_input )`

**5.75.2.2 svc()** `int dbasync_native::dbasync_t::svc ( )`

### 5.75.3 Member Data Documentation

**5.75.3.1 plugloader** `MHAParser::mhaplugloader_t& dbasync_native::dbasync_t::plugloader [private]`

**5.75.3.2 inner\_input** `MHASignal::waveform_t dbasync_native::dbasync_t::inner_input [private]`

**5.75.3.3 outer\_output** `MHASignal::waveform_t dbasync_native::dbasync_t::outer_output [private]`

**5.75.3.4 attr** `pthread_attr_t dbasync_native::dbasync_t::attr [private]`

**5.75.3.5 priority** `struct sched_param dbasync_native::dbasync_t::priority [private]`

**5.75.3.6 scheduler** `int dbasync_native::dbasync_t::scheduler [private]`

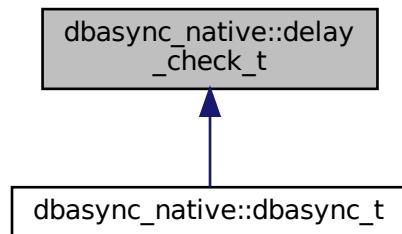
### 5.75.3.7 thread pthread\_t dbasync\_native::dbasync\_t::thread [private]

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

- **dbasync.cpp**

## 5.76 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)

### 5.76.1 Constructor & Destructor Documentation

#### 5.76.1.1 delay\_check\_t() dbasync\_native::delay\_check\_t::delay\_check\_t (

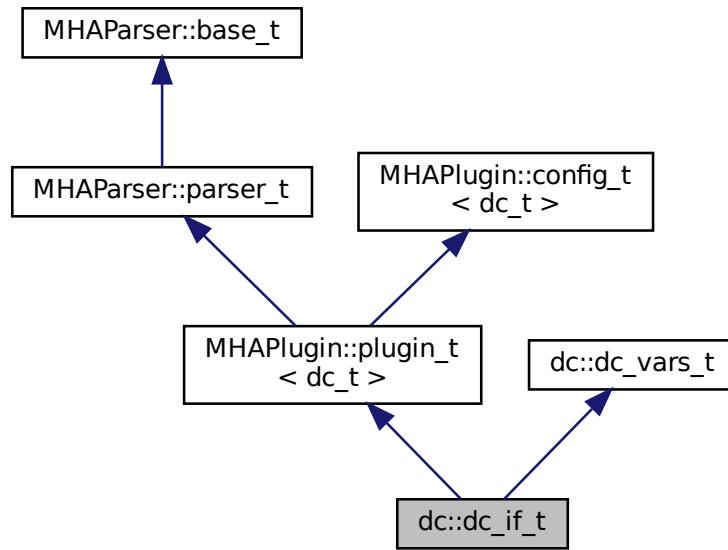
```
int delay,
unsigned inner_fragsize,
unsigned outer_fragsize ) [protected]
```

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

- **dbasync.cpp**

## 5.77 dc::dc\_if\_t Class Reference

Inheritance diagram for dc::dc\_if\_t:



### Public Member Functions

- `dc_if_t ( algo_comm_t iac, const std::string &configured_name)`
- `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

### 5.77.1 Constructor & Destructor Documentation

```
5.77.1.1 dc_if_t() dc_if_t::dc_if_t (
    algo_comm_t iac,
    const std::string & configured_name )
```

### 5.77.2 Member Function Documentation

```
5.77.2.1 prepare() void dc_if_t::prepare (
    mhaconfig_t & tf ) [virtual]
```

Implements **MHAPlugin::plugin\_t< dc\_t >** (p. 1149).

```
5.77.2.2 process() [1/2] mha_wave_t * dc_if_t::process (
    mha_wave_t * s_in )
```

```
5.77.2.3 process() [2/2] mha_spec_t * dc_if_t::process (
    mha_spec_t * s_in )
```

```
5.77.2.4 update_monitors() void dc_if_t::update_monitors ( ) [private]
```

Called from within the processing routines: updates the monitor variables.

**5.77.2.5 update()** void dc\_if\_t::update ( ) [private]

Called by MHA configuration change event mechanism: creates new runtime configuration.

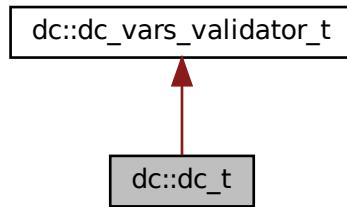
**5.77.3 Member Data Documentation****5.77.3.1 algo** std::string dc::dc\_if\_t::algo [private]**5.77.3.2 patchbay** MHAEvents::patchbay\_t< dc\_if\_t> dc::dc\_if\_t::patchbay [private]

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

- **dc.hh**
- **dc.cpp**

**5.78 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

### 5.78.1 Constructor & Destructor Documentation

**5.78.1.1 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 = {} )
```

## 5.78.2 Member Function Documentation

**5.78.2.1 process() [1/2]** `mha_wave_t * dc_t::process (`

```

    mha_wave_t * s )
```

**5.78.2.2 process() [2/2]** `mha_spec_t * dc_t::process (`

```

    mha_spec_t * s )
```

**5.78.2.3 explicit\_insert()** `void dc_t::explicit_insert ( )`

**5.78.2.4 get\_nbands()** `unsigned dc::dc_t::get_nbands ( ) const [inline]`

Number of frequency bands accessor.

**5.78.2.5 get\_nch()** `unsigned dc::dc_t::get_nch ( ) const [inline]`

Total number of channels accessor.

**5.78.2.6 get\_level\_in\_db()** const MHASignal::waveform\_t& dc::dc\_t::get\_level\_in\_db  
( ) const [inline]

**5.78.2.7 get\_level\_in\_db\_adjusted()** const MHASignal::waveform\_t& dc::dc\_t::get\_level\_in\_db\_adjusted ( ) const [inline]

**5.78.2.8 get\_rmslevel\_filter\_state()** std::vector< mha\_real\_t> dc::dc\_t::get\_rmslevel\_filter\_state ( ) const [inline]

**5.78.2.9 get\_attack\_filter\_state()** std::vector< mha\_real\_t> dc::dc\_t::get\_attack\_filter\_state ( ) const [inline]

**5.78.2.10 get\_decay\_filter\_state()** std::vector< mha\_real\_t> dc::dc\_t::get\_decay\_filter\_state ( ) const [inline]

### 5.78.3 Member Data Documentation

**5.78.3.1 gt** std::vector< MHATableLookup::linear\_table\_t> dc::dc\_t::gt [private]

**5.78.3.2 offset** std::vector< mha\_real\_t> dc::dc\_t::offset [private]

**5.78.3.3 rmslevel** MHAFilter::olflt\_lowpass\_t dc::dc\_t::rmslevel [private]

**5.78.3.4 attack** `MHAFilter::olflt_lowpass_t` `dc::dc_t::attack` [private]

**5.78.3.5 decay** `MHAFilter::olflt_maxtrack_t` `dc::dc_t::decay` [private]

**5.78.3.6 bypass** `bool` `dc::dc_t::bypass` [private]

**5.78.3.7 log\_interp** `bool` `dc::dc_t::log_interp` [private]

**5.78.3.8 naudiochannels** `unsigned int` `dc::dc_t::naudiochannels` [private]

**5.78.3.9 nbands** `unsigned int` `dc::dc_t::nbands` [private]

**5.78.3.10 nch** `unsigned int` `dc::dc_t::nch` [private]

**5.78.3.11 level\_in\_db** `MHA_AC::waveform_t` `dc::dc_t::level_in_db` [private]

**5.78.3.12 level\_in\_db\_adjusted** `MHA_AC::waveform_t` `dc::dc_t::level_in_db_adjusted` [private]

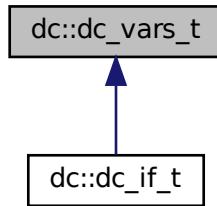
**5.78.3.13 fftlen** unsigned int dc::dc\_t::fftlen [private]

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

- dc.hh
- dc.cpp

## 5.79 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

- `MHAParser::bool_t log_interp`
- `MHAParser::string_t clientid`
- `MHAParser::string_t gainrule`
- `MHAParser::string_t preset`
- `MHAParser::int_mon_t modified`
- `MHAParser::vfloat_mon_t input_level`
- `MHAParser::vfloat_mon_t filtered_level`
- `MHAParser::vfloat_mon_t center_frequencies`
- `MHAParser::vfloat_mon_t edge_frequencies`
- `MHAParser::vfloat_mon_t band_weights`

## 5.79.1 Constructor & Destructor Documentation

**5.79.1.1 `dc_vars_t()`** `dc_vars_t::dc_vars_t (`  
`MHAParser::parser_t & p )` [explicit]

## 5.79.2 Member Data Documentation

**5.79.2.1 `gtdata`** `MHAParser::mfloat_t dc::dc_vars_t::gtdata`

**5.79.2.2 `gtmin`** `MHAParser::vfloat_t dc::dc_vars_t::gtmin`

**5.79.2.3 `gtstep`** `MHAParser::vfloat_t dc::dc_vars_t::gtstep`

**5.79.2.4 `taurmslevel`** `MHAParser::vfloat_t dc::dc_vars_t::taurmslevel`

**5.79.2.5 tauattack** `MHAParser::vfloat_t dc::dc_vars_t::tauattack`

**5.79.2.6 taudecay** `MHAParser::vfloat_t dc::dc_vars_t::taudecay`

**5.79.2.7 offset** `MHAParser::vfloat_t dc::dc_vars_t::offset`

**5.79.2.8 filterbank** `MHAParser::string_t dc::dc_vars_t::filterbank`

**5.79.2.9 cf\_name** `std::string dc::dc_vars_t::cf_name`

**5.79.2.10 ef\_name** `std::string dc::dc_vars_t::ef_name`

**5.79.2.11 bw\_name** `std::string dc::dc_vars_t::bw_name`

**5.79.2.12 chname** `MHAParser::string_t dc::dc_vars_t::chname`

**5.79.2.13 bypass** `MHAParser::bool_t dc::dc_vars_t::bypass`

**5.79.2.14 log\_interp** `MHAParser::bool_t dc::dc_vars_t::log_interp`

**5.79.2.15 clientid** `MHAParser::string_t dc::dc_vars_t::clientid`

**5.79.2.16 gainrule** `MHAParser::string_t dc::dc_vars_t::gainrule`

**5.79.2.17 preset** `MHAParser::string_t dc::dc_vars_t::preset`

**5.79.2.18 modified** `MHAParser::int_mon_t dc::dc_vars_t::modified`

**5.79.2.19 input\_level** `MHAParser::vfloat_mon_t dc::dc_vars_t::input_level`

**5.79.2.20 filtered\_level** `MHAParser::vfloat_mon_t dc::dc_vars_t::filtered_level`

**5.79.2.21 center\_frequencies** `MHAParser::vfloat_mon_t dc::dc_vars_t::center_frequencies`

**5.79.2.22 edge\_frequencies** `MHAParser::vfloat_mon_t dc::dc_vars_t::edge_frequencies`

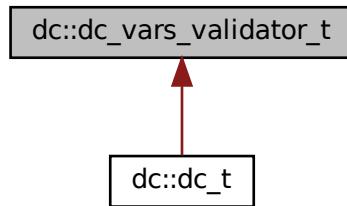
**5.79.2.23 band\_weights    `MHAParser::vfloat_mon_t dc::dc_vars_t::band_weights`**

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

- `dc.hh`
- `dc.cpp`

**5.80 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)`

**5.80.1 Constructor & Destructor Documentation****5.80.1.1 dc\_vars\_validator\_t()** `dc_vars_validator_t::dc_vars_validator_t (`  
`dc_vars_t & v,`  
`unsigned int s,`  
`mha_domain_t domain )`

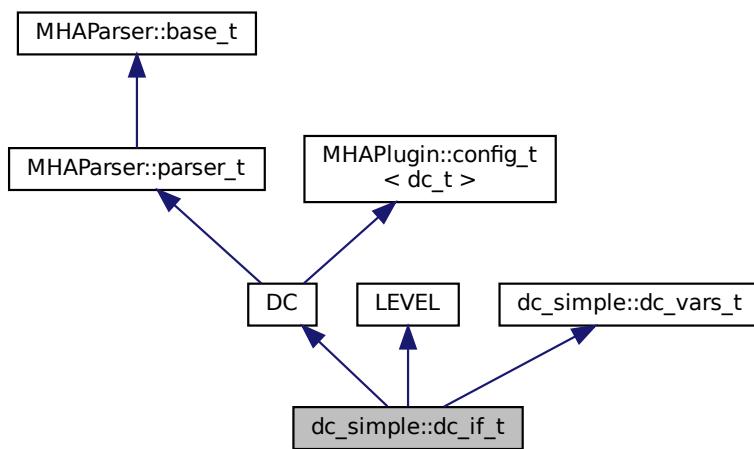
The documentation for this class was generated from the following files:

- `dc.hh`
- `dc.cpp`

## 5.81 dc\_simple::dc\_if\_t Class Reference

interface class

Inheritance diagram for dc\_simple::dc\_if\_t:



### Public Member Functions

- `dc_if_t ( algo_comm_t iac, const std::string &configured_name)`
- `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

- MHParse::string\_t clientid
- MHParse::string\_t gainrule
- MHParse::string\_t preset
- MHParse::int\_mon\_t modified
- MHParse::vfloat\_mon\_t mon\_l
- MHParse::vfloat\_mon\_t mon\_g
- MHParse::string\_t filterbank
- MHParse::vfloat\_mon\_t center\_frequencies
- MHParse::vfloat\_mon\_t edge\_frequencies
- MHAEvents::patchbay\_t< dc\_if\_t > patchbay
- bool prepared

## Additional Inherited Members

### 5.81.1 Detailed Description

interface class

### 5.81.2 Constructor & Destructor Documentation

```
5.81.2.1 dc_if_t() dc_simple::dc_if_t::dc_if_t (
    algo_comm_t iac,
    const std::string & configured_name )
```

### 5.81.3 Member Function Documentation

```
5.81.3.1 prepare() void dc_simple::dc_if_t::prepare (
    mhaconfig_t & tf ) [virtual]
```

Implements **MHAPlugin::plugin\_t< dc\_t >** (p. 1149).

**5.81.3.2 `release()`** void dc\_simple::dc\_if\_t::release ( ) [virtual]

Reimplemented from **MHAPlugin::plugin\_t<dc\_t>** (p. 1150).

**5.81.3.3 `process() [1/2]`** mha\_spec\_t \* dc\_simple::dc\_if\_t::process ( mha\_spec\_t \* s )

**5.81.3.4 `process() [2/2]`** mha\_wave\_t \* dc\_simple::dc\_if\_t::process ( mha\_wave\_t \* s )

**5.81.3.5 `update_dc()`** void dc\_simple::dc\_if\_t::update\_dc ( ) [private]

**5.81.3.6 `update_level()`** void dc\_simple::dc\_if\_t::update\_level ( ) [private]

**5.81.3.7 `has_been_modified()`** void dc\_simple::dc\_if\_t::has\_been\_modified ( ) [inline], [private]

**5.81.3.8 `read_modified()`** void dc\_simple::dc\_if\_t::read\_modified ( ) [inline], [private]

**5.81.3.9 `update_level_mon()`** void dc\_simple::dc\_if\_t::update\_level\_mon ( ) [private]

**5.81.3.10 `update_gain_mon()`** void dc\_simple::dc\_if\_t::update\_gain\_mon ( ) [private]

## 5.81.4 Member Data Documentation

**5.81.4.1 clientid** `MHAParser::string_t` `dc_simple::dc_if_t::clientid` [private]

**5.81.4.2 gainrule** `MHAParser::string_t` `dc_simple::dc_if_t::gainrule` [private]

**5.81.4.3 preset** `MHAParser::string_t` `dc_simple::dc_if_t::preset` [private]

**5.81.4.4 modified** `MHAParser::int_mon_t` `dc_simple::dc_if_t::modified` [private]

**5.81.4.5 mon\_l** `MHAParser::vfloat_mon_t` `dc_simple::dc_if_t::mon_l` [private]

**5.81.4.6 mon\_g** `MHAParser::vfloat_mon_t` `dc_simple::dc_if_t::mon_g` [private]

**5.81.4.7 filterbank** `MHAParser::string_t` `dc_simple::dc_if_t::filterbank` [private]

**5.81.4.8 center\_frequencies** `MHAParser::vfloat_mon_t` `dc_simple::dc_if_t::center_frequencies` [private]

**5.81.4.9 edge\_frequencies** `MHAPARSER::vfloat_mon_t dc_simple::dc_if_t::edge_frequencies` [private]

**5.81.4.10 patchbay** `MHAEVENTS::patchbay_t< dc_if_t> dc_simple::dc_if_t::patchbay` [private]

**5.81.4.11 prepared** `bool dc_simple::dc_if_t::prepared` [private]

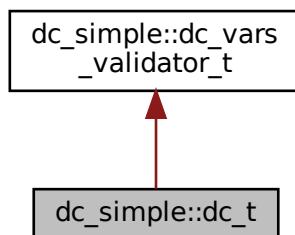
The documentation for this class was generated from the following files:

- `dc_simple.hh`
- `dc_simple.cpp`

## 5.82 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, unsigned int nch)`
- `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_l`
- `std::vector< float > mon_g`

## Private Attributes

- `std::vector< mha_real_t > expansion_threshold`
- `std::vector< mha_real_t > limiter_threshold`
- `std::vector< line_t > compression`
- `std::vector< line_t > expansion`
- `std::vector< line_t > limiter`
- `std::vector< mha_real_t > maxgain`
- `unsigned int nbands`

## Additional Inherited Members

### 5.82.1 Detailed Description

Runtime config class for `dc_simple` (p. 86) plugin.

### 5.82.2 Constructor & Destructor Documentation

```
5.82.2.1 dc_t() dc_simple::dc_t::dc_t (
    const dc_vars_t & vars,
    unsigned int nch )
```

### 5.82.3 Member Function Documentation

**5.82.3.1 process()** [1/2] `mha_spec_t * dc_simple::dc_t::process (`  
`mha_spec_t * s,`  
`mha_wave_t * level_db )`

**5.82.3.2 process()** [2/2] `mha_wave_t * dc_simple::dc_t::process (`  
`mha_wave_t * s,`  
`mha_wave_t * level_db )`

## 5.82.4 Member Data Documentation

**5.82.4.1 expansion\_threshold** `std::vector< mha_real_t > dc_simple::dc_t::expansion←`  
`_threshold` [private]

**5.82.4.2 limiter\_threshold** `std::vector< mha_real_t > dc_simple::dc_t::limiter←`  
`threshold` [private]

**5.82.4.3 compression** `std::vector< line_t > dc_simple::dc_t::compression` [private]

**5.82.4.4 expansion** `std::vector< line_t > dc_simple::dc_t::expansion` [private]

**5.82.4.5 limiter** `std::vector< line_t > dc_simple::dc_t::limiter` [private]

**5.82.4.6 maxgain** `std::vector< mha_real_t > dc_simple::dc_t::maxgain` [private]

**5.82.4.7 nbands** `unsigned int dc_simple::dc_t::nbands [private]`

**5.82.4.8 mon\_l** `std::vector<float> dc_simple::dc_t::mon_l`

**5.82.4.9 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`

## 5.83 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`

### 5.83.1 Constructor & Destructor Documentation

**5.83.1.1 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 )`

---

**5.83.1.2 line\_t() [2/2]** `dc_simple::dc_t::line_t::line_t (`  
 `mha_real_t x1,`  
 `mha_real_t y1,`  
 `mha_real_t m_ )`

## 5.83.2 Member Function Documentation

**5.83.2.1 operator()()** `mha_real_t dc_simple::dc_t::line_t::operator() (`  
 `mha_real_t x )` [inline]

## 5.83.3 Member Data Documentation

**5.83.3.1 m** `mha_real_t dc_simple::dc_t::line_t::m` [private]

**5.83.3.2 y0** `mha_real_t dc_simple::dc_t::line_t::y0` [private]

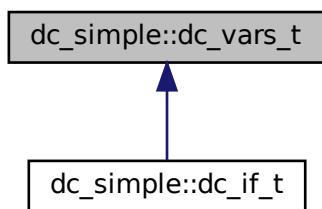
The documentation for this class was generated from the following files:

- `dc_simple.hh`
- `dc_simple.cpp`

## 5.84 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 ( MHParse::parser_t &p)`

## Public Attributes

- `MHParse::vfloat_t g50`
- `MHParse::vfloat_t g80`
- `MHParse::vfloat_t maxgain`
- `MHParse::vfloat_t expansion_threshold`
- `MHParse::vfloat_t expansion_slope`
- `MHParse::vfloat_t limiter_threshold`
- `MHParse::vfloat_t tauattack`
- `MHParse::vfloat_t taudecay`
- `MHParse::bool_t bypass`

### 5.84.1 Detailed Description

class for **dc\_simple** (p. 86) plugin which registers variables to **MHParse** (p. 122).

### 5.84.2 Constructor & Destructor Documentation

#### 5.84.2.1 `dc_vars_t()` `dc_simple::dc_vars_t::dc_vars_t ( MHParse::parser_t & p )`

### 5.84.3 Member Data Documentation

#### 5.84.3.1 `g50` `MHParse::vfloat_t dc_simple::dc_vars_t::g50`

#### 5.84.3.2 `g80` `MHParse::vfloat_t dc_simple::dc_vars_t::g80`

**5.84.3.3 maxgain** `MHAParser::vfloat_t dc_simple::dc_vars_t::maxgain`

**5.84.3.4 expansion\_threshold** `MHAParser::vfloat_t dc_simple::dc_vars_t::expansion_threshold`

**5.84.3.5 expansion\_slope** `MHAParser::vfloat_t dc_simple::dc_vars_t::expansion_slope`

**5.84.3.6 limiter\_threshold** `MHAParser::vfloat_t dc_simple::dc_vars_t::limiter_threshold`

**5.84.3.7 tauattack** `MHAParser::vfloat_t dc_simple::dc_vars_t::tauattack`

**5.84.3.8 taudecay** `MHAParser::vfloat_t dc_simple::dc_vars_t::taudecay`

**5.84.3.9 bypass** `MHAParser::bool_t dc_simple::dc_vars_t::bypass`

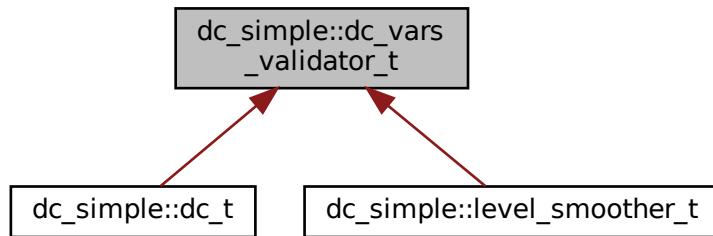
The documentation for this class was generated from the following files:

- `dc_simple.hh`
- `dc_simple.cpp`

## 5.85 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.*

#### 5.85.1 Detailed Description

Helper class to check sizes of configuration variable vectors.

#### 5.85.2 Constructor & Destructor Documentation

```
5.85.2.1 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

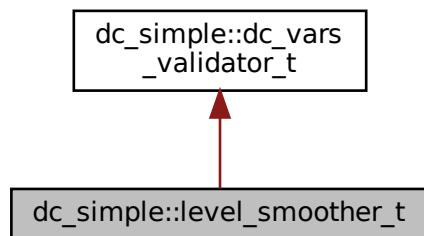
<b>MHA_Error</b> (p. 760)	if <i>s</i> == 0.
<b>MHA_Error</b> (p. 760)	if the size of any vector in <i>v</i> is neither <i>s</i> nor 1.

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

- **dc\_simple.hh**
- **dc\_simple.cpp**

## 5.86 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

### 5.86.1 Constructor & Destructor Documentation

**5.86.1.1 level\_smoothen\_t()** dc\_simple::level\_smoothen\_t::level\_smoothen\_t ( const dc\_vars\_t & vars, mha\_real\_t filter\_rate, mhaconfig\_t buscfg )

### 5.86.2 Member Function Documentation

**5.86.2.1 process() [1/2]** mha\_wave\_t \* dc\_simple::level\_smoothen\_t::process ( mha\_spec\_t \* s )

**5.86.2.2 process() [2/2]** mha\_wave\_t \* dc\_simple::level\_smoothen\_t::process ( mha\_wave\_t \* s )

### 5.86.3 Member Data Documentation

**5.86.3.1 attack** `MHAFilter::olflt_lowpass_t` `dc_simple::level_smoothen_t::attack`  
[private]

**5.86.3.2 decay** `MHAFilter::olflt_maxtrack_t` `dc_simple::level_smoothen_t::decay`  
[private]

**5.86.3.3 nbands** `unsigned int` `dc_simple::level_smoothen_t::nbands` [private]

**5.86.3.4 fftlen** `unsigned int` `dc_simple::level_smoothen_t::fftlens` [private]

**5.86.3.5 level\_wave** `MHASignal::waveform_t` `dc_simple::level_smoothen_t::level_wave`  
[private]

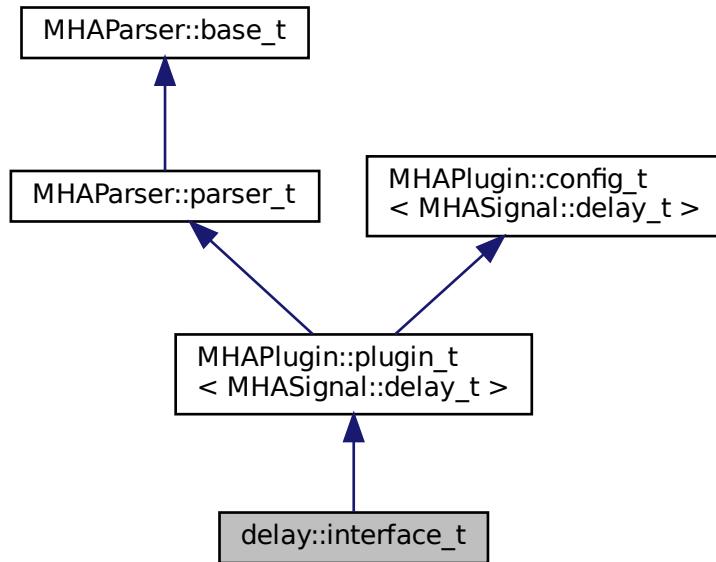
**5.86.3.6 level\_spec** `MHASignal::waveform_t` `dc_simple::level_smoothen_t::level_spec`  
[private]

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

- `dc_simple.hh`
- `dc_simple.cpp`

## 5.87 delay::interface\_t Class Reference

Inheritance diagram for delay::interface\_t:



### Public Member Functions

- `interface_t ( algo_comm_t iac, const std::string &configured_name)`
- `void prepare ( mhaconfig_t &)`
- `mha_wave_t * process ( mha_wave_t *)`

### Private Member Functions

- `void update ()`

### Private Attributes

- `MHAParser::vint_t delays`
- `MHAEvents::patchbay_t< interface_t > patchbay`

## Additional Inherited Members

### 5.87.1 Constructor & Destructor Documentation

**5.87.1.1 `interface_t()`** `delay::interface_t::interface_t ( algo_comm_t iac, const std::string & configured_name )`

### 5.87.2 Member Function Documentation

**5.87.2.1 `prepare()`** `void delay::interface_t::prepare ( mhaconfig_t & tf ) [virtual]`

Implements `MHAParser::plugin_t< MHASignal::delay_t >` (p. [1149](#)).

**5.87.2.2 `process()`** `mha_wave_t * delay::interface_t::process ( mha_wave_t * s )`

**5.87.2.3 `update()`** `void delay::interface_t::update ( ) [private]`

### 5.87.3 Member Data Documentation

**5.87.3.1 `delays`** `MHAParser::vint_t delay::interface_t::delays [private]`

**5.87.3.2 patchbay** `MHAEVENTS::patchbay_t< interface_t>` `delay::interface_t::patchbay`  
 [private]

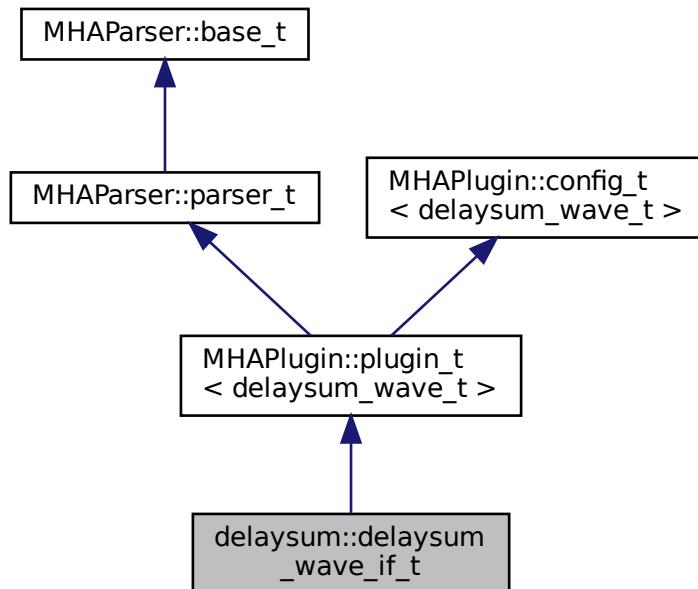
The documentation for this class was generated from the following files:

- `delay.hh`
- `delay.cpp`

## 5.88 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 ( algo_comm_t iac, const std::string &configured_name)`
- `mha_wave_t * process ( mha_wave_t *)`
- `void prepare ( mhaconfig_t &)`
- `void release ()`

## Private Member Functions

- void **update\_cfg ()**

## Private Attributes

- **MHAParser::vfloat\_t weights**  
*Linear weights to be multiplied with the audio signal, one factor for each channel.*
- **MHAParser::vint\_t delay**  
*vector of channel-specific delays, in samples.*
- **MHAEvents::patchbay\_t< delaysum\_wave\_if\_t > patchbay**  
*The patchbay to react to config changes.*

## Additional Inherited Members

### 5.88.1 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 output channel.

### 5.88.2 Constructor & Destructor Documentation

```
5.88.2.1 delaysum_wave_if_t() delaysum::delaysum_wave_if_t::delaysum_wave_if_t (
    algo_comm_t iac,
    const std::string & configured_name )
```

### 5.88.3 Member Function Documentation

```
5.88.3.1 process() mha_wave_t * delaysum::delaysum_wave_if_t::process (
    mha_wave_t * wave )
```

**5.88.3.2 `prepare()`** `void delaysum::delaysum_wave_if_t::prepare ( mhaconfig_t & tfcfg ) [virtual]`

Implements **MHAPlugIn::plugin\_t< delaysum\_wave\_t >** (p. 1149).

**5.88.3.3 `release()`** `void delaysum::delaysum_wave_if_t::release ( ) [virtual]`

Reimplemented from **MHAPlugIn::plugin\_t< delaysum\_wave\_t >** (p. 1150).

**5.88.3.4 `update_cfg()`** `void delaysum::delaysum_wave_if_t::update_cfg ( ) [private]`

## 5.88.4 Member Data Documentation

**5.88.4.1 `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, ...]

**5.88.4.2 `delay`** `MHAParser::vint_t delaysum::delaysum_wave_if_t::delay [private]`

vector of channel-specific delays, in samples.

**5.88.4.3 `patchbay`** `MHAEvents::patchbay_t< delaysum_wave_if_t > delaysum::delaysum<-wave_if_t::patchbay [private]`

The patchbay to react to config changes.

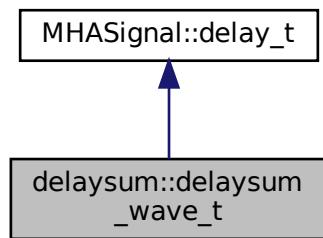
The documentation for this class was generated from the following file:

- **delaysum\_wave.cpp**

## 5.89 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.*

### 5.89.1 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.

## 5.89.2 Constructor & Destructor Documentation

### 5.89.2.1 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.

## 5.89.3 Member Function Documentation

### 5.89.3.1 process()

```
mha_wave_t * delaysum::delaysum_wave_t::process (
```

```
        mha_wave_t * signal )
```

## 5.89.4 Member Data Documentation

### 5.89.4.1 weights

```
std::vector< mha_real_t > delaysum::delaysum_wave_t::weights [private]
```

Relative weights for each channel. Order is [chan0, chan1, ...].

### 5.89.4.2 **out** `MHASignal::waveform_t delaysum::delaysum_wave_t::out` [private]

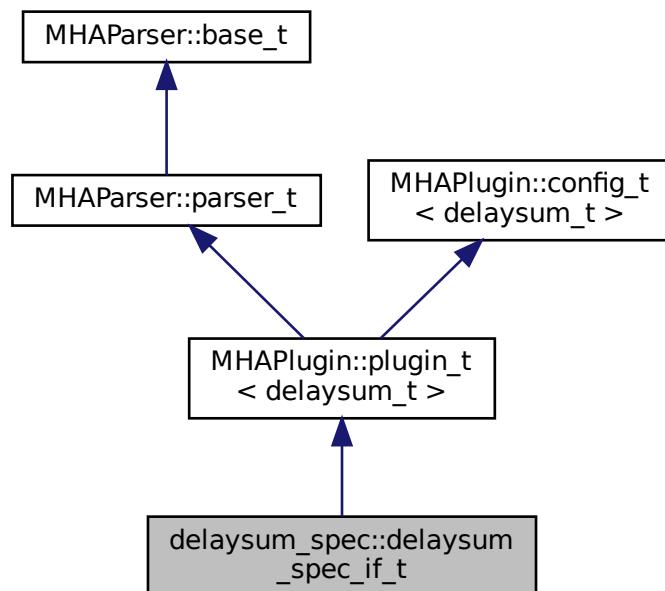
Output waveform.

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

- `delaysum_wave.cpp`

## 5.90 `delaysum_spec::delaysum_spec_if_t` Class Reference

Inheritance diagram for `delaysum_spec::delaysum_spec_if_t`:



### Public Member Functions

- `delaysum_spec_if_t ( algo_comm_t iac, const std::string &configured_name)`
- `mha_spec_t * process ( mha_spec_t *)`
- `void prepare ( mhaconfig_t &)`

### Private Member Functions

- `void update_cfg ()`

## Private Attributes

- **MHAParser::vfloat\_t groupdelay**
- **MHAParser::vfloat\_t gain**
- **MHAEvents::patchbay\_t< delaysum\_spec\_if\_t > patchbay**

## Additional Inherited Members

### 5.90.1 Constructor & Destructor Documentation

```
5.90.1.1 delaysum_spec_if_t() delaysum_spec::delaysum_spec_if_t::delaysum_spec_if_t (
    algo_comm_t iac,
    const std::string & configured_name )
```

### 5.90.2 Member Function Documentation

```
5.90.2.1 process() mha_spec_t * delaysum_spec::delaysum_spec_if_t::process (
    mha_spec_t * spec )
```

```
5.90.2.2 prepare() void delaysum_spec::delaysum_spec_if_t::prepare (
    mhaconfig_t & signal_info ) [virtual]
```

Implements **MHAPlugin::plugin\_t< delaysum\_t >** (p. [1149](#)).

```
5.90.2.3 update_cfg() void delaysum_spec::delaysum_spec_if_t::update_cfg ( ) [private]
```

### 5.90.3 Member Data Documentation

---

**5.90.3.1 groupdelay** `MHAParser::vfloat_t delaysum_spec::delaysum_spec_if_t::groupdelay` [private]

**5.90.3.2 gain** `MHAParser::vfloat_t delaysum_spec::delaysum_spec_if_t::gain` [private]

**5.90.3.3 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**

## 5.91 delaysum\_spec::delaysum\_t Class Reference

### Public Member Functions

- **delaysum\_t** (`std::vector< float > groupdelay, std::vector< float > gain, unsigned int nChannels, unsigned int nFFT, float fs)`)
- **mha\_spec\_t \* process ( mha\_spec\_t \*)**

### Private Attributes

- **MHASignal::spectrum\_t scale**
- **MHASignal::spectrum\_t output**

### 5.91.1 Constructor & Destructor Documentation

**5.91.1.1 delaysum\_t()** `delaysum_spec::delaysum_t::delaysum_t (`  
`std::vector< float > groupdelay,`  
`std::vector< float > gain,`  
`unsigned int nChannels,`  
`unsigned int nFFT,`  
`float fs )`

## 5.91.2 Member Function Documentation

**5.91.2.1 process()** `mha_spec_t * delaysum_spec::delaysum_t::process (mha_spec_t * spec )`

## 5.91.3 Member Data Documentation

**5.91.3.1 scale** `MHASignal::spectrum_t delaysum_spec::delaysum_t::scale [private]`

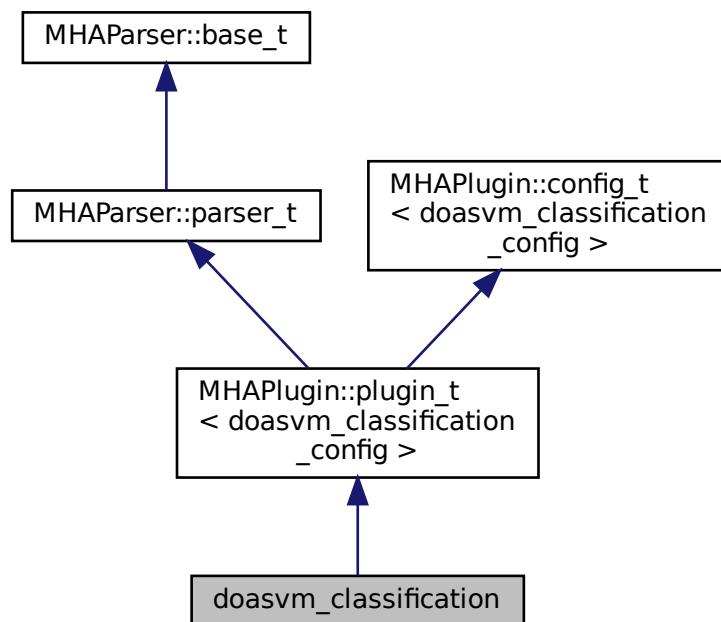
**5.91.3.2 output** `MHASignal::spectrum_t delaysum_spec::delaysum_t::output [private]`

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

- `delaysum_spec.cpp`

## 5.92 doasvm\_classification Class Reference

Inheritance diagram for doasvm\_classification:



## Public Member Functions

- **doasvm\_classification** ( **algo\_comm\_t** iac, const std::string &configured\_name)  
*Constructs our plugin.*
- **~doasvm\_classification** ()
- **mha\_wave\_t \* process** ( **mha\_wave\_t** \*)  
*Checks for the most recent configuration and defers processing to it.*
- **void prepare** ( **mhaconfig\_t** &)  
*Plugin preparation.*
- **void release** (void)

## Public Attributes

- **MHAParser::vfloat\_t angles**
- **MHAParser::mfloat\_t w**
- **MHAParser::vfloat\_t b**
- **MHAParser::vfloat\_t x**
- **MHAParser::vfloat\_t y**
- **MHAParser::string\_t p\_name**
- **MHAParser::string\_t max\_p\_ind\_name**
- **MHAParser::string\_t vGCC\_name**

## Private Member Functions

- **void update\_cfg** ()

## Private Attributes

- **MHAEvents::patchbay\_t< doasvm\_classification > patchbay**

## Additional Inherited Members

### 5.92.1 Constructor & Destructor Documentation

**5.92.1.1 doasvm\_classification()** `doasvm_classification::doasvm_classification ( algo_comm_t iac, const std::string & configured_name )`

Constructs our plugin.

```
5.92.1.2 ~doasvm_classification() doasvm_classification::~doasvm_classification ( )
```

## 5.92.2 Member Function Documentation

```
5.92.2.1 process() mha_wave_t * doasvm_classification::process ( mha_wave_t * signal )
```

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

```
5.92.2.2 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. [1149](#)).

```
5.92.2.3 release() void doasvm_classification::release ( void ) [inline], [virtual]
```

Reimplemented from **MHAPlugin::plugin\_t< doasvm\_classification\_config >** (p. [1150](#)).

```
5.92.2.4 update_cfg() void doasvm_classification::update_cfg ( ) [private]
```

### 5.92.3 Member Data Documentation

**5.92.3.1 angles** `MHAParser::vfloat_t doasvm_classification::angles`

**5.92.3.2 w** `MHAParser::mfloat_t doasvm_classification::w`

**5.92.3.3 b** `MHAParser::vfloat_t doasvm_classification::b`

**5.92.3.4 x** `MHAParser::vfloat_t doasvm_classification::x`

**5.92.3.5 y** `MHAParser::vfloat_t doasvm_classification::y`

**5.92.3.6 p\_name** `MHAParser::string_t doasvm_classification::p_name`

**5.92.3.7 max\_p\_ind\_name** `MHAParser::string_t doasvm_classification::max_p_ind_←  
name`

**5.92.3.8 vGCC\_name** `MHAParser::string_t doasvm_classification::vGCC_name`

**5.92.3.9 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`

## 5.93 doasvm\_classification\_config Class Reference

### Public Member Functions

- `doasvm_classification_config ( algo_comm_t & ac, doasvm_classification * _doasvm )`
- `~doasvm_classification_config ()`
- `mha_wave_t * process ( mha_wave_t * )`

### Public Attributes

- `algo_comm_t & ac`
- `doasvm_classification * doasvm`
- `MHA_AC::waveform_t p`
- `MHA_AC::int_t p_max`
- `mha_wave_t c`

### 5.93.1 Constructor & Destructor Documentation

**5.93.1.1 doasvm\_classification\_config()** `doasvm_classification_config::doasvm_classification_config ( algo_comm_t & ac, doasvm_classification * _doasvm )`

**5.93.1.2 ~doasvm\_classification\_config()** `doasvm_classification_config::~doasvm_classification_config ( )`

## 5.93.2 Member Function Documentation

**5.93.2.1 process()** `mha_wave_t * doasvm_classification_config::process (`  
`mha_wave_t * wave )`

## 5.93.3 Member Data Documentation

**5.93.3.1 ac** `algo_comm_t& doasvm_classification_config::ac`

**5.93.3.2 doasvm** `doasvm_classification* doasvm_classification_config::doasvm`

**5.93.3.3 p** `MHA_AC::waveform_t doasvm_classification_config::p`

**5.93.3.4 p\_max** `MHA_AC::int_t doasvm_classification_config::p_max`

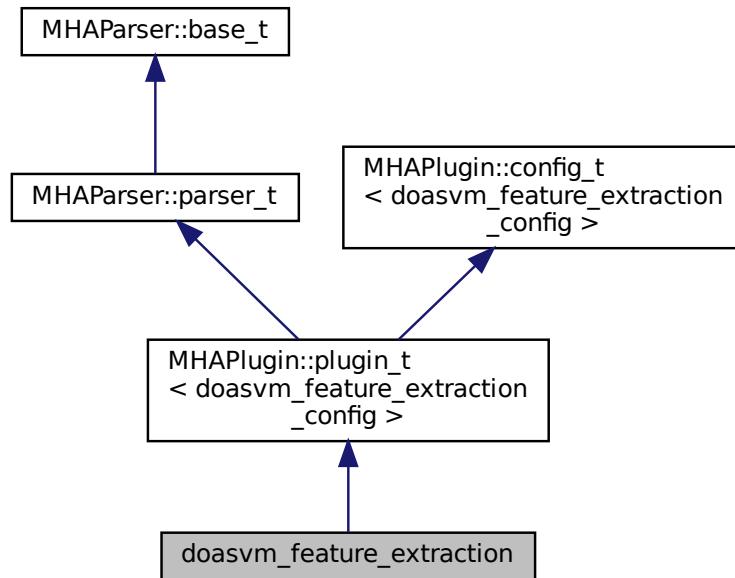
**5.93.3.5 c** `mha_wave_t doasvm_classification_config::c`

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

- **doasvm\_classification.h**
- **doasvm\_classification.cpp**

## 5.94 doasvm\_feature\_extraction Class Reference

Inheritance diagram for doasvm\_feature\_extraction:



### Public Member Functions

- **doasvm\_feature\_extraction ( algo\_comm\_t iac, const std::string &configured\_name)**  
*Constructs our plugin.*
- **~doasvm\_feature\_extraction ()**
- **mha\_wave\_t \* process ( mha\_wave\_t \*)**  
*Checks for the most recent configuration and defers processing to it.*
- **void prepare ( mhaconfig\_t &)**  
*Plugin preparation.*
- **void release (void)**

### Public Attributes

- **MHAParser::int\_t fftlen**
- **MHAParser::int\_t max\_lag**
- **MHAParser::int\_t nupsample**
- **MHAParser::string\_t vGCC\_name**

**Private Member Functions**

- void **update\_cfg ()**

**Private Attributes**

- **MHAEvents::patchbay\_t< doasvm\_feature\_extraction > patchbay**

**Additional Inherited Members****5.94.1 Constructor & Destructor Documentation**

**5.94.1.1 doasvm\_feature\_extraction()** doasvm\_feature\_extraction::doasvm\_feature\_extraction (   
`algo_comm_t iac,`  
`const std::string & configured_name )`

Constructs our plugin.

**5.94.1.2 ~doasvm\_feature\_extraction()** doasvm\_feature\_extraction::~doasvm\_feature\_extraction ( )

**5.94.2 Member Function Documentation**

**5.94.2.1 process()** mha\_wave\_t \* doasvm\_feature\_extraction::process (   
`mha_wave_t * signal )`

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

**5.94.2.2 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. [1149](#)).

**5.94.2.3 release()** void doasvm\_feature\_extraction::release ( void ) [inline], [virtual]

Reimplemented from **MHAParser::plugin\_t< doasvm\_feature\_extraction\_config >** (p. [1150](#)).

**5.94.2.4 update\_cfg()** void doasvm\_feature\_extraction::update\_cfg ( ) [private]

### 5.94.3 Member Data Documentation

**5.94.3.1 fftlen** **MHAParser::int\_t** doasvm\_feature\_extraction::fftlen

**5.94.3.2 max\_lag** **MHAParser::int\_t** doasvm\_feature\_extraction::max\_lag

**5.94.3.3 nupsample** **MHAParser::int\_t** doasvm\_feature\_extraction::nupsample

**5.94.3.4 vGCC\_name** **MHAParser::string\_t** doasvm\_feature\_extraction::vGCC\_name

**5.94.3.5 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`

## 5.95 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`

### 5.95.1 Constructor & Destructor Documentation

```
5.95.1.1 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 )
```

```
5.95.1.2 ~doasvm_feature_extraction_config() doasvm_feature_extraction_config<=
::~doasvm_feature_extraction_config ( )
```

## 5.95.2 Member Function Documentation

```
5.95.2.1 process() mha_wave_t * doasvm_feature_extraction_config::process (
    mha_wave_t * wave )
```

## 5.95.3 Member Data Documentation

```
5.95.3.1 doagcc doasvm_feature_extraction* doasvm_feature_extraction_config<=
::doagcc
```

```
5.95.3.2 wndlen unsigned int doasvm_feature_extraction_config::wndlen
```

```
5.95.3.3 fftlen unsigned int doasvm_feature_extraction_config::fftlens
```

```
5.95.3.4 G_length unsigned int doasvm_feature_extraction_config::G_length
```

**5.95.3.5 `GCC_start`** `unsigned int doasvm_feature_extraction_config::GCC_start`

**5.95.3.6 `GCC_end`** `unsigned int doasvm_feature_extraction_config::GCC_end`

**5.95.3.7 `vGCC_ac`** `MHA_AC::waveform_t doasvm_feature_extraction_config::vGCC_ac`

**5.95.3.8 `fft`** `mha_fft_t doasvm_feature_extraction_config::fft`

**5.95.3.9 `ifft`** `mha_fft_t doasvm_feature_extraction_config::ifft`

**5.95.3.10 `hifftwin_sum`** `double doasvm_feature_extraction_config::hifftwin_sum`

**5.95.3.11 `proc_wave`** `MHASignal::waveform_t doasvm_feature_extraction_config::proc_wave`

**5.95.3.12 `hwin`** `MHASignal::waveform_t doasvm_feature_extraction_config::hwin`

**5.95.3.13 `hifftwin`** `MHASignal::waveform_t doasvm_feature_extraction_config::hifftwin`

**5.95.3.14 vGCC** `MHASignal::waveform_t` `doasvm_feature_extraction_config::vGCC`**5.95.3.15 in\_spec** `MHASignal::spectrum_t` `doasvm_feature_extraction_config::in_spec`**5.95.3.16 G** `MHASignal::spectrum_t` `doasvm_feature_extraction_config::G`

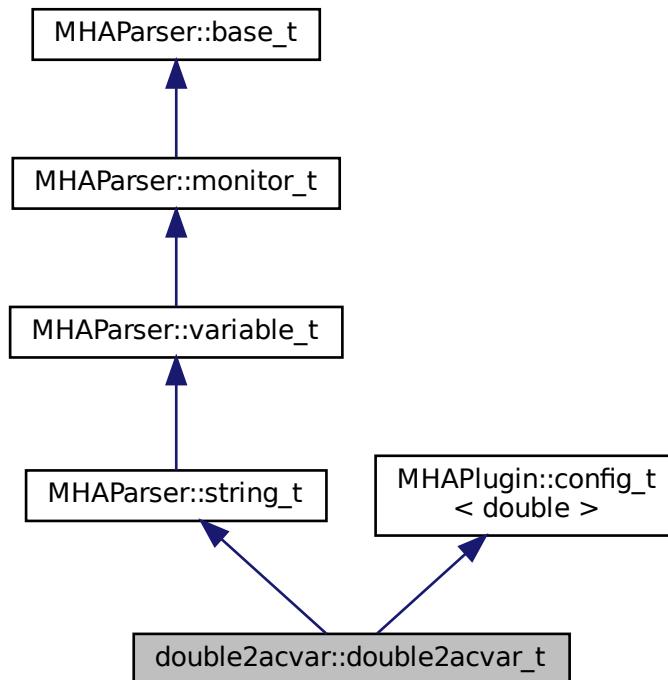
The documentation for this class was generated from the following files:

- `doasvm_feature_extraction.h`
- `doasvm_feature_extraction.cpp`

**5.96 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** iac, const std::string &configured\_name)  
*Standard plugin constructor.*
- **~double2acvar\_t** ()=default
- template<class T >  
**T \* process** (T \*s)  
*process() (p. 431) does not alter the signal and has same implementation regardless of signal domain.*
- **void poll\_latest\_value\_and\_reinsert ()**  
*Called from process() (p. 431) and, when allowed, also from on\_configuration\_update() (p. 432).*
- **void prepare\_** (**mhaconfig\_t** &)  
*Prepare method as expected by the plugin interface macros.*
- **void release\_** ()  
*Release method as expected by the plugin interface macros.*
- **void on\_configuration\_update ()**  
*Callback function on write access to the string configuration value.*

## Private Attributes

- **MHA\_AC::double\_t ac\_double**  
*AC variable inserted by this plugin.*
- **MHAEvents::patchbay\_t< double2acvar\_t > patchbay**  
*Callback router.*
- **bool is\_prepared**  
*Flag to keep track if we are currently prepared.*

## Additional Inherited Members

### 5.96.1 Detailed Description

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

### 5.96.2 Constructor & Destructor Documentation

#### 5.96.2.1 **double2acvar\_t()** `double2acvar::double2acvar_t::double2acvar_t (` `algo_comm_t iac,` `const std::string & configured_name )`

Standard plugin constructor.

**Parameters**

<i>iac</i>	Algorithm communication variable space.
<i>configured_name</i>	Configured name of this plugin, also used as name of the AC variable.

**5.96.2.2 ~double2acvar\_t()** `double2acvar::double2acvar_t::~double2acvar_t ( ) [default]`

**5.96.3 Member Function Documentation**

**5.96.3.1 process()** `template<class T >`  
`T * double2acvar::double2acvar_t::process (`  
`T * s )`

**process()** (p. 431) does not alter the signal and has same implementation regardless of signal domain.

**Parameters**

<i>s</i>	Pointer to input signal structure, <b>mha_wave_t</b> (p. 836) or <b>mha_spec_t</b> (p. 790).
----------	--

**Returns**

*s*, unaltered.

**5.96.3.2 poll\_latest\_value\_and\_reinsert()** `void double2acvar::double2acvar_t::poll_latest_value_and_reinsert ( )`

Called from **process()** (p. 431) and, when allowed, also from **on\_configuration\_update()** (p. 432).

**poll\_latest\_value\_and\_reinsert()** (p. 431) retrieves the latest configured value and reinserts the AC variable into the AC space.

```
5.96.3.3 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.

```
5.96.3.4 release_() void double2acvar::double2acvar_t::release_ ( )
```

Release method as expected by the plugin interface macros.

Resets is\_prepared flag.

```
5.96.3.5 on_configuration_update() void double2acvar::double2acvar_t::on_configuration_update ( )
```

Callback function on write access to the string configuration value.

## 5.96.4 Member Data Documentation

```
5.96.4.1 ac_double MHA_AC::double_t double2acvar::double2acvar_t::ac_double [private]
```

AC variable inserted by this plugin.

```
5.96.4.2 patchbay MHAEvents::patchbay_t< double2acvar_t> double2acvar::double2acvar_t::patchbay [private]
```

Callback router.

### 5.96.4.3 `is_prepared` `bool double2acvar::double2acvar_t::is_prepared [private]`

Flag to keep track if we are currently prepared.

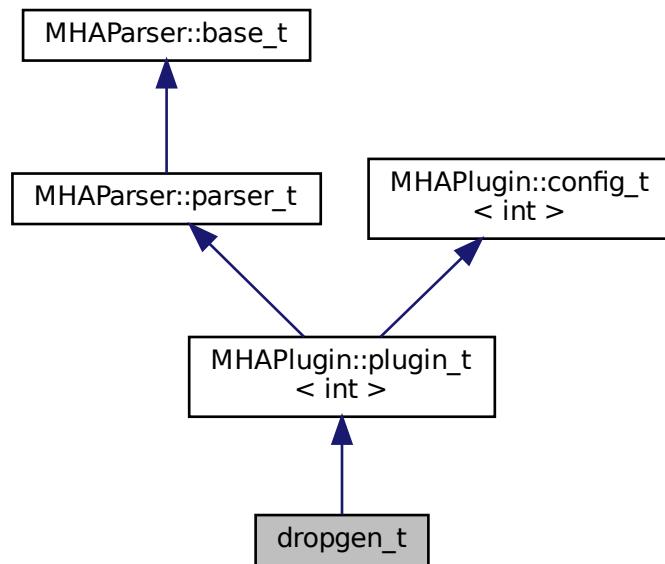
If we are, then signal processing is active and AC variables may only be accessed when MHA is currently executing out `process()` (p. 431) method.

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

- `double2acvar.cpp`

## 5.97 dropgen\_t Class Reference

Inheritance diagram for dropgen\_t:



### Public Member Functions

- `dropgen_t ( algo_comm_t iac, const std::string &configured_name)`
- `mha_wave_t * process ( mha_wave_t *)`
- `mha_spec_t * process ( mha_spec_t *)`
- `void prepare ( mhaconfig_t &)`
- `void release ()`

## Public Attributes

- `MHAParser::float_t min_sleep_time`
- `MHAParser::float_t max_sleep_time`
- `MHAParser::float_t chance`
- `MHAEvents::patchbay_t< dropgen_t > patchbay`
- `std::random_device r`
- `std::mt19937 random_engine`
- `std::uniform_real_distribution dis`

## Additional Inherited Members

### 5.97.1 Constructor & Destructor Documentation

**5.97.1.1 `dropgen_t()`** `dropgen_t::dropgen_t (`  
    `algo_comm_t iac,`  
    `const std::string & configured_name )`

### 5.97.2 Member Function Documentation

**5.97.2.1 `process() [1/2]`** `mha_wave_t * dropgen_t::process (`  
    `mha_wave_t * s )`

**5.97.2.2 `process() [2/2]`** `mha_spec_t * dropgen_t::process (`  
    `mha_spec_t * s )`

**5.97.2.3 `prepare()`** `void dropgen_t::prepare (`  
    `mhaconfig_t & ) [virtual]`

Implements `MHAParser::MHAPlugin< int >` (p. [1149](#)).

**5.97.2.4 `release()`** `void dropgen_t::release () [virtual]`

Reimplemented from **MHAParser::float\_t< int >** (p. 1150).

### 5.97.3 Member Data Documentation

**5.97.3.1 `min_sleep_time`** `MHAParser::float_t dropgen_t::min_sleep_time`

**5.97.3.2 `max_sleep_time`** `MHAParser::float_t dropgen_t::max_sleep_time`

**5.97.3.3 `chance`** `MHAParser::float_t dropgen_t::chance`

**5.97.3.4 `patchbay`** `MHAEvents::patchbay_t< dropgen_t > dropgen_t::patchbay`

**5.97.3.5 `r`** `std::random_device dropgen_t::r`

**5.97.3.6 `random_engine`** `std::mt19937 dropgen_t::random_engine`

**5.97.3.7 `dis`** `std::uniform_real_distribution dropgen_t::dis`

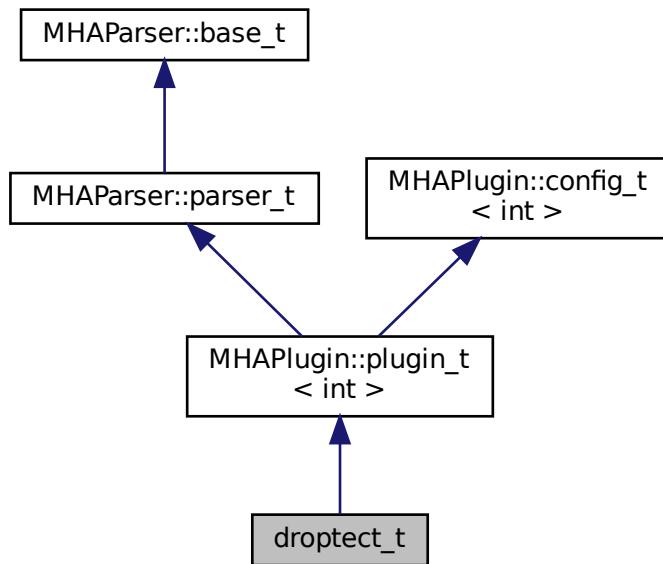
The documentation for this class was generated from the following file:

- **dropgen.cpp**

## 5.98 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 iac, const std::string &configured_name)`**  
*This constructor initializes the configuration language variables and inserts them into the MHA configuration tree.*
- **`void prepare ( mhaconfig_t &signal_info)`**  
*Allocates and initializes storage for this algorithm.*
- **`void release (void)`**  
*Deallocates storage.*
- **`mha_spec_t * process ( mha_spec_t *signal)`**  
*Compares current spectrum against history.*

### Private Attributes

- `MHAParser::vint_mon_t dropouts`
- `MHAParser::vint_mon_t consecutive_dropouts`
- `MHAParser::int_mon_t blocks`

- **MHAParser::bool\_t reset**
- **MHAParser::float\_t threshold**
- **MHASignal::waveform\_t \* current\_powspec**
- **MHASignal::waveform\_t \* filtered\_powspec**
- **MHAParser::float\_t tau**
- **std::vector< bool > filter\_activated**
- **float period**  
*The period of the process callback (duration of fragsize in seconds)*
- **MHAParser::mfloat\_mon\_t filtered\_powspec\_mon**  
*User access to filtered spectrum.*
- **MHAParser::vfloat\_mon\_t level\_mon**  
*User access to broadband levels.*

## Additional Inherited Members

### 5.98.1 Detailed Description

Detect dropouts in a signal with a constant spectrum.

### 5.98.2 Constructor & Destructor Documentation

```
5.98.2.1 droptect_t() droptect_t::droptect_t (
    algo_comm_t iac,
    const std::string & configured_name )
```

This constructor initializes the configuration language variables and inserts them into the MHA configuration tree.

### 5.98.3 Member Function Documentation

```
5.98.3.1 prepare() void droptect_t::prepare (
    mhaconfig_t & signal_info ) [virtual]
```

Allocates and initializes storage for this algorithm.

**Parameters**

<code>signal_info</code>	contains fft length, number of channels, fft length and hop size.
--------------------------	---

Implements **MHAPlugIn::plugin\_t< int >** (p. 1149).

**5.98.3.2 `release()`** `void droptect_t::release (`  
`void ) [virtual]`

Deallocates storage.

Reimplemented from **MHAPlugIn::plugin\_t< int >** (p. 1150).

**5.98.3.3 `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.

## 5.98.4 Member Data Documentation

**5.98.4.1 `dropouts`** `MHAParser::vint_mon_t droptect_t::dropouts [private]`

**5.98.4.2 `consecutive_dropouts`** `MHAParser::vint_mon_t droptect_t::consecutive_←`  
`dropouts [private]`

**5.98.4.3 `blocks`** `MHAParser::int_mon_t droptect_t::blocks [private]`

**5.98.4.4 reset** `MHAParser::bool_t droptect_t::reset` [private]

**5.98.4.5 threshold** `MHAParser::float_t droptect_t::threshold` [private]

**5.98.4.6 current\_powspec** `MHASignal::waveform_t* droptect_t::current_powspec` [private]

**5.98.4.7 filtered\_powspec** `MHASignal::waveform_t* droptect_t::filtered_powspec` [private]

**5.98.4.8 tau** `MHAParser::float_t droptect_t::tau` [private]

**5.98.4.9 filter\_activated** `std::vector<bool> droptect_t::filter_activated` [private]

**5.98.4.10 period** `float droptect_t::period` [private]

The period of the process callback (duration of fragsize in seconds)

**5.98.4.11 filtered\_powspec\_mon** `MHAParser::mfloat_mon_t droptect_t::filtered_<→powspec_mon` [private]

User access to filtered spectrum.

### 5.98.4.12 **level\_mon** `MHAParser::vfloat_mon_t droptect_t::level_mon [private]`

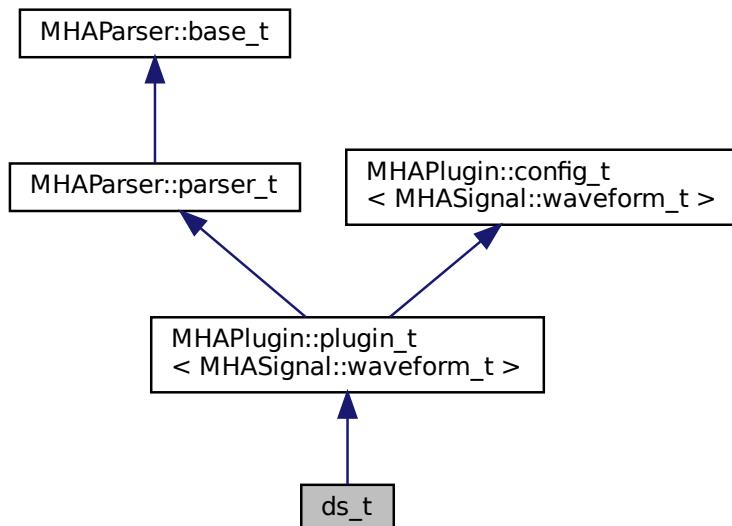
User access to broadband levels.

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

- `droptect.cpp`

## 5.99 **ds\_t** Class Reference

Inheritance diagram for `ds_t`:



### Public Member Functions

- `ds_t ( algo_comm_t iac, const std::string &configured_name )`
- `mha_wave_t * process ( mha_wave_t * )`
- `void prepare ( mhaconfig_t & )`
- `void release ()`

### Private Attributes

- `MHAParser::int_t ratio`
- `MHAFilter::iir_filter_t antialias`

## Additional Inherited Members

### 5.99.1 Constructor & Destructor Documentation

```
5.99.1.1 ds_t() ds_t::ds_t (
    algo_comm_t iac,
    const std::string & configured_name )
```

### 5.99.2 Member Function Documentation

```
5.99.2.1 process() mha_wave_t * ds_t::process (
    mha_wave_t * s )
```

```
5.99.2.2 prepare() void ds_t::prepare (
    mhaconfig_t & cf ) [virtual]
```

Implements **MHAParser::plugin\_t< MHASignal::waveform\_t >** (p. [1149](#)).

```
5.99.2.3 release() void ds_t::release () [virtual]
```

Reimplemented from **MHAParser::plugin\_t< MHASignal::waveform\_t >** (p. [1150](#)).

### 5.99.3 Member Data Documentation

```
5.99.3.1 ratio MHAParser::int_t ds_t::ratio [private]
```

### 5.99.3.2 **antialias** `MHAFilter::iir_filter_t ds_t::antialias [private]`

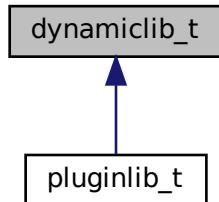
The documentation for this class was generated from the following file:

- `downsample.cpp`

## 5.100 **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.*

### 5.100.1 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

### 5.100.2 Constructor & Destructor Documentation

#### 5.100.2.1 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 library 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

<i>name_</i>	File name of the shared library, without suffix
--------------	---

## Exceptions

<b>MHA_Error</b> (p. 760)	if the library can not be found or can not be loaded
---------------------------	--

### 5.100.2.2 ~dynamiclib\_t() dynamiclib\_t::~dynamiclib\_t ( ) [virtual]

D'tor.

Closes the library handle.

### 5.100.2.3 dynamiclib\_t() [2/2] dynamiclib\_t::dynamiclib\_t ( ) [protected]

Default constructor.

## 5.100.3 Member Function Documentation

### 5.100.3.1 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

<i>name</i> ↵ —	Name of the function to be resolved
--------------------	-------------------------------------

#### Returns

Pointer to the function

Reimplemented in **pluginlib\_t** (p. 1358).

**5.100.3.2 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

**5.100.3.3 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

**5.100.3.4 getname()** virtual const std::string& dynamiclib\_t::getname ( ) const [inline], [virtual]

**5.100.3.5 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
—	

**5.100.4 Member Data Documentation****5.100.4.1 fullname** std::string dynamiclib\_t::fullname [protected]

Fully qualified file name of the library.

**5.100.4.2 modulename** std::string dynamiclib\_t::modulename [protected]

Unqualified file name of the library.

**5.100.4.3 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**

**5.101 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 srte, const **dc\_afterburn\_vars\_t** &vars)
- void **burn** (float &Gin, float Lin, unsigned int band, unsigned int channel)  
*gain modifier method (afterburn).*

## Private Attributes

- std::vector< float > **drain\_inv**
- std::vector< float > **conflux**
- std::vector< float > **maxgain**
- std::vector< float > **mpo\_inv**
- std::vector< MHAFilter::o1flt\_lowpass\_t > **lp**

### 5.101.1 Detailed Description

Real-time class for after burn effect.

The constructor processes the parameters and creates pre-processed variables for efficient realtime processing.

### 5.101.2 Constructor & Destructor Documentation

```
5.101.2.1 dc_afterburn_rt() DynComp::dc_afterburn_rt_t::dc_afterburn_rt_t (
    const std::vector< float > & cf,
    unsigned int channels,
    float srate,
    const dc_afterburn_vars_t & vars )
```

### 5.101.3 Member Function Documentation

```
5.101.3.1 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 *Gin* \* *Lin*.

### 5.101.4 Member Data Documentation

**5.101.4.1 drain\_inv** std::vector<float> DynComp::dc\_afterburn\_rt\_t::drain\_inv [private]

**5.101.4.2 conflux** std::vector<float> DynComp::dc\_afterburn\_rt\_t::conflux [private]

**5.101.4.3 maxgain** std::vector<float> DynComp::dc\_afterburn\_rt\_t::maxgain [private]

**5.101.4.4 mpo\_inv** std::vector<float> DynComp::dc\_afterburn\_rt\_t::mpo\_inv [private]

**5.101.4.5 lp** std::vector< **MHAFilter::olflt\_lowpass\_t**> DynComp::dc\_afterburn\_rt\_t::lp [private]

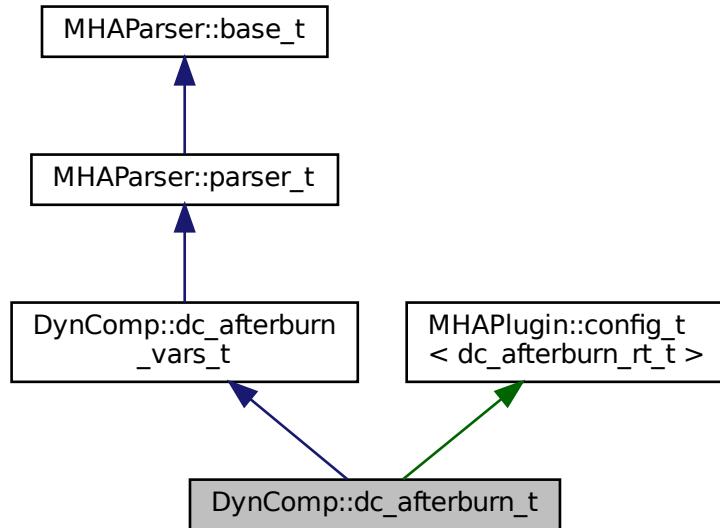
The documentation for this class was generated from the following files:

- **dc\_afterburn.h**
- **dc\_afterburn.cpp**

## 5.102 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

### 5.102.1 Detailed Description

Afterburn class, to be defined as a member of compressors.

### 5.102.2 Constructor & Destructor Documentation

#### 5.102.2.1 **dc\_afterburn\_t()** `DynComp::dc_afterburn_t::dc_afterburn_t ()`

### 5.102.3 Member Function Documentation

#### 5.102.3.1 **set\_fb\_pars()** `void DynComp::dc_afterburn_t::set_fb_pars ( const std::vector< float > & cf, unsigned int channels, float srate )`

#### 5.102.3.2 **unset\_fb\_pars()** `void DynComp::dc_afterburn_t::unset_fb_pars ()`

#### 5.102.3.3 **update\_burner()** `void DynComp::dc_afterburn_t::update_burner () [inline]`

#### 5.102.3.4 **burn()** `void DynComp::dc_afterburn_t::burn ( float & Gin, float Lin, unsigned int band, unsigned int channel ) [inline]`

**5.102.3.5 update()** void DynComp::dc\_afterburn\_t::update ( ) [private]

## 5.102.4 Member Data Documentation

**5.102.4.1 patchbay** `MHAEvents::patchbay_t< dc_afterburn_t>` DynComp::dc\_afterburn\_t::patchbay [private]

**5.102.4.2 \_cf** std::vector<float> DynComp::dc\_afterburn\_t::\_cf [private]

**5.102.4.3 \_channels** unsigned int DynComp::dc\_afterburn\_t::\_channels [private]

**5.102.4.4 \_srate** float DynComp::dc\_afterburn\_t::\_srate [private]

**5.102.4.5 commit\_pending** bool DynComp::dc\_afterburn\_t::commit\_pending [private]

**5.102.4.6 fb\_pars\_configured** bool DynComp::dc\_afterburn\_t::fb\_pars\_configured [private]

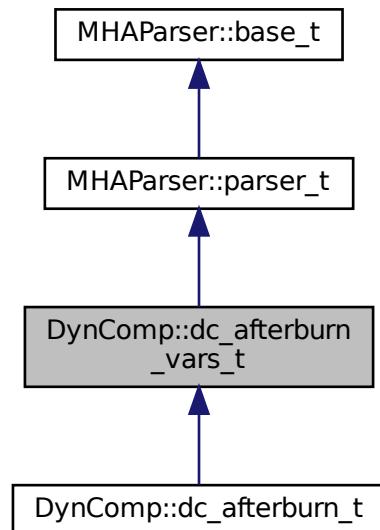
The documentation for this class was generated from the following files:

- **dc\_afterburn.h**
- **dc\_afterburn.cpp**

## 5.103 DynComp::dc\_afterburn\_vars\_t Class Reference

Variables for `dc_afterburn_t` (p. 449) 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

### 5.103.1 Detailed Description

Variables for **dc\_afterburn\_t** (p. 449) class.

### 5.103.2 Constructor & Destructor Documentation

**5.103.2.1 dc\_afterburn\_vars\_t()** DynComp::dc\_afterburn\_vars\_t::dc\_afterburn\_vars\_t ( )

### 5.103.3 Member Data Documentation

**5.103.3.1 f MHPARSER::vfloat\_t** DynComp::dc\_afterburn\_vars\_t::f

**5.103.3.2 drain MHPARSER::vfloat\_t** DynComp::dc\_afterburn\_vars\_t::drain

**5.103.3.3 conflux MHPARSER::vfloat\_t** DynComp::dc\_afterburn\_vars\_t::conflux

**5.103.3.4 maxgain MHPARSER::vfloat\_t** DynComp::dc\_afterburn\_vars\_t::maxgain

**5.103.3.5 mpo MHPARSER::vfloat\_t** DynComp::dc\_afterburn\_vars\_t::mpo

### 5.103.3.6 **taugain** `MHAParser::float_t DynComp::dc_afterburn_vars_t::taugain`

### 5.103.3.7 **commit** `MHAParser::kw_t DynComp::dc_afterburn_vars_t::commit`

### 5.103.3.8 **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`

## 5.104 `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**

### 5.104.1 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. 456) member function via a gain prescription rule from an auditory profile.

### 5.104.2 Constructor & Destructor Documentation

```
5.104.2.1 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).

### 5.104.2.2 ~gaintable\_t()

### 5.104.3 Member Function Documentation

**5.104.3.1 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.

**5.104.3.2 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

**5.104.3.3 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

**5.104.3.4 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.

**5.104.3.5 nbands()** unsigned int DynComp::gaintable\_t::nbands ( ) const [inline]

Return number of frequency bands.

**5.104.3.6 nchannels()** unsigned int DynComp::gaintable\_t::nchannels ( ) const [inline]

Return number of audio channels.

**5.104.3.7 get\_iofun()** std::vector< std::vector< mha\_real\_t > > gaintable\_t::get\_iofun ( ) const

Return current input-output function.

**5.104.3.8 get\_vL()** std::vector< mha\_real\_t > DynComp::gaintable\_t::get\_vL ( ) const [inline]

**5.104.3.9 `get_vF()`** `std::vector< mha_real_t> DynComp::gaintable_t::get_vF ( ) const [inline]`

## 5.104.4 Member Data Documentation

**5.104.4.1 `num_L`** `unsigned int DynComp::gaintable_t::num_L [private]`

**5.104.4.2 `num_F`** `unsigned int DynComp::gaintable_t::num_F [private]`

**5.104.4.3 `num_channels`** `unsigned int DynComp::gaintable_t::num_channels [private]`

**5.104.4.4 `vL`** `std::vector< mha_real_t> DynComp::gaintable_t::vL [private]`

**5.104.4.5 `vF`** `std::vector< mha_real_t> DynComp::gaintable_t::vF [private]`

**5.104.4.6 `vFlag`** `std::vector< mha_real_t> DynComp::gaintable_t::vFlag [private]`

**5.104.4.7 `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`

## 5.105 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**

#### 5.105.1 Constructor & Destructor Documentation

**5.105.1.1 cfg\_t() [1/2]** `cfg_t::cfg_t (`  
    `int infft,`  
    `int inchannels,`  
    `std::vector< std::vector< float > > ifgains )`

**5.105.1.2 cfg\_t() [2/2]** `equalize::cfg_t::cfg_t (`  
    `const cfg_t & )` [delete]

**5.105.1.3 ~cfg\_t()** `cfg_t::~cfg_t ( )`

#### 5.105.2 Member Function Documentation

---

**5.105.2.1 operator=()**    `cfg_t& equalize::cfg_t::operator= ( const cfg_t & ) [delete]`

### 5.105.3 Member Data Documentation

**5.105.3.1 num\_bins**    `int equalize::cfg_t::num_bins`

**5.105.3.2 nchannels**    `int equalize::cfg_t::nchannels`

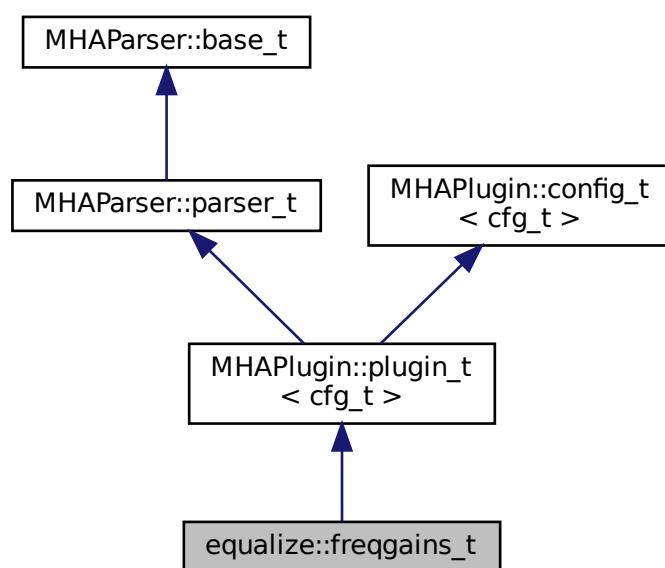
**5.105.3.3 fftgains**    `mha_real_t* equalize::cfg_t::fftgains`

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

- **equalize.cpp**

## 5.106 equalize::freqgains\_t Class Reference

Inheritance diagram for equalize::freqgains\_t:



## Public Member Functions

- `freqgains_t ( algo_comm_t iac, const std::string &configured_name)`
- `mha_spec_t * process ( mha_spec_t *)`
- `void prepare ( mhaconfig_t &)`

## Private Member Functions

- `void update_gains ()`
- `void update_id ()`

## Private Attributes

- `MHAParser::mfloat_t fftgains`
- `MHAParser::string_t id`
- `MHAEvents::patchbay_t< freqgains_t > patchbay`

## Additional Inherited Members

### 5.106.1 Constructor & Destructor Documentation

```
5.106.1.1 freqgains_t() equalize::freqgains_t::freqgains_t (
    algo_comm_t iac,
    const std::string & configured_name )
```

### 5.106.2 Member Function Documentation

```
5.106.2.1 process() mha_spec_t * equalize::freqgains_t::process (
    mha_spec_t * s )
```

**5.106.2.2 `prepare()`** `void equalize::freqgains_t::prepare ( mhaconfig_t & tf ) [virtual]`

Implements `MHAPlugin::plugin_t< cfg_t >` (p. 1149).

**5.106.2.3 `update_gains()`** `void equalize::freqgains_t::update_gains ( ) [private]`

**5.106.2.4 `update_id()`** `void equalize::freqgains_t::update_id ( ) [private]`

### 5.106.3 Member Data Documentation

**5.106.3.1 `fftgains`** `MHAParser::mfloat_t equalize::freqgains_t::fftgains [private]`

**5.106.3.2 `id`** `MHAParser::string_t equalize::freqgains_t::id [private]`

**5.106.3.3 `patchbay`** `MHAEEvents::patchbay_t< freqgains_t > equalize::freqgains_t::patchbay [private]`

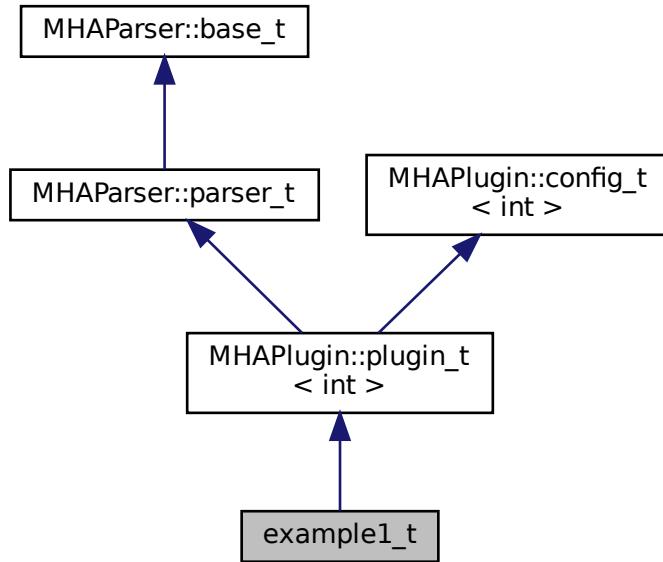
The documentation for this class was generated from the following file:

- `equalize.cpp`

## 5.107 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 iac, const std::string &configured\_name)**  
*Do-nothing constructor.*
- void **release (void)**  
*Release may be empty.*
- void **prepare ( mhaconfig\_t &signal\_info)**  
*Plugin preparation.*
- **mha\_wave\_t \* process ( mha\_wave\_t \*signal)**  
*Signal processing performed by the plugin.*

### Additional Inherited Members

#### 5.107.1 Detailed Description

This C++ class implements the simplest example plugin for the step-by-step tutorial.

It inherits from **MHAParser::parser\_t** (p. 1147) for correct integration in the configuration language interface.

## 5.107.2 Constructor & Destructor Documentation

```
5.107.2.1 example1_t() example1_t::example1_t (
    algo_comm_t iac,
    const std::string & configured_name ) [inline]
```

Do-nothing constructor.

The constructor has to take these two arguments, but it does not have to use them. The base class has to be initialized.

## 5.107.3 Member Function Documentation

```
5.107.3.1 release() void example1_t::release (
    void ) [inline], [virtual]
```

Release may be empty.

Reimplemented from **MHAPlugin::plugin\_t< int >** (p. 1150).

```
5.107.3.2 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

<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. 1149).

**5.107.3.3 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)

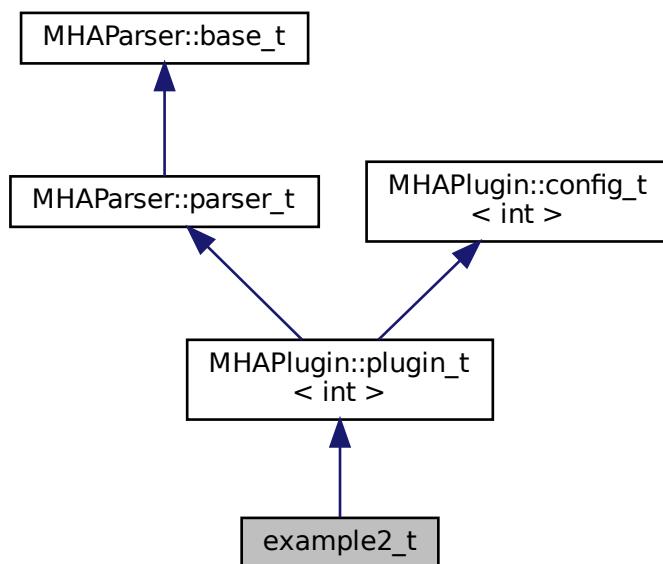
The documentation for this class was generated from the following file:

- `example1.cpp`

## 5.108 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 iac, const std::string &configured\_name)**  
*This constructor initializes the configuration language variables and inserts them into the openMHA configuration tree.*
- **void prepare ( mhaconfig\_t &signal\_info)**  
*Plugin preparation.*
- **void release (void)**  
*Undo restrictions posed in prepare.*
- **mha\_wave\_t \* process ( mha\_wave\_t \*signal)**  
*Signal processing performed by the plugin.*

## Private Attributes

- **MHAParser::int\_t scale\_ch**  
*Index of audio channel to scale.*
- **MHAParser::float\_t factor**  
*The scaling factor applied to the selected channel.*

## Additional Inherited Members

### 5.108.1 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.

### 5.108.2 Constructor & Destructor Documentation

#### 5.108.2.1 **example2\_t()** example2\_t::example2\_t (

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

This constructor initializes the configuration language variables and inserts them into the openMHA configuration tree.

### 5.108.3 Member Function Documentation

**5.108.3.1 `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. 1149).

```
5.108.3.2 release() void example2_t::release (
    void ) [virtual]
```

Undo restrictions posed in prepare.

Reimplemented from **MHAPlugIn::plugin\_t< int >** (p. 1150).

```
5.108.3.3 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)

## 5.108.4 Member Data Documentation

```
5.108.4.1 scale_ch MHAParser::int_t example2_t::scale_ch [private]
```

Index of audio channel to scale.

#### 5.108.4.2 **factor** `MHAParser::float_t example2_t::factor [private]`

The scaling factor applied to the selected channel.

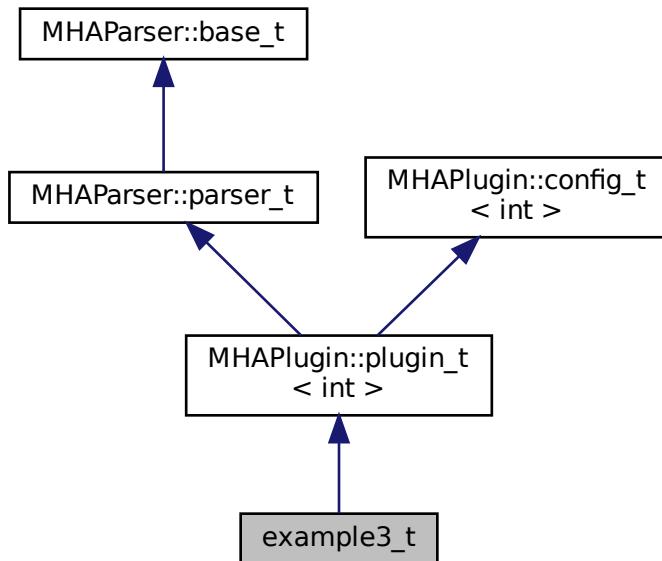
The documentation for this class was generated from the following file:

- **example2.cpp**

## 5.109 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 iac, const std::string &configured\_name)**  
*This constructor initializes the configuration language variables and inserts them into the openMHA configuration tree.*
- **void prepare ( mhaconfig\_t &signal\_info)**  
*Plugin preparation.*
- **void release (void)**  
*Bookkeeping only.*
- **mha\_wave\_t \* process ( mha\_wave\_t \*signal)**  
*Signal processing performed by the plugin.*

## Private Member Functions

- void **on\_scale\_ch\_writeaccess ()**
- void **on\_scale\_ch\_valuechanged ()**
- void **on\_scale\_ch\_readaccess ()**
- void **on\_prereadaccess ()**

## Private Attributes

- **MHAParser::int\_t scale\_ch**  
*Index of audio channel to scale.*
- **MHAParser::float\_t factor**  
*The scaling factor applied to the selected channel.*
- **MHAParser::int\_mon\_t prepared**  
*Keep Track of the prepare/release calls.*
- **MHAEvents::patchbay\_t< example3\_t > patchbay**  
*The Event connector.*

## Additional Inherited Members

### 5.109.1 Detailed Description

A Plugin class using the openMHA Event mechanism.

This is the third example plugin for the step-by-step tutorial.

### 5.109.2 Constructor & Destructor Documentation

#### 5.109.2.1 **example3\_t()** `example3_t::example3_t (`

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

This constructor initializes the configuration language variables and inserts them into the openMHA configuration tree.

It connects the openMHA Events triggered by these configuration variables to the respective callbacks.

### 5.109.3 Member Function Documentation

**5.109.3.1 on\_scale\_ch\_writeaccess()** void example3\_t::on\_scale\_ch\_writeaccess ( )  
[private]

**5.109.3.2 on\_scale\_ch\_valuechanged()** void example3\_t::on\_scale\_ch\_valuechanged ( )  
[private]

**5.109.3.3 on\_scale\_ch\_readaccess()** void example3\_t::on\_scale\_ch\_readaccess ( )  
[private]

**5.109.3.4 on\_prereadaccess()** void example3\_t::on\_prereadaccess ( ) [private]

**5.109.3.5 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

<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. [1149](#)).

**5.109.3.6 `release()`** `void example3_t::release (`  
`void ) [virtual]`

Bookkeeping only.

Reimplemented from **MHAPlugIn::plugin\_t< int >** (p. 1150).

**5.109.3.7 `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)

## 5.109.4 Member Data Documentation

**5.109.4.1 `scale_ch`** `MHAParser::int_t example3_t::scale_ch [private]`

Index of audio channel to scale.

**5.109.4.2 `factor`** `MHAParser::float_t example3_t::factor [private]`

The scaling factor applied to the selected channel.

**5.109.4.3 prepared** `MHAParser::int_mon_t example3_t::prepared [private]`

Keep Track of the prepare/release calls.

**5.109.4.4 patchbay** `MHAEvents::patchbay_t< example3_t> example3_t::patchbay [private]`

The Event connector.

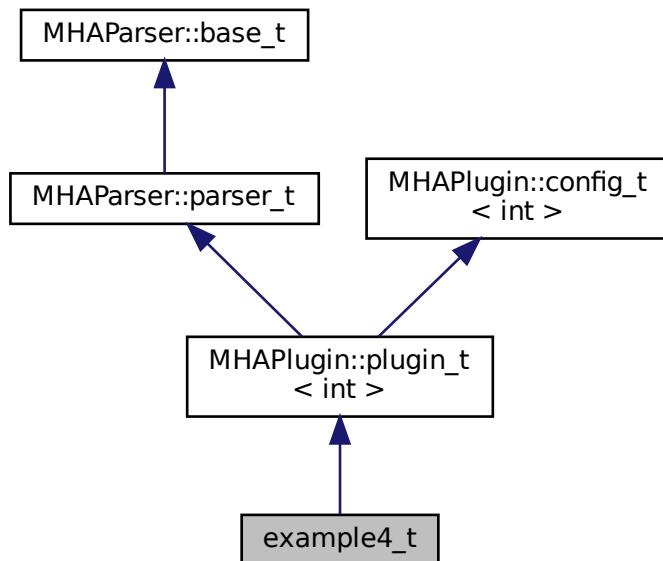
The documentation for this class was generated from the following file:

- `example3.cpp`

**5.110 example4\_t Class Reference**

A Plugin class using the spectral signal.

Inheritance diagram for example4\_t:



## Public Member Functions

- **example4\_t ( algo\_comm\_t iac, const std::string &configured\_name)**  
*This constructor initializes the configuration language variables and inserts them into the openMHA configuration tree.*
- **void prepare ( mhaconfig\_t &signal\_info)**  
*Plugin preparation.*
- **void release (void)**  
*Bookkeeping only.*
- **mha\_spec\_t \* process ( mha\_spec\_t \*signal)**  
*Signal processing performed by the plugin.*

## Private Member Functions

- **void on\_scale\_ch\_writeaccess ()**
- **void on\_scale\_ch\_valuechanged ()**
- **void on\_scale\_ch\_readaccess ()**
- **void on\_prereadaccess ()**

## Private Attributes

- **MHAParser::int\_t scale\_ch**  
*Index of audio channel to scale.*
- **MHAParser::float\_t factor**  
*The scaling factor applied to the selected channel.*
- **MHAParser::int\_mon\_t prepared**  
*Keep Track of the prepare/release calls.*
- **MHAEvents::patchbay\_t< example4\_t > patchbay**  
*The Event connector.*

## Additional Inherited Members

### 5.110.1 Detailed Description

A Plugin class using the spectral signal.

This is the fourth example plugin for the step-by-step tutorial.

### 5.110.2 Constructor & Destructor Documentation

```
5.110.2.1 example4_t() example4_t::example4_t (
    algo_comm_t iac,
    const std::string & configured_name )
```

This constructor initializes the configuration language variables and inserts them into the openMHA configuration tree.

It connects the openMHA Events triggered by these configuration variables to the respective callbacks.

### 5.110.3 Member Function Documentation

```
5.110.3.1 on_scale_ch_writeaccess() void example4_t::on_scale_ch_writeaccess ( )
[private]
```

```
5.110.3.2 on_scale_ch_valuechanged() void example4_t::on_scale_ch_valuechanged ( )
) [private]
```

```
5.110.3.3 on_scale_ch_readaccess() void example4_t::on_scale_ch_readaccess ( )
[private]
```

```
5.110.3.4 on_prereadaccess() void example4_t::on_prereadaccess ( ) [private]
```

```
5.110.3.5 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**

<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. 1149).

**5.110.3.6 release()** `void example4_t::release ( void ) [virtual]`

Bookkeeping only.

Reimplemented from **MHAPlugIn::plugin\_t< int >** (p. 1150).

**5.110.3.7 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 a the signal modified by this plugin.  
(In-place processing)

**5.110.4 Member Data Documentation**

**5.110.4.1 scale\_ch** `MHAParser::int_t example4_t::scale_ch [private]`

Index of audio channel to scale.

**5.110.4.2 factor** `MHAParser::float_t example4_t::factor [private]`

The scaling factor applied to the selected channel.

**5.110.4.3 prepared** `MHAParser::int_mon_t example4_t::prepared [private]`

Keep Track of the prepare/release calls.

**5.110.4.4 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`

## 5.111 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`

### 5.111.1 Constructor & Destructor Documentation

**5.111.1.1 example5\_t()** `example5_t::example5_t (`  
`unsigned int ichannel,`  
`unsigned int numchannels,`  
`mha_real_t iscale )`

## 5.111.2 Member Function Documentation

**5.111.2.1 process()** `mha_spec_t * example5_t::process (mha_spec_t * spec )`

## 5.111.3 Member Data Documentation

**5.111.3.1 channel** `unsigned int example5_t::channel [private]`

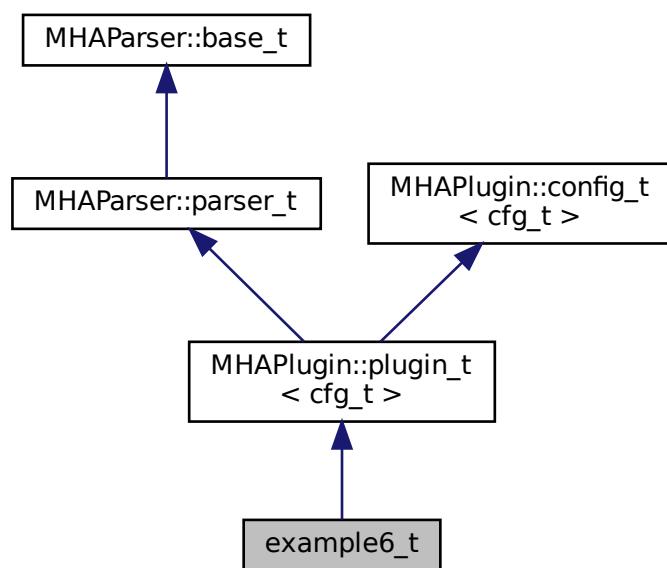
**5.111.3.2 scale** `mha_real_t example5_t::scale [private]`

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

- **example5.cpp**

## 5.112 example6\_t Class Reference

Inheritance diagram for example6\_t:



## Public Member Functions

- `example6_t ( algo_comm_t iac, const std::string &configured_name)`
- `mha_wave_t * process ( mha_wave_t *)`
- `void prepare ( mhaconfig_t &)`

## Private Member Functions

- `void update_cfg ()`

## Private Attributes

- `MHAParser::int_t channel_no`
- `float rmsdb`
- `MHAEvents::patchbay_t< example6_t > patchbay`

## Additional Inherited Members

### 5.112.1 Constructor & Destructor Documentation

```
5.112.1.1 example6() example6_t::example6_t (
    algo_comm_t iac,
    const std::string & configured_name )
```

### 5.112.2 Member Function Documentation

```
5.112.2.1 process() mha_wave_t * example6_t::process (
    mha_wave_t * wave )
```

**5.112.2.2 `prepare()`** `void example6_t::prepare ( mhaconfig_t & tfcfg ) [virtual]`

Implements `MHAPlugIn::plugin_t< cfg_t >` (p. 1149).

**5.112.2.3 `update_cfg()`** `void example6_t::update_cfg ( ) [private]`

### 5.112.3 Member Data Documentation

**5.112.3.1 `channel_no`** `MHAParser::int_t example6_t::channel_no [private]`

**5.112.3.2 `rmsdb`** `float example6_t::rmsdb [private]`

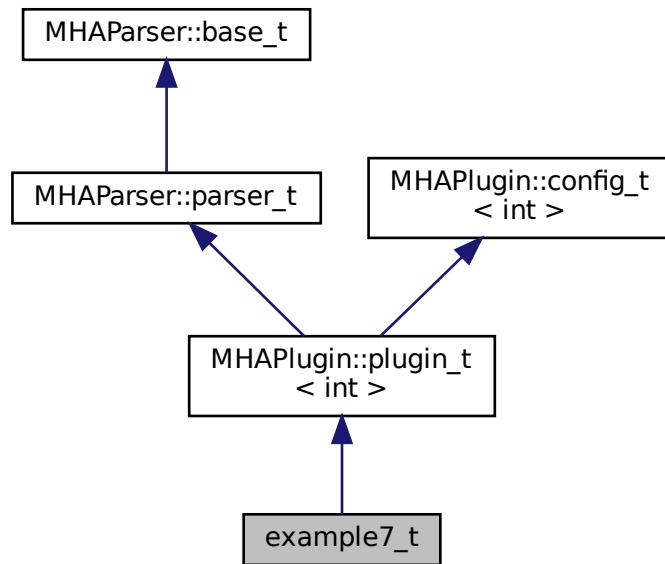
**5.112.3.3 `patchbay`** `MHAEvents::patchbay_t< example6_t > example6_t::patchbay [private]`

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

- `example6.cpp`

## 5.113 example7\_t Class Reference

Inheritance diagram for example7\_t:



### Public Member Functions

- `example7_t ( algo_comm_t iac, const std::string &configured_name)`
- `void release (void)`
- `void prepare ( mhaconfig_t &)`
- `mha_wave_t * process ( mha_wave_t *)`

### Additional Inherited Members

#### 5.113.1 Constructor & Destructor Documentation

**5.113.1.1 `example7_t()`** `example7_t::example7_t (`  
    `algo_comm_t iac,`  
    `const std::string & configured_name )`

## 5.113.2 Member Function Documentation

**5.113.2.1 `release()`** `void example7_t::release ( void ) [virtual]`

Reimplemented from **MHAPlugIn::plugin\_t< int >** (p. 1150).

**5.113.2.2 `prepare()`** `void example7_t::prepare ( mhaconfig_t & signal_info ) [virtual]`

Implements **MHAPlugIn::plugin\_t< int >** (p. 1149).

**5.113.2.3 `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**

## 5.114 expression\_t Class Reference

Class for separating a string into a left hand value and a right hand value.

### 5.114.1 Detailed Description

Class for separating a string into a left hand value and a right hand value.

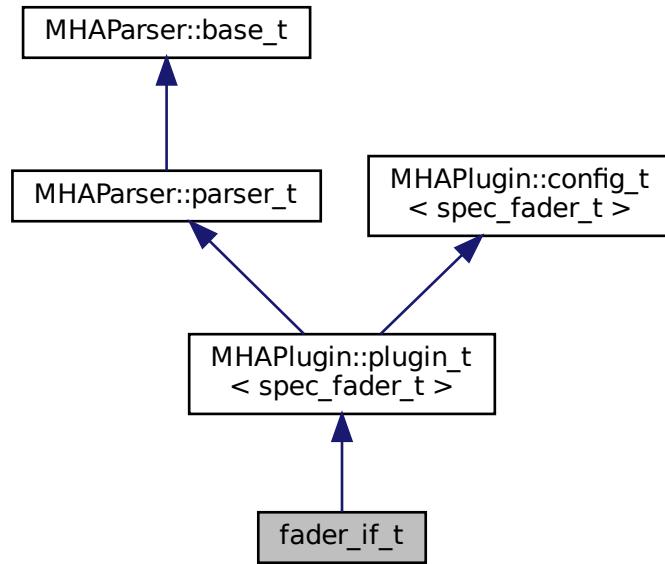
A list of valid operators can be provided. After construction, the class members lval, rval and op contain the appropriate contents.

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

- **mha\_parser.cpp**

## 5.115 fader\_if\_t Class Reference

Inheritance diagram for fader\_if\_t:



### Public Member Functions

- `fader_if_t ( algo_comm_t iac, const std::string &configured_name)`
- `mha_spec_t * process ( mha_spec_t *)`
- `void prepare ( mhaconfig_t &)`

### Private Member Functions

- `void update_cfg ()`

### Private Attributes

- `MHAEvents::patchbay_t< fader_if_t > patchbay`
- `MHParse::float_t tau`
- `MHParse::vfloat_t newgains`
- `mha_real_t * actgains`

## Additional Inherited Members

### 5.115.1 Constructor & Destructor Documentation

```
5.115.1.1 fader_if_t() fader_if_t::fader_if_t (
    algo_comm_t iac,
    const std::string & configured_name )
```

### 5.115.2 Member Function Documentation

```
5.115.2.1 process() mha_spec_t * fader_if_t::process (
    mha_spec_t * s )
```

```
5.115.2.2 prepare() void fader_if_t::prepare (
    mhaconfig_t & tf ) [virtual]
```

Implements **MHAPlugin::plugin\_t< spec\_fader\_t >** (p. [1149](#)).

```
5.115.2.3 update_cfg() void fader_if_t::update_cfg ( ) [private]
```

### 5.115.3 Member Data Documentation

```
5.115.3.1 patchbay MHAEvents::patchbay_t< fader_if_t > fader_if_t::patchbay [private]
```

**5.115.3.2 tau** `MHAParser::float_t` `fader_if_t::tau` [private]

**5.115.3.3 newgains** `MHAParser::vfloat_t` `fader_if_t::newgains` [private]

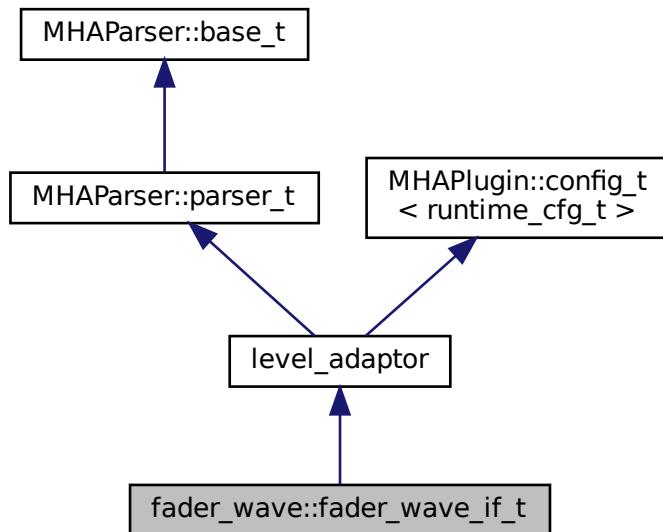
**5.115.3.4 actgains** `mha_real_t*` `fader_if_t::actgains` [private]

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

- `fader_spec.cpp`

## 5.116 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 iac, const std::string &configured_name)`
- `mha_wave_t * process ( mha_wave_t *)`
- `void prepare ( mhaconfig_t &)`
- `void release ()`

## Private Member Functions

- void **set\_level()**

## Private Attributes

- **MHAParser::vfloat\_t gain**
- **MHAParser::float\_t ramplen**
- **MHAEvents::patchbay\_t< fader\_wave\_if\_t > patchbay**
- bool **prepared**

## Additional Inherited Members

### 5.116.1 Constructor & Destructor Documentation

```
5.116.1.1 fader_wave_if_t() fader_wave::fader_wave_if_t::fader_wave_if_t (
    algo_comm_t iac,
    const std::string & configured_name )
```

### 5.116.2 Member Function Documentation

```
5.116.2.1 process() mha_wave_t * fader_wave::fader_wave_if_t::process (
    mha_wave_t * s )
```

```
5.116.2.2 prepare() void fader_wave::fader_wave_if_t::prepare (
    mhaconfig_t & tf ) [virtual]
```

Implements **MHAPlugin::plugin\_t< runtime\_cfg\_t >** (p. [1149](#)).

**5.116.2.3 release()** void fader\_wave::fader\_wave\_if\_t::release ( ) [virtual]

Reimplemented from **MHAPlugin::plugin\_t< runtime\_cfg\_t >** (p. 1150).

**5.116.2.4 set\_level()** void fader\_wave::fader\_wave\_if\_t::set\_level ( ) [private]

### 5.116.3 Member Data Documentation

**5.116.3.1 gain** **MHAParser::vfloat\_t** fader\_wave::fader\_wave\_if\_t::gain [private]

**5.116.3.2 ramplen** **MHAParser::float\_t** fader\_wave::fader\_wave\_if\_t::ramplen [private]

**5.116.3.3 patchbay** **MHAEvents::patchbay\_t< fader\_wave\_if\_t >** fader\_wave::fader\_wave\_if\_t::patchbay [private]

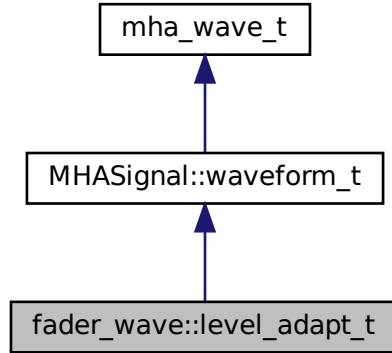
**5.116.3.4 prepared** bool fader\_wave::fader\_wave\_if\_t::prepared [private]

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

- **fader\_wave.cpp**

## 5.117 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 > **I\_new\_**, std::vector< float > **I\_old\_**)
- void **update\_frame** ()
- std::vector< float > **get\_level** () const
- bool **can\_update** () const

### Private Attributes

- unsigned int **llen**
- unsigned int **pos**
- **MHAWindow::fun\_t** **wnd**
- std::vector< float > **I\_new**
- std::vector< float > **I\_old**

### Additional Inherited Members

#### 5.117.1 Constructor & Destructor Documentation

```
5.117.1.1 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 )
```

## 5.117.2 Member Function Documentation

**5.117.2.1 update\_frame()** void fader\_wave::level\_adapt\_t::update\_frame ( )

**5.117.2.2 get\_level()** std::vector<float> fader\_wave::level\_adapt\_t::get\_level ( )
const [inline]

**5.117.2.3 can\_update()** bool fader\_wave::level\_adapt\_t::can\_update ( ) const [inline]

## 5.117.3 Member Data Documentation

**5.117.3.1 ilen** unsigned int fader\_wave::level\_adapt\_t::ilen [private]

**5.117.3.2 pos** unsigned int fader\_wave::level\_adapt\_t::pos [private]

**5.117.3.3 wnd** MHAWindow::fun\_t fader\_wave::level\_adapt\_t::wnd [private]

### 5.117.3.4 **l\_new** std::vector<float> fader\_wave::level\_adapt\_t::l\_new [private]

### 5.117.3.5 **l\_old** std::vector<float> fader\_wave::level\_adapt\_t::l\_old [private]

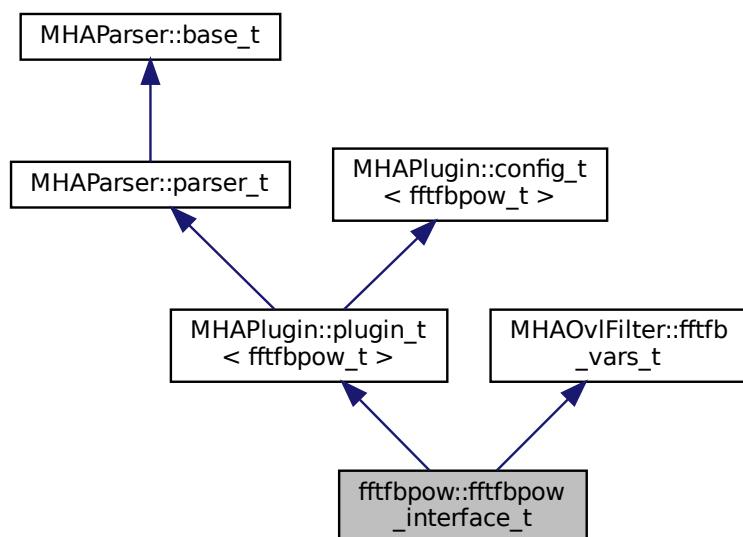
The documentation for this class was generated from the following file:

- **fader\_wave.cpp**

## 5.118 fftfbpow::fftfbpow\_interface\_t Class Reference

Interface class for fftfbpow plugin.

Inheritance diagram for fftfbpow::fftfbpow\_interface\_t:



### Public Member Functions

- **fftfbpow\_interface\_t ( algo\_comm\_t iac, const std::string &configured\_name)**  
*Constructor with standard MHA constructor parameters.*
- **void prepare ( mhaconfig\_t &tf)**  
*Standard MHA plugin prepare function.*
- **mha\_spec\_t \* process ( mha\_spec\_t \*s)**  
*Standard MHA plugin process fct.*

## Private Member Functions

- void **update\_cfg ()**  
*Constructs new runtime configuration in thread-safe manner.*

## Private Attributes

- std::string **name**  
*Configured name of this plugin instance.*
- **MHAEvents::patchbay\_t< fftfbpow\_interface\_t > patchbay**  
*Patchbay to connect to MHA configuration interface.*

## Additional Inherited Members

### 5.118.1 Detailed Description

Interface class for fftfbpow plugin.

### 5.118.2 Constructor & Destructor Documentation

```
5.118.2.1 fftfbpow_interface_t() fftfbpow::fftfbpow_interface_t::fftfbpow_interface_t( algo_comm_t iac, const std::string & configured_name )
```

Constructor with standard MHA constructor parameters.

#### Parameters

<i>iac</i>	Handle to algorithm communication variable space
<i>configured_name</i>	Configured name of this plugin instance

### 5.118.3 Member Function Documentation

---

**5.118.3.1 `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. 492) 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. 1149).

**5.118.3.2 `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. 492) of the runtime configuration.

#### Parameters

<code>s</code>	Input spectrum
----------------	----------------

#### Returns

Unchanged input spectrum

**5.118.3.3 `update_cfg()`** `void fftfbpow::fftfbpow_interface_t::update_cfg ( ) [private]`

Constructs new runtime configuration in thread-safe manner.

## 5.118.4 Member Data Documentation

**5.118.4.1 name** std::string fftfbpow::fftfbpow\_interface\_t::name [private]

Configured name of this plugin instance.

**5.118.4.2 patchbay** MHAEvents::patchbay\_t< fftfbpow\_interface\_t> fftfbpow::fftfbpow<->\_interface\_t::patchbay [private]

Patchbay to connect to MHA configuration interface.

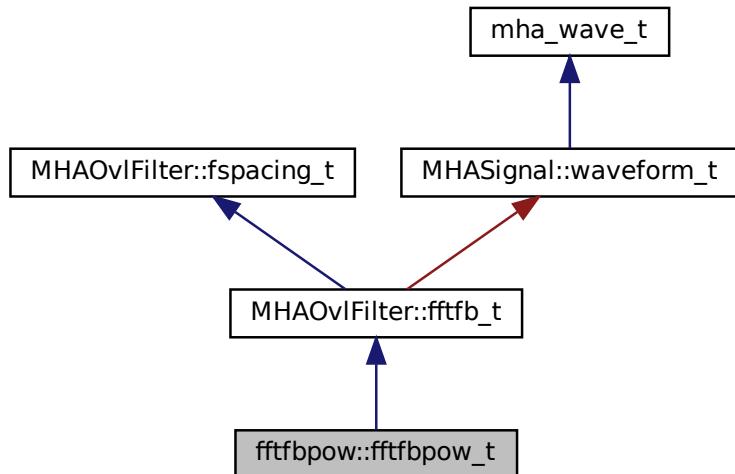
The documentation for this class was generated from the following file:

- **fftfbpow.cpp**

**5.119 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

### 5.119.1 Detailed Description

Run time configuration for the fftfbpow plugin.

### 5.119.2 Constructor & Destructor Documentation

```
5.119.2.1 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

### 5.119.3 Member Data Documentation

### 5.119.3.1 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**

## 5.120 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.*

### 5.120.1 Detailed Description

Run-time configuration class for the fftfilter MHA plugin.

### 5.120.2 Constructor & Destructor Documentation

#### 5.120.2.1 **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> <sub>←</sub>	The block size (samples per channel) for waveform audio data
<i>channels</i> <sub>←</sub>	The number of prepared audio channels for this MHA plugin.
<i>fftlens</i> <sub>_</sub>	FFT length used for filtering

## 5.120.3 Member Function Documentation

**5.120.3.1 process()** `mha_wave_t * fftfilter::fftfilter_t::process ( mha_wave_t * s )`

Let fftfiler object handle the filtering

## 5.120.4 Member Data Documentation

**5.120.4.1 irslen** `unsigned int fftfilter::fftfilter_t::irslen [private]`

Length of the longest impulse response applied.

**5.120.4.2 fragsize** `unsigned int fftfilter::fftfilter_t::fragsize [private]`

The block size (samples per channel) for waveform audio data.

**5.120.4.3 fftlen** `unsigned int fftfilter::fftfilter_t::fftlens [private]`

FFT length used for filtering.

**5.120.4.4 channels** `unsigned int fftfilter::fftfilter_t::channels [private]`

Number of prepared audio channels processed by this MHA plugin.

**5.120.4.5 fftfilt** `MHAFilter::fftfilter_t fftfilter::fftfilter_t::fftfilt [private]`

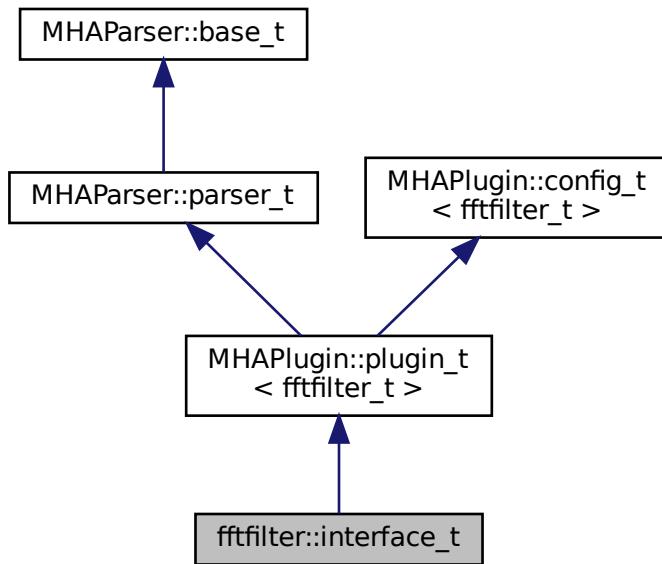
The filter object.

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

- `fftfilter.cpp`

**5.121 fftfilter::interface\_t Class Reference**

Inheritance diagram for fftfilter::interface\_t:

**Public Member Functions**

- `interface_t ( algo_comm_t iac, const std::string &configured_name )`
- `mha_wave_t * process ( mha_wave_t * )`
- `void prepare ( mhaconfig_t & )`

## Private Member Functions

- void **update ()**

## Private Attributes

- **MHAParser::mfloat\_t irs**
- **MHAParser::int\_t fftlen**
- **MHAParser::int\_mon\_t fftlen\_final**
- **MHAEvents::patchbay\_t< interface\_t > patchbay**

## Additional Inherited Members

### 5.121.1 Detailed Description

Implements the MHA plugin interface for FFTFilter

### 5.121.2 Constructor & Destructor Documentation

```
5.121.2.1 interface_t() fftfilter::interface_t::interface_t (
    algo_comm_t iac,
    const std::string & configured_name )
```

### 5.121.3 Member Function Documentation

```
5.121.3.1 process() mha_wave_t * fftfilter::interface_t::process (
    mha_wave_t * s )
```

**5.121.3.2 `prepare()`** void fftfilter::interface\_t::prepare (   
   mhaconfig\_t & tf ) [virtual]

Implements **MHAPlugIn::plugin\_t< fftfilter\_t >** (p. 1149).

**5.121.3.3 `update()`** void fftfilter::interface\_t::update ( ) [private]

#### 5.121.4 Member Data Documentation

**5.121.4.1 `irs`** MHAParser::mfloat\_t fftfilter::interface\_t::irs [private]

**5.121.4.2 `fftlen`** MHAParser::int\_t fftfilter::interface\_t::fftlen [private]

**5.121.4.3 `fftlen_final`** MHAParser::int\_mon\_t fftfilter::interface\_t::fftlen\_final [private]

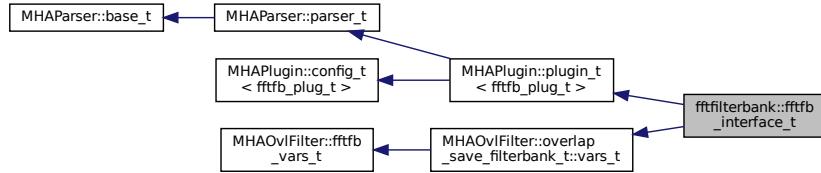
**5.121.4.4 `patchbay`** MHAEvents::patchbay\_t< interface\_t > fftfilter::interface\_t::patchbay [private]

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

- **fftfilter.cpp**

## 5.122 fftfilterbank::fftfb\_interface\_t Class Reference

Inheritance diagram for fftfilterbank::fftfb\_interface\_t:



### Public Member Functions

- **fftfb\_interface\_t ( algo\_comm\_t iac, const std::string &configured\_name)**  
*Default values are set and MHA configuration variables registered into the parser.*
- void **prepare ( mhaconfig\_t &)**  
*Prepare all variables for processing.*
- void **release ()**
- **mha\_spec\_t \* process ( mha\_spec\_t \*)**
- **mha\_wave\_t \* process ( mha\_wave\_t \*)**

### Private Member Functions

- void **update\_cfg ()**

### Private Attributes

- **MHAParser::bool\_t return\_imag**
- **MHAEvents::patchbay\_t< fftfb\_interface\_t > patchbay**
- **MHA\_AC::int\_t nchannels**
- std::string **algo**
- bool **prepared**
- unsigned int **nbands**

### Additional Inherited Members

#### 5.122.1 Constructor & Destructor Documentation

**5.122.1.1 fftfb\_interface\_t()** fftfilterbank::fftfb\_interface\_t::fftfb\_interface\_t (   
     **algo\_comm\_t** *iac*,  
     const std::string & *configured\_name* )

Default values are set and MHA configuration variables registered into the parser.

### Parameters

<i>ac</i>	algorithm communication handle
<i>th</i>	chain name
<i>al</i>	algorithm name

## 5.122.2 Member Function Documentation

**5.122.2.1 `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

$$W(f) = W(T(S(f))) = W(x), \quad x = T(S(f)) = T(\hat{f}),$$

$W(x)$  beeing a symmetric window function in the interval  $[-1, 1]$  and  $S(f)$  the transformation from the linear scale to the given frequency scale (see functions in FreqScaleFun). The function  $T(\hat{f})$  transforms the frequency range between the center frequencies  $[\hat{f}_{k-1}, \hat{f}_k]$  and  $[\hat{f}_k, \hat{f}_{k+1}]$  into the interval  $[-1, 0]$  and  $[0, 1]$ , respectively. This function is realised by the function linscale().

### Parameters

<i>tf</i>	Channel configuration
-----------	-----------------------

Implements **MHAPlugIn::plugin\_t< fftfb\_plug\_t >** (p. 1149).

**5.122.2.2 `release()`** `void fftfilterbank::fftfb_interface_t::release ( ) [virtual]`

Reimplemented from **MHAPlugIn::plugin\_t< fftfb\_plug\_t >** (p. 1150).

**5.122.2.3 process()** [1/2] `mha_spec_t * fftfilterbank::fftfb_interface_t::process ( mha_spec_t * s )`

**5.122.2.4 process()** [2/2] `mha_wave_t * fftfilterbank::fftfb_interface_t::process ( mha_wave_t * s )`

**5.122.2.5 update\_cfg()** `void fftfilterbank::fftfb_interface_t::update_cfg ( )` [private]

### 5.122.3 Member Data Documentation

**5.122.3.1 return\_imag** `MHAParser::bool_t fftfilterbank::fftfb_interface_t::return_imag` [private]

**5.122.3.2 patchbay** `MHAEvents::patchbay_t< fftfb_interface_t> fftfilterbank::fftfb_interface_t::patchbay` [private]

**5.122.3.3 nchannels** `MHA_AC::int_t fftfilterbank::fftfb_interface_t::nchannels` [private]

**5.122.3.4 algo** `std::string fftfilterbank::fftfb_interface_t::algo` [private]

**5.122.3.5 prepared** `bool fftfilterbank::fftfb_interface_t::prepared` [private]

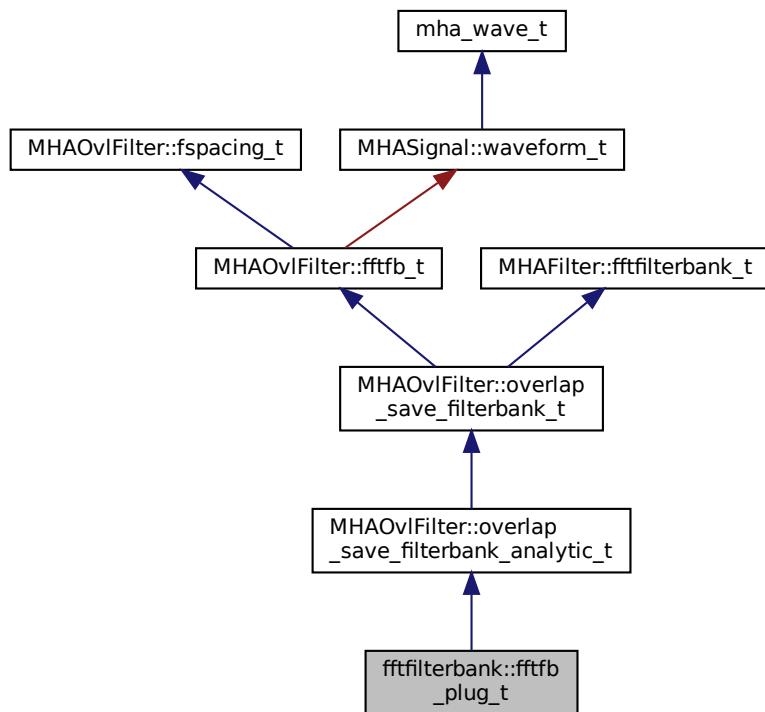
### 5.122.3.6 nbands `unsigned int fftfilterbank::fftfb_interface_t::nbands` [private]

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

- `fftfilterbank.cpp`

## 5.123 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

### 5.123.1 Constructor & Destructor Documentation

**5.123.1.1 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 )

### 5.123.2 Member Function Documentation

**5.123.2.1 process() [1/2]** mha\_spec\_t \* fftfilterbank::fftfb\_plug\_t::process ( mha\_spec\_t \* s )

**5.123.2.2 process() [2/2]** mha\_wave\_t \* fftfilterbank::fftfb\_plug\_t::process ( mha\_wave\_t \* s )

**5.123.2.3 insert()** void fftfilterbank::fftfb\_plug\_t::insert ( )

### 5.123.3 Member Data Documentation

**5.123.3.1 fb\_acinfo** `MHAOvlFilter::fftfb_ac_info_t fftfilterbank::fftfb_plug_t::fb_acinfo` [private]

**5.123.3.2 s\_out** `MHASignal::spectrum_t fftfilterbank::fftfb_plug_t::s_out` [private]

**5.123.3.3 imag** `MHA_AC::waveform_t fftfilterbank::fftfb_plug_t::imag` [private]

**5.123.3.4 return\_imag\_** `bool fftfilterbank::fftfb_plug_t::return_imag_` [private]

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

- `fftfilterbank.cpp`

## 5.124 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.*

### 5.124.1 Detailed Description

fshift runtime config class

### 5.124.2 Constructor & Destructor Documentation

**5.124.2.1 `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

<code><i>plug</i></code>	ptr to the plugin interface class. Configuration information is given this way to keep the argument list small.
--------------------------	---

**5.124.2.2 `~fshift_config_t()`** `fshift::fshift_config_t::~fshift_config_t ( ) [default]`

### 5.124.3 Member Function Documentation

**5.124.3.1 process()** `mha_spec_t * fshift::fshift_config_t::process (mha_spec_t * in )`

#### 5.124.4 Member Data Documentation

**5.124.4.1 kmin** `const unsigned int fshift::fshift_config_t::kmin [private]`

FFT bin corresponding to fmin.

**5.124.4.2 kmax** `const unsigned fshift::fshift_config_t::kmax [private]`

FFT bin corresponding to fmax.

**5.124.4.3 df** `const int fshift::fshift_config_t::df [private]`

Frequency shift expressed in FFT bins.

**5.124.4.4 delta\_phi** `const mha_complex_t fshift::fshift_config_t::delta_phi [private]`

Phase advance per fft frame.

**5.124.4.5 delta\_phi\_total** `mha_complex_t fshift::fshift_config_t::delta_phi_total [private]`

Sum of all phase advances.

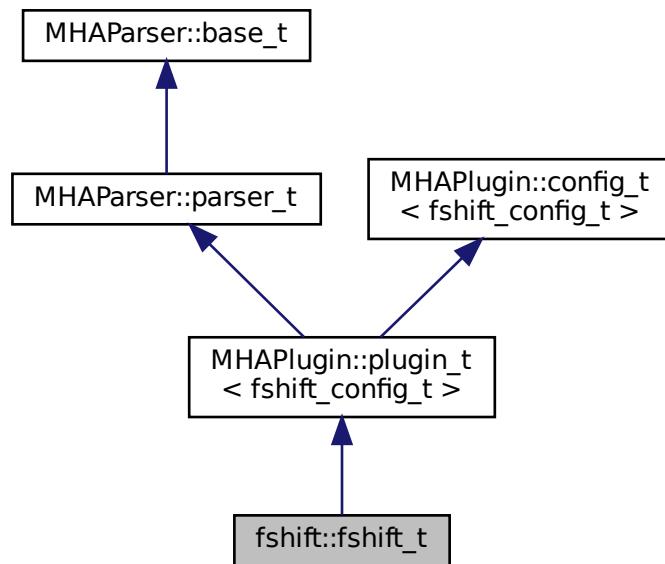
The documentation for this class was generated from the following files:

- **fshift.hh**
- **fshift.cpp**

## 5.125 fshift::fshift\_t Class Reference

fshift plugin interface class

Inheritance diagram for fshift::fshift\_t:



### Public Member Functions

- **`fshift_t ( algo_comm_t iac, const std::string &configured_name)`**  
*Constructs our plugin.*
- **`~fshift_t ()`**
- **`mha_spec_t * process ( mha_spec_t *)`**  
*Checks for the most recent configuration and defers processing to it.*
- **`void prepare ( mhaconfig_t &)`**  
*Plugin preparation.*
- **`void release (void)`**
- **`mha_real_t fmin () const`**
- **`mha_real_t fmax () const`**
- **`mha_real_t df () const`**

### Private Member Functions

- **`void update_cfg ()`**

## Private Attributes

- **MHAEvents::patchbay\_t < fshift\_t > patchbay**  
*patch bay for connecting configuration parser events with local member functions:*
- **MHAParser::float\_t m\_fmin**
- **MHAParser::float\_t m\_fmax**  
*upper boundary for frequency shifter*
- **MHAParser::float\_t m\_df**  
*Shift frequency in Hz.*

## Additional Inherited Members

### 5.125.1 Detailed Description

fshift plugin interface class

### 5.125.2 Constructor & Destructor Documentation

```
5.125.2.1 fshift_t() fshift::fshift_t::fshift_t (
    algo_comm_t iac,
    const std::string & configured_name )
```

Constructs our plugin.

```
5.125.2.2 ~fshift_t() fshift::fshift_t::~fshift_t ( )
```

### 5.125.3 Member Function Documentation

```
5.125.3.1 process() mha_spec_t * fshift::fshift_t::process (
    mha_spec_t * signal )
```

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

```
5.125.3.2 prepare() void fshift::fshift_t::prepare (
    mhacconfig_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. [1149](#)).

**5.125.3.3 release()** `void fshift::fshift_t::release ( void ) [inline], [virtual]`

Reimplemented from **MHAPlugin::plugin\_t< fshift\_config\_t >** (p. [1150](#)).

**5.125.3.4 fmin()** `mha_real_t fshift::fshift_t::fmin ( ) const [inline]`

**5.125.3.5 fmax()** `mha_real_t fshift::fshift_t::fmax ( ) const [inline]`

**5.125.3.6 df()** `mha_real_t fshift::fshift_t::df ( ) const [inline]`

**5.125.3.7 update\_cfg()** `void fshift::fshift_t::update_cfg ( ) [private]`

## 5.125.4 Member Data Documentation

**5.125.4.1 patchbay** `MHAEvents::patchbay_t< fshift_t > fshift::fshift_t::patchbay [private]`

patch bay for connecting configuration parser events with local member functions:

**5.125.4.2 m\_fmin** `MHAParser::float_t fshift::fshift_t::m_fmin [private]`**5.125.4.3 m\_fmax** `MHAParser::float_t fshift::fshift_t::m_fmax [private]`

upper boundary for frequency shifter

**5.125.4.4 m\_df** `MHAParser::float_t fshift::fshift_t::m_df [private]`

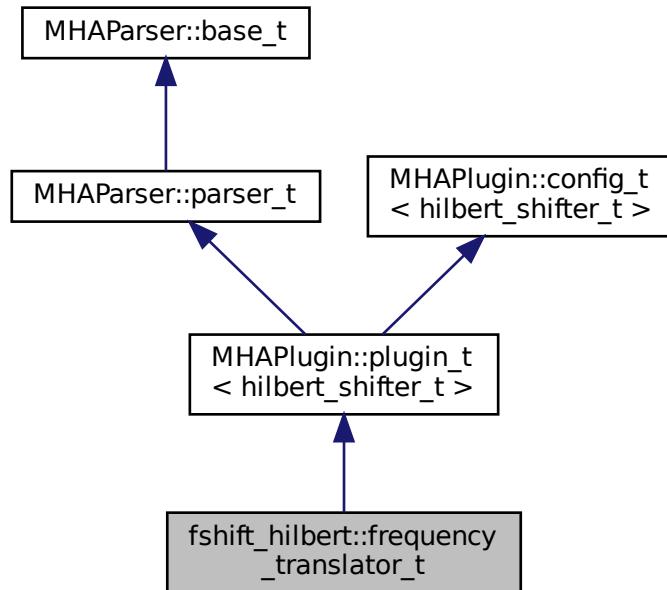
Shift frequency in Hz.

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

- `fshift.hh`
- `fshift.cpp`

**5.126 fshift\_hilbert::frequency\_translator\_t Class Reference**

Inheritance diagram for `fshift_hilbert::frequency_translator_t`:



## Public Member Functions

- **frequency\_translator\_t** (**algo\_comm\_t** iac, const std::string &configured\_name)
- **mha\_spec\_t \* process** (**mha\_spec\_t \***)
- **void prepare** (**mhaconfig\_t** &)
- **void release** ()

## Private Member Functions

- **void update** ()

## Private Attributes

- **MHAEvents::patchbay\_t< frequency\_translator\_t > patchbay**
- **MHAParser::vfloat\_t df**  
*Vector containing the shift frequencies in Hz.*
- **MHAParser::float\_t fmin**  
*Lower boundary for frequency shifter.*
- **MHAParser::float\_t fmax**  
*Upper boundary for frequency shifter.*
- **MHAParser::int\_t irslen**  
*Maximum length of cut off filter response.*
- **MHAParser::kw\_t phasemode**  
*Mode of gain smoothing.*

## Additional Inherited Members

### 5.126.1 Constructor & Destructor Documentation

**5.126.1.1 frequency\_translator\_t()** `fshift_hilbert::frequency_translator_t::frequency<_translator_t (`  
`algo_comm_t iac,`  
`const std::string & configured_name )`

### 5.126.2 Member Function Documentation

**5.126.2.1 process()** `mha_spec_t * fshift_hilbert::frequency_translator_t::process ( mha_spec_t * s )`

**5.126.2.2 prepare()** `void fshift_hilbert::frequency_translator_t::prepare ( mhaconfig_t & tf ) [virtual]`

Implements **MHAPlugIn::plugin\_t< hilbert\_shifter\_t >** (p. 1149).

**5.126.2.3 release()** `void fshift_hilbert::frequency_translator_t::release ( ) [virtual]`

Reimplemented from **MHAPlugIn::plugin\_t< hilbert\_shifter\_t >** (p. 1150).

**5.126.2.4 update()** `void fshift_hilbert::frequency_translator_t::update ( ) [private]`

### 5.126.3 Member Data Documentation

**5.126.3.1 patchbay** `MHAEvents::patchbay_t< frequency_translator_t > fshift_hilbert->::frequency_translator_t::patchbay [private]`

**5.126.3.2 df** `MHAParser::vfloat_t fshift_hilbert::frequency_translator_t::df [private]`

Vector containing the shift frequencies in Hz.

**5.126.3.3 fmin** `MHAParser::float_t fshift_hilbert::frequency_translator_t::fmin [private]`

Lower boundary for frequency shifter.

**5.126.3.4 fmax** `MHAParser::float_t fshift_hilbert::frequency_translator_t::fmax`  
 [private]

Upper boundary for frequency shifter.

**5.126.3.5 irslen** `MHAParser::int_t fshift_hilbert::frequency_translator_t::irslen`  
 [private]

Maximum length of cut off filter response.

**5.126.3.6 phasemode** `MHAParser::kw_t fshift_hilbert::frequency_translator_t::phasemode`  
 [private]

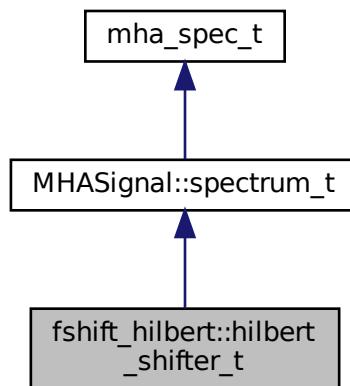
Mode of gain smoothing.

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

- `fshift_hilbert.cpp`

## 5.127 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  $a(t)=x(t)+i*H(x(t))$*
- **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

### 5.127.1 Constructor & Destructor Documentation

---

**5.127.1.1 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 )
```

**5.127.1.2 ~hilbert\_shifter\_t()** fshift\_hilbert::hilbert\_shifter\_t::~hilbert\_shifter\_t ( )

## 5.127.2 Member Function Documentation

**5.127.2.1 process()** void fshift\_hilbert::hilbert\_shifter\_t::process (

```
mha_spec_t * s )
```

## 5.127.3 Member Data Documentation

**5.127.3.1 fullspec** MHASignal::spectrum\_t fshift\_hilbert::hilbert\_shifter\_t::fullspec  
[private]

Part of the spectrum to be frequency shifted.

**5.127.3.2 analytic** MHASignal::spectrum\_t fshift\_hilbert::hilbert\_shifter\_t::analytic  
[private]

Analytic signal, defined as  $a(t) = x(t) + i \cdot H(x(t))$

**5.127.3.3 shifted** `MHASignal::waveform_t fshift_hilbert::hilbert_shifter_t::shifted` [private]

The frequency shifted signal in the time domain.

**5.127.3.4 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

**5.127.3.5 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

**5.127.3.6 plan\_spec2analytic** `fftw_plan fshift_hilbert::hilbert_shifter_t::plan_spec2analytic` [private]

FFT plan for the transformation of fullspec into the time domain.

**5.127.3.7 mhafft** `mha_fft_t fshift_hilbert::hilbert_shifter_t::mhafft` [private]

MHA wrapper object for fftw.

**5.127.3.8 df** `MHASignal::waveform_t fshift_hilbert::hilbert_shifter_t::df` [private]

Vector holding one delta f value for every channel.

**5.127.3.9 kmin** `unsigned int fshift_hilbert::hilbert_shifter_t::kmin [private]`

FFT frame that corresponds to `f_min`.

**5.127.3.10 kmax** `unsigned int fshift_hilbert::hilbert_shifter_t::kmax [private]`

FFT frame that corresponds to `f_max`.

**5.127.3.11 frameshift** `unsigned int fshift_hilbert::hilbert_shifter_t::frameshift [private]`

Total phase advance within one fragment.

**5.127.3.12 delta\_phi** `std::vector< mha_complex_t > fshift_hilbert::hilbert_shifter_t::delta_phi [private]`

Phase advance per frame.

**5.127.3.13 delta\_phi\_total** `std::vector< mha_complex_t > fshift_hilbert::hilbert_shifter_t::delta_phi_total [private]`

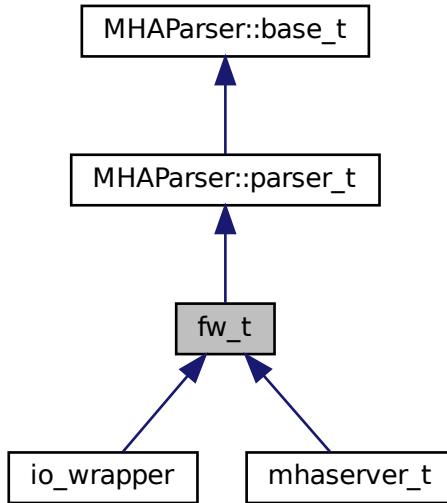
Sum of all phase advances.

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

- `fshift_hilbert.cpp`

## 5.128 fw\_t Class Reference

Inheritance diagram for fw\_t:



### Public Member Functions

- `fw_t()`
- `~fw_t()`
- `bool exit_request() const`

### Protected Attributes

- `io_lib_t * io_lib`
- `int proc_error`
- `int io_error`

### Private Types

- `enum state_t {  
 fw_unprepared, fw_stopped, fw_starting, fw_running,  
 fw_stopping, fw_exiting }`

## Private Member Functions

- **void prepare ()**  
*preparation for processing*
- **void start ()**  
*start of processing*
- **void stop ()**  
*stop/pause of processing*
- **void release ()**  
*release of IO device*
- **void quit ()**  
*controlled quit*
- **void stopped (int, int)**
- **void started ()**
- **int process ( mha\_wave\_t \*, mha\_wave\_t \*\*)**
- **void exec\_fw\_command ()**
- **void load\_proc\_lib ()**
- **void load\_io\_lib ()**
- **void fw\_sleep\_cmd ()**
- **void fw\_until\_cmd ()**
- **void get\_input\_signal\_dimension ()**
- **void async\_read ()**
- **void async\_poll\_msg ()**
- **void get\_parserstate ()**

## Static Private Member Functions

- **static void stopped (void \*h, int proc\_err, int io\_err)**
- **static void started (void \*h)**
- **static int process (void \*h, mha\_wave\_t \*sIn, mha\_wave\_t \*\*sOut)**

## Private Attributes

- **fw\_vars\_t prepare\_vars**
- **MHAParser::int\_mon\_t nchannels\_out**
- **MHAParser::string\_t proc\_name**
- **MHAParser::string\_t io\_name**
- **MHAParser::bool\_t exit\_on\_stop**
- **MHAParser::int\_t fw\_sleep**
- **MHAParser::string\_t fw\_until**
- **MHAParser::kw\_t fw\_cmd**
- **MHAParser::string\_mon\_t parserstate**
- **MHAParser::string\_t errorlog**
- **MHAParser::string\_t fatallog**

- **MHAParser::vstring\_t plugins**
- **MHAParser::vstring\_t plugin\_paths**
- **MHAParser::bool\_t dump\_mha**
- **MHAParser::string\_t inst\_name**  
*A variable for naming MHA instances.*
- **MHAKernel::algo\_comm\_class\_t ac**
- **PluginLoader::mhaplugloader\_t \* proc\_lib**
- **mhaconfig\_t cfin**
- **mhaconfig\_t cfout**
- **state\_t state**
- **bool b\_exit\_request**
- **MHAParser::string\_mon\_t proc\_error\_string**
- **MHAEvents::patchbay\_t< fw\_t > patchbay**

## Additional Inherited Members

### 5.128.1 Member Enumeration Documentation

#### 5.128.1.1 state\_t enum fw\_t::state\_t [private]

##### Enumerator

fw_unprepared	
fw_stopped	
fw_starting	
fw_running	
fw_stopping	
fw_exiting	

### 5.128.2 Constructor & Destructor Documentation

#### 5.128.2.1 fw\_t() fw\_t::fw\_t ( )

**5.128.2.2 ~fw\_t()** `fw_t::~fw_t ( )`

### 5.128.3 Member Function Documentation

**5.128.3.1 exit\_request()** `bool fw_t::exit_request ( ) const [inline]`

**5.128.3.2 prepare()** `void fw_t::prepare ( ) [private]`

preparation for processing

**5.128.3.3 start()** `void fw_t::start ( ) [private]`

start of processing

**5.128.3.4 stop()** `void fw_t::stop ( ) [private]`

stop/pause of processing

**5.128.3.5 release()** `void fw_t::release ( ) [private]`

release of IO device

**5.128.3.6 quit()** `void fw_t::quit ( ) [private]`

controlled quit

**5.128.3.7 stopped() [1/2]** static void fw\_t::stopped ( void \* *h*, int *proc\_err*, int *io\_err* ) [inline], [static], [private]

**5.128.3.8 started() [1/2]** static void fw\_t::started ( void \* *h* ) [inline], [static], [private]

**5.128.3.9 process() [1/2]** static int fw\_t::process ( void \* *h*, mha\_wave\_t \* *sIn*, mha\_wave\_t \*\* *sOut* ) [inline], [static], [private]

**5.128.3.10 stopped() [2/2]** void fw\_t::stopped ( int *proc\_err*, int *io\_err* ) [private]

**5.128.3.11 started() [2/2]** void fw\_t::started ( ) [private]

**5.128.3.12 process() [2/2]** int fw\_t::process ( mha\_wave\_t \* *s\_in*, mha\_wave\_t \*\* *s\_out* ) [private]

**5.128.3.13 exec\_fw\_command()** void fw\_t::exec\_fw\_command ( ) [private]

**5.128.3.14 `load_proc_lib()`** void fw\_t::load\_proc\_lib ( ) [private]

**5.128.3.15 `load_io_lib()`** void fw\_t::load\_io\_lib ( ) [private]

**5.128.3.16 `fw_sleep_cmd()`** void fw\_t::fw\_sleep\_cmd ( ) [private]

**5.128.3.17 `fw_until_cmd()`** void fw\_t::fw\_until\_cmd ( ) [private]

**5.128.3.18 `get_input_signal_dimension()`** void fw\_t::get\_input\_signal\_dimension ( ) [private]

**5.128.3.19 `async_read()`** void fw\_t::async\_read ( ) [inline], [private]

**5.128.3.20 `async_poll_msg()`** void fw\_t::async\_poll\_msg ( ) [private]

**5.128.3.21 `get_parserstate()`** void fw\_t::get\_parserstate ( ) [private]

## 5.128.4 Member Data Documentation

**5.128.4.1 prepare\_vars** `MHAParser::fw_vars_t fw_t::prepare_vars` [private]

**5.128.4.2 nchannels\_out** `MHAParser::int_mon_t fw_t::nchannels_out` [private]

**5.128.4.3 proc\_name** `MHAParser::string_t fw_t::proc_name` [private]

**5.128.4.4 io\_name** `MHAParser::string_t fw_t::io_name` [private]

**5.128.4.5 exit\_on\_stop** `MHAParser::bool_t fw_t::exit_on_stop` [private]

**5.128.4.6 fw\_sleep** `MHAParser::int_t fw_t::fw_sleep` [private]

**5.128.4.7 fw\_until** `MHAParser::string_t fw_t::fw_until` [private]

**5.128.4.8 fw\_cmd** `MHAParser::kw_t fw_t::fw_cmd` [private]

**5.128.4.9 parserstate** `MHAParser::string_mon_t fw_t::parserstate` [private]

**5.128.4.10 errorlog** `MHAParser::string_t fw_t::errorlog [private]`

**5.128.4.11 fatallog** `MHAParser::string_t fw_t::fatallog [private]`

**5.128.4.12 plugins** `MHAParser::vstring_t fw_t::plugins [private]`

**5.128.4.13 plugin\_paths** `MHAParser::vstring_t fw_t::plugin_paths [private]`

**5.128.4.14 dump\_mha** `MHAParser::bool_t fw_t::dump_mha [private]`

**5.128.4.15 inst\_name** `MHAParser::string_t fw_t::inst_name [private]`

A variable for naming MHA instances.

**5.128.4.16 ac** `MHAKernel::algo_comm_class_t fw_t::ac [private]`

**5.128.4.17 proc\_lib** `PluginLoader::mhaplugloader_t* fw_t::proc_lib [private]`

**5.128.4.18 cfin** `mhaconfig_t fw_t::cfin [private]`

**5.128.4.19 cfout mhaconfig\_t fw\_t::cfout [private]**

**5.128.4.20 state state\_t fw\_t::state [private]**

**5.128.4.21 b\_exit\_request bool fw\_t::b\_exit\_request [private]**

**5.128.4.22 io\_lib io\_lib\_t\* fw\_t::io\_lib [protected]**

**5.128.4.23 proc\_error int fw\_t::proc\_error [protected]**

**5.128.4.24 io\_error int fw\_t::io\_error [protected]**

**5.128.4.25 proc\_error\_string MHAParser::string\_mon\_t fw\_t::proc\_error\_string [private]**

**5.128.4.26 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**

## 5.129 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**

### 5.129.1 Constructor & Destructor Documentation

**5.129.1.1 fw\_vars\_t()** fw\_vars\_t::fw\_vars\_t ( MHAParser::parser\_t & p ) [explicit]

### 5.129.2 Member Function Documentation

**5.129.2.1 lock\_srate\_fragsize()** void fw\_vars\_t::lock\_srate\_fragsize ( )

**5.129.2.2 lock\_channels()** void fw\_vars\_t::lock\_channels ( )

**5.129.2.3 unlock\_srate\_fragsize()** void fw\_vars\_t::unlock\_srate\_fragsize ( )

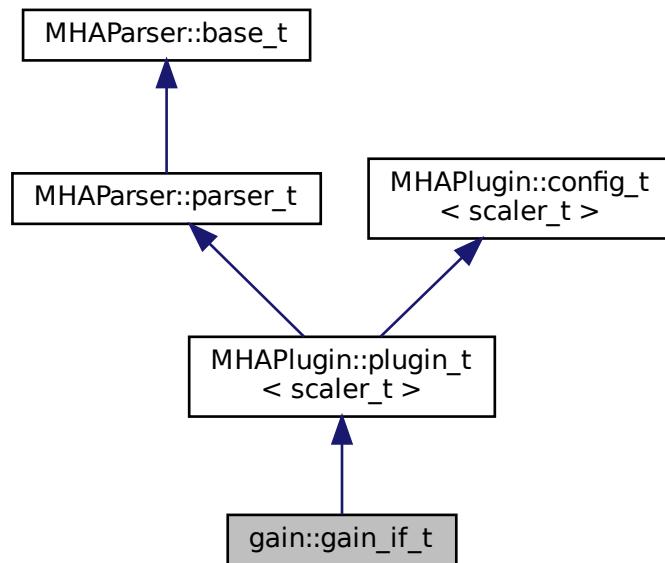
**5.129.2.4 unlock\_channels()** void fw\_vars\_t::unlock\_channels ( )**5.129.3 Member Data Documentation****5.129.3.1 pinchannels** MHAParser::int\_t fw\_vars\_t::pinchannels**5.129.3.2 pfragmentsize** MHAParser::int\_t fw\_vars\_t::pfragmentsize**5.129.3.3 psrate** MHAParser::float\_t fw\_vars\_t::psrate

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

- **mhafw\_lib.h**
- **mhafw\_lib.cpp**

**5.130 gain::gain\_if\_t Class Reference**

Inheritance diagram for gain::gain\_if\_t:



## Public Member Functions

- `gain_if_t ( algo_comm_t iac, const std::string &configured_name)`
- `mha_wave_t * process ( mha_wave_t *)`
- `mha_spec_t * process ( mha_spec_t *)`
- `void prepare ( mhaconfig_t &)`
- `void release ()`

## Private Member Functions

- `void update_gain ()`
- `void update_minmax ()`

## Private Attributes

- `MHAEvents::patchbay_t< gain_if_t > patchbay`
- `MHAParser::vfloat_t gains`
- `MHAParser::float_t vmin`
- `MHAParser::float_t vmax`

## Additional Inherited Members

### 5.130.1 Constructor & Destructor Documentation

```
5.130.1.1 gain_if_t() gain::gain_if_t::gain_if_t (
    algo_comm_t iac,
    const std::string & configured_name )
```

### 5.130.2 Member Function Documentation

```
5.130.2.1 process() [1/2] mha_wave_t * gain::gain_if_t::process (
    mha_wave_t * s )
```

**5.130.2.2 process()** [2/2] `mha_spec_t * gain::gain_if_t::process ( mha_spec_t * s )`

**5.130.2.3 prepare()** `void gain::gain_if_t::prepare ( mhaconfig_t & tf ) [virtual]`

Implements **MHAPlugIn::plugin\_t<scaler\_t>** (p. [1149](#)).

**5.130.2.4 release()** `void gain::gain_if_t::release ( ) [virtual]`

Reimplemented from **MHAPlugIn::plugin\_t<scaler\_t>** (p. [1150](#)).

**5.130.2.5 update\_gain()** `void gain::gain_if_t::update_gain ( ) [private]`

**5.130.2.6 update\_minmax()** `void gain::gain_if_t::update_minmax ( ) [private]`

### 5.130.3 Member Data Documentation

**5.130.3.1 patchbay** `MHAEvents::patchbay_t< gain_if_t > gain::gain_if_t::patchbay [private]`

**5.130.3.2 gains** `MHAParser::vfloat_t gain::gain_if_t::gains [private]`

**5.130.3.3 vmin** `MHAParser::float_t gain::gain_if_t::vmin` [private]

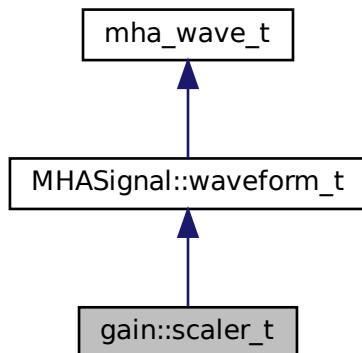
**5.130.3.4 vmax** `MHAParser::float_t gain::gain_if_t::vmax` [private]

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

- `gain.cpp`

## 5.131 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

#### 5.131.1 Constructor & Destructor Documentation

```
5.131.1.1 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**

## **5.132 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.*

### 5.132.1 Constructor & Destructor Documentation

```
5.132.1.1 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

<i>ac</i>	Handle to AC space
<i>in_cfg</i>	Input signal configuration
<i>lenOldSamps</i>	How many old samples to buffer
<i>doCircularComp</i>	Compensate for circular convolution?
<i>mu</i>	Scalar coefficient for gradient
<i>alp</i>	Autoregressive coefficient for estimating PSD
<i>useVAD</i>	Use voice activity detection?
<i>vadName_</i>	Name of the VAD AC variable

```
5.132.1.2 ~gsc_adaptive_stage() gsc_adaptive_stage::gsc_adaptive_stage::~gsc_adaptive_stage ( ) [default]
```

## 5.132.2 Member Function Documentation

```
5.132.2.1 process() mha_wave_t * gsc_adaptive_stage::gsc_adaptive_stage::process (
    mha_wave_t * wavin )
```

Processing callback.

#### Parameters

<i>wavin</i>	input signal
--------------	--------------

**Returns**

Returns a pointer to the output signal

**5.132.2.2 insert()** void gsc\_adaptive\_stage::gsc\_adaptive\_stage::insert ( ) [private]

Re-insert all AC variables into the AC space.

**5.132.3 Member Data Documentation****5.132.3.1 ac algo\_comm\_t** gsc\_adaptive\_stage::gsc\_adaptive\_stage::ac [private]

Handle to AC space.

**5.132.3.2 lenOldSamps** unsigned int gsc\_adaptive\_stage::gsc\_adaptive\_stage::len←  
OldSamps [private]

Number of old samples to buffer.

**5.132.3.3 lenNewSamps** unsigned int gsc\_adaptive\_stage::gsc\_adaptive\_stage::len←  
NewSamps [private]

Number of new samples.

**5.132.3.4 bufSize** unsigned int gsc\_adaptive\_stage::gsc\_adaptive\_stage::bufSize  
[private]

Total buffer size.

Must be lenOldSamps+lenNewSamps

**5.132.3.5 frac\_old** float gsc\_adaptive\_stage::gsc\_adaptive\_stage::frac\_old [private]

Fraction of new samples to total buffer size.

**5.132.3.6 mha\_fft** mha\_fft\_t gsc\_adaptive\_stage::gsc\_adaptive\_stage::mha\_fft [private]

FFT handle.

**5.132.3.7 nfreq** unsigned int gsc\_adaptive\_stage::gsc\_adaptive\_stage::nfreq [private]

Number of frequency bins.

**5.132.3.8 nchan** unsigned int gsc\_adaptive\_stage::gsc\_adaptive\_stage::nchan [private]

Number of channels in input signal.

**5.132.3.9 desired\_chan** unsigned int gsc\_adaptive\_stage::gsc\_adaptive\_stage::desired\_chan [private]

Channel index containing the desired response.

Always last channel by convention

**5.132.3.10 doCircularComp** bool gsc\_adaptive\_stage::gsc\_adaptive\_stage::doCircularComp [private]

Whether to compensate for circular convolution.

**5.132.3.11 mu** float gsc\_adaptive\_stage::gsc\_adaptive\_stage::mu [private]

Linear coefficient for gradient.

**5.132.3.12 alp** float gsc\_adaptive\_stage::gsc\_adaptive\_stage::alp [private]

Autoregressive coefficient for estimating PSD.

**5.132.3.13 useVAD** bool gsc\_adaptive\_stage::gsc\_adaptive\_stage::useVAD [private]

Wether to use VAD.

**5.132.3.14 vadName** std::string gsc\_adaptive\_stage::gsc\_adaptive\_stage::vadName [private]

Name of VAD AC variable.

**5.132.3.15 x** MHA\_AC::waveform\_t gsc\_adaptive\_stage::gsc\_adaptive\_stage::x [private]

Buffered input signal.

**5.132.3.16 X** MHA\_AC::spectrum\_t gsc\_adaptive\_stage::gsc\_adaptive\_stage::X [private]

FFT of the buffered input signal.

**5.132.3.17 W** MHA\_AC::spectrum\_t gsc\_adaptive\_stage::gsc\_adaptive\_stage::W [private]

Time-varying filter.

**5.132.3.18 Y** `MHA_AC::spectrum_t` `gsc_adaptive_stage::gsc_adaptive_stage::Y` [private]

Filter output, frequency domain.

**5.132.3.19 y** `MHA_AC::waveform_t` `gsc_adaptive_stage::gsc_adaptive_stage::y` [private]

Filter output, time domain.

**5.132.3.20 d** `MHA_AC::waveform_t` `gsc_adaptive_stage::gsc_adaptive_stage::d` [private]

Desired response.

**5.132.3.21 e** `MHA_AC::waveform_t` `gsc_adaptive_stage::gsc_adaptive_stage::e` [private]

Error signal, time domain.

**5.132.3.22 E** `MHA_AC::spectrum_t` `gsc_adaptive_stage::gsc_adaptive_stage::E` [private]

Error signal, frequency domain.

**5.132.3.23 E2** `MHA_AC::spectrum_t` `gsc_adaptive_stage::gsc_adaptive_stage::E2` [private]

Error spectrum multiplied by input spectrum:  $E2=X*E$ .

**5.132.3.24 grad** `MHA_AC::waveform_t` `gsc_adaptive_stage::gsc_adaptive_stage::grad` [private]

Gradient.

**5.132.3.25 Grad** `MHA_AC::spectrum_t gsc_adaptive_stage::gsc_adaptive_stage::Grad`  
[private]

FT of the gradient.

**5.132.3.26 e\_out** `MHA_AC::waveform_t gsc_adaptive_stage::gsc_adaptive_stage::e_out`  
[private]

Error signal.

**5.132.3.27 P** `MHA_AC::waveform_t gsc_adaptive_stage::gsc_adaptive_stage::P` [private]

Signal power estimate.

**5.132.3.28 Psum** `MHA_AC::waveform_t gsc_adaptive_stage::gsc_adaptive_stage::Psum`  
[private]

Signal power estimate, summed over all channels.

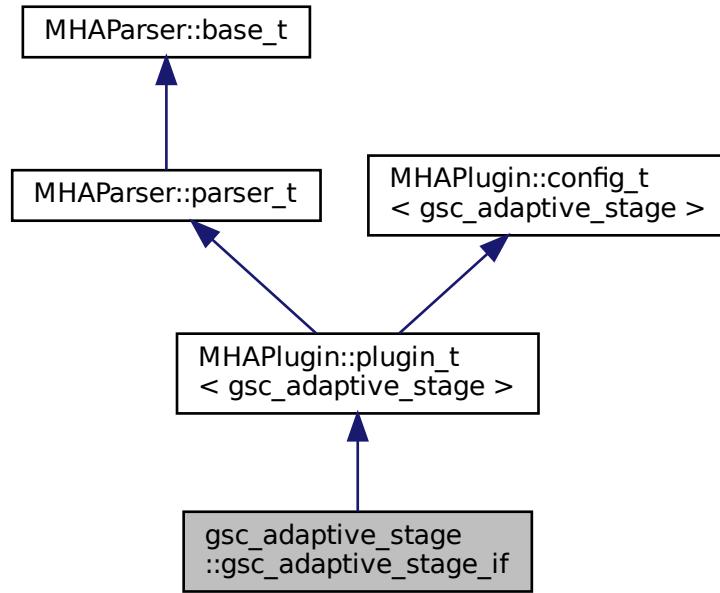
The documentation for this class was generated from the following files:

- `gsc_adaptive_stage.hh`
- `gsc_adaptive_stage.cpp`

## 5.133 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 iac, const std::string &configured\_name)**  
*Constructs the interface to the adaptive filter plugin.*
- **~gsc\_adaptive\_stage\_if ()=default**
- **mha\_wave\_t \* process ( mha\_wave\_t \*)**  
*This plugin implements noise reduction using spectral subtraction: By nonnegative subtraction from the output magnitude of the estimated noise magnitude spectrum.*
- **void prepare ( mhaconfig\_t &)**  
*Plugin preparation.*
- **void release (void)**

### Private Member Functions

- **void update\_cfg ()**  
*Update the rt config.*
- **void on\_model\_param\_valuechanged ()**

## Private Attributes

- **MHAEvents::patchbay\_t < gsc\_adaptive\_stage\_if > patchbay**
- **MHAParser::int\_t lenOldSamps**  
*How many old samples should be buffered per filter block.*
- **MHAParser::bool\_t doCircularComp**  
*Whether to compensate for circular convolution.*
- **MHAParser::float\_t mu**  
*Linear coefficient for gradient used in filter adaption.*
- **MHAParser::float\_t alp**  
*Autoregressive coefficient for PSD estimation.*
- **MHAParser::bool\_t useVAD**  
*Whether to use VAD for conditional filter adaption.*
- **MHAParser::string\_t vadName**  
*Name of VAD AC variable.*

## Additional Inherited Members

### 5.133.1 Detailed Description

Plugin interface class.

### 5.133.2 Constructor & Destructor Documentation

```
5.133.2.1 gsc_adaptive_stage_if() gsc_adaptive_stage::gsc_adaptive_stage_if::gsc_adaptive_stage_if (
    algo_comm_t iac,
    const std::string & configured_name )
```

Constructs the interface to the adaptive filter plugin.

#### Parameters

<i>ac</i>	Handle to the ac space
-----------	------------------------

### 5.133.2.2 ~gsc\_adaptive\_stage\_if() gsc\_adaptive\_stage::gsc\_adaptive\_stage\_if ~

```
::~gsc_adaptive_stage_if ( ) [default]
```

### 5.133.3 Member Function Documentation

**5.133.3.1 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 a the signal modified by this plugin.

**5.133.3.2 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. [1149](#)).

**5.133.3.3 release()** void gsc\_adaptive\_stage::gsc\_adaptive\_stage\_if::release ( void ) [inline], [virtual]

Reimplemented from **MHAPlugIn::plugin\_t< gsc\_adaptive\_stage >** (p. 1150).

**5.133.3.4 update\_cfg()** void gsc\_adaptive\_stage::gsc\_adaptive\_stage\_if::update\_cfg ( ) [private]

Update the rt config.

**5.133.3.5 on\_model\_param\_valuechanged()** void gsc\_adaptive\_stage::gsc\_adaptive\_stage\_if::on\_model\_param\_valuechanged ( ) [private]

## 5.133.4 Member Data Documentation

**5.133.4.1 patchbay** **MHAEvents::patchbay\_t< gsc\_adaptive\_stage\_if >** gsc\_adaptive\_stage::gsc\_adaptive\_stage\_if::patchbay [private]

**5.133.4.2 lenOldSamps** **MHAParser::int\_t** gsc\_adaptive\_stage::gsc\_adaptive\_stage\_if::lenOldSamps [private]

How many old samples should be buffered per filter block.

**5.133.4.3 doCircularComp** **MHAParser::bool\_t** gsc\_adaptive\_stage::gsc\_adaptive\_stage\_if::doCircularComp [private]

Whether to compensate for circular convolution.

**5.133.4.4 mu** `MHAParser::float_t` `gsc_adaptive_stage::gsc_adaptive_stage_if::mu`  
[private]

Linear coefficient for gradient used in filter adaption.

**5.133.4.5 alp** `MHAParser::float_t` `gsc_adaptive_stage::gsc_adaptive_stage_if::alp`  
[private]

Autoregressive coefficient for PSD estimation.

**5.133.4.6 useVAD** `MHAParser::bool_t` `gsc_adaptive_stage::gsc_adaptive_stage_if::useVAD`  
[private]

Wether to use VAD for conditional filter adaption.

**5.133.4.7 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`

## 5.134 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.*

### 5.134.1 Detailed Description

Configuration for Gammatone Filterbank Analyzer.

### 5.134.2 Constructor & Destructor Documentation

```
5.134.2.1 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.

**5.134.2.2 ~gtfb\_analyzer\_cfg\_t()** `gtfb_analyzer::gtfb_analyzer_cfg_t::~gtfb_analyzer_cfg_t ( ) [inline]`

### 5.134.3 Member Function Documentation

**5.134.3.1 bands()** `unsigned gtfb_analyzer::gtfb_analyzer_cfg_t::bands ( ) const [inline]`

Each band is split into this number of bands.

**5.134.3.2 channels()** `unsigned gtfb_analyzer::gtfb_analyzer_cfg_t::channels ( ) const [inline]`

The number of separate audio channels.

**5.134.3.3 frames()** `unsigned gtfb_analyzer::gtfb_analyzer_cfg_t::frames ( ) const [inline]`

The number of frames in one chunk.

**5.134.3.4 states()** `mha_complex_t* gtfb_analyzer::gtfb_analyzer_cfg_t::states (`  
    `unsigned channel,`  
    `unsigned band ) [inline]`

Returns pointer to filter states for that band.

**5.134.3.5 cvalue()** `mha_complex_t& gtfb_analyzer::gtfb_analyzer_cfg_t::cvalue (`  
    `unsigned frame,`  
    `unsigned channel,`  
    `unsigned band ) [inline]`

#### 5.134.4 Member Data Documentation

**5.134.4.1 order** `unsigned gtfb_analyzer::gtfb_analyzer_cfg_t::order`

The order of the gammatone filters.

**5.134.4.2 coeff** `std::vector< mha_complex_t> gtfb_analyzer::gtfb_analyzer_cfg_t::coeff`

The complex coefficients of the gammatone filter bands.

**5.134.4.3 norm\_phase** `std::vector< mha_complex_t> gtfb_analyzer::gtfb_analyzer_cfg_t::norm_phase`

Combination of normalization and phase correction factor.

**5.134.4.4 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. 836) 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

**5.134.5 state std::vector< mha\_complex\_t > gtfb\_analyzer::gtfb\_analyzer\_cfg\_t::state**

Storage for Filter state.

Holds **channels()** (p. 547) \* **bands()** (p. 547) \* order complex filter states. Layout: state[(**bands()** (p. 547)\*channel+band)\*order+stage]

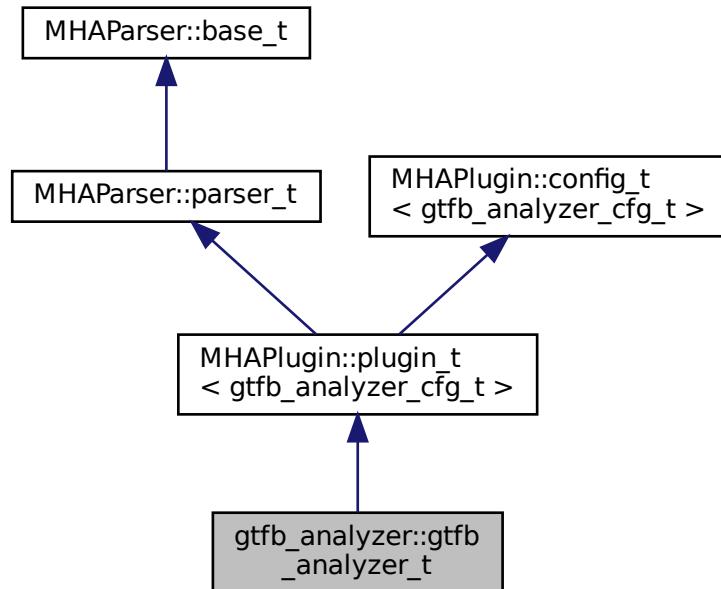
The documentation for this struct was generated from the following file:

- **gtfb\_analyzer.cpp**

**5.135 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 ( algo_comm_t iac, const std::string &configured_name)`
- `mha_wave_t * process ( mha_wave_t *)`
- `void prepare ( mhaconfig_t &)`

## Private Member Functions

- `void update_cfg ()`

## Private Attributes

- `MHAEvents::patchbay_t< gtfb_analyzer_t > patchbay`
- `bool prepared`
- `MHAParser::int_t order`
- `MHAParser::vcomplex_t coeff`
- `MHAParser::vcomplex_t norm_phase`

## Additional Inherited Members

### 5.135.1 Detailed Description

Gammatone Filterbank Analyzer Plugin.

### 5.135.2 Constructor & Destructor Documentation

```
5.135.2.1 gtfb_analyzer_t() gtfb_analyzer::gtfb_analyzer_t::gtfb_analyzer_t (
    algo_comm_t iac,
    const std::string & configured_name )
```

### 5.135.3 Member Function Documentation

```
5.135.3.1 process() mha_wave_t * gtfb_analyzer::gtfb_analyzer_t::process (
    mha_wave_t * s )
```

```
5.135.3.2 prepare() void gtfb_analyzer::gtfb_analyzer_t::prepare (
    mhaconfig_t & tf ) [virtual]
```

Implements **MHAPlugin::plugin\_t< gtfb\_analyzer\_cfg\_t >** (p. [1149](#)).

```
5.135.3.3 update_cfg() void gtfb_analyzer::gtfb_analyzer_t::update_cfg ( ) [private]
```

### 5.135.4 Member Data Documentation

**5.135.4.1 patchbay** `MHAEvents::patchbay_t< gtfb_analyzer_t> gtfb_analyzer::gtfb↔_analyzer_t::patchbay` [private]

**5.135.4.2 prepared** `bool gtfb_analyzer::gtfb_analyzer_t::prepared` [private]

**5.135.4.3 order** `MHAParser::int_t gtfb_analyzer::gtfb_analyzer_t::order` [private]

**5.135.4.4 coeff** `MHAParser::vcomplex_t gtfb_analyzer::gtfb_analyzer_t::coeff` [private]

**5.135.4.5 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`

## 5.136 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**

### 5.136.1 Detailed Description

Configuration for Gammatone Filterbank SIMD Analyzer.

### 5.136.2 Constructor & Destructor Documentation

#### 5.136.2.1 gtfb\_simd\_cfg\_t() [1/3] `gtfb_simd_cfg_t::gtfb_simd_cfg_t (` `gtfb_simd_cfg_t && ) [delete]`

---

**5.136.2.2 `gtfb_simd_cfg_t()` [2/3]** `gtfb_simd_cfg_t::gtfb_simd_cfg_t (`  
`const gtfb_simd_cfg_t & ) [delete]`

**5.136.2.3 `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

<code>ch</code>	Number of Audio channels.
<code>frames</code>	Number of Audio frames per chunk.
<code>ord</code>	The order of the gammatone filters.
<code>_coeff</code>	Complex gammatone filter coefficients.
<code>_norm_phase</code>	Normalization and phase correction factors.

**5.136.2.4 `~gtfb_simd_cfg_t()`** `gtfb_simd_cfg_t::~gtfb_simd_cfg_t ( ) [inline]`

### 5.136.3 Member Function Documentation

**5.136.3.1 `get_bands()`** `unsigned gtfb_simd_cfg_t::get_bands ( ) const [inline]`

Each band is split into this number of bands.

**5.136.3.2 `get_channels()`** `unsigned gtfb_simd_cfg_t::get_channels ( ) const [inline]`

The number of separate audio channels.

**5.136.3.3 get\_frames()** `unsigned gtfb_simd_cfg_t::get_frames () const [inline]`

The number of frames in one chunk.

**5.136.3.4 operator=() [1/2]** `gtfb_simd_cfg_t& gtfb_simd_cfg_t::operator= ( gtfb_simd_cfg_t && ) [delete]`**5.136.3.5 operator=() [2/2]** `gtfb_simd_cfg_t& gtfb_simd_cfg_t::operator= ( const gtfb_simd_cfg_t & ) [delete]`**5.136.3.6 process()** `mha_wave_t* gtfb_simd_cfg_t::process ( mha_wave_t * s_in ) [inline]`

## 5.136.4 Member Data Documentation

**5.136.4.1 order** `unsigned gtfb_simd_cfg_t::order [private]`

The order of the gammatone filters.

**5.136.4.2 bands** `unsigned gtfb_simd_cfg_t::bands [private]`

Number of frequency bands per channel.

**5.136.4.3 channels** `unsigned gtfb_simd_cfg_t::channels [private]`

Number of input audio channels.

**5.136.4.4 bandsXchannels** `unsigned gtfb_simd_cfg_t::bandsXchannels [private]`

Product of bands and channels.

**5.136.4.5 norm\_phase** `std::vector< mha_complex_t> gtfb_simd_cfg_t::norm_phase [private]`

Combination of normalization and phase correction factor.

**5.136.4.6 rinputs** `float* gtfb_simd_cfg_t::rinputs [private]`

input signal (1 sample) multiplied with norm\_phase

**5.136.4.7 iinputs** `float* gtfb_simd_cfg_t::iinputs [private]`**5.136.4.8 rcoefficients** `float* gtfb_simd_cfg_t::rcoefficients [private]`

The complex coefficients of the gammatone filter bands.

**5.136.4.9 icoefficients** `float* gtfb_simd_cfg_t::icoefficients [private]`**5.136.4.10 rstates** `float* gtfb_simd_cfg_t::rstates [private]`

Storage for Filter state. Holds **channels()** (p. 555) \* **bands()** (p. 555) \* order complex filter states. Layout: state[stage \* bandsXchannels + **bands()** (p. 555)\*channel+band]

**5.136.4.11 istates** float\* gtfb\_simd\_cfg\_t::istates [private]**5.136.4.12 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

**5.136.4.13 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.

**5.136.4.14 s\_out** mha\_wave\_t gtfb\_simd\_cfg\_t::s\_out [private]

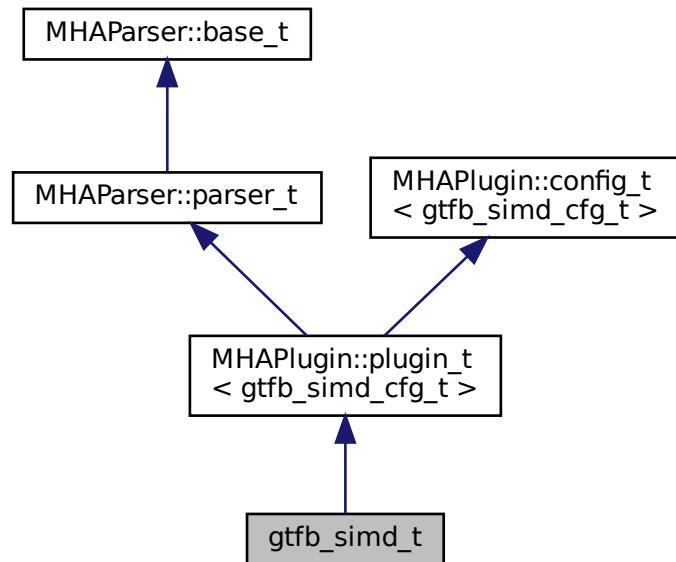
Storage for the (complex) output signal. Each of the real input audio channels is split into frequency bands with complex time signal output. The split complex time signal is again stored in a **mha\_wave\_t** (p. 836) buffer. Each complex time signal is stored as adjacent real and imaginary channels. Complex output from one source channel is stored in adjacent complex output channels.

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

- **gtfb\_simd.cpp**

## 5.137 gtfb\_simd\_t Class Reference

Inheritance diagram for gtfb\_simd\_t:



### Public Member Functions

- `gtfb_simd_t ( algo_comm_t iac, const std::string &configured_name)`
- `mha_wave_t * process ( mha_wave_t *)`
- `void prepare ( mhaconfig_t &)`

### Private Member Functions

- `void update_cfg ()`

### Private Attributes

- `MHAEvents::patchbay_t< gtfb_simd_t > patchbay`
- `bool prepared`
- `MHAParser::int_t order`
- `MHAParser::vcomplex_t coeff`
- `MHAParser::vcomplex_t norm_phase`

## Additional Inherited Members

### 5.137.1 Constructor & Destructor Documentation

```
5.137.1.1 gtfb_simd_t() gtfb_simd_t::gtfb_simd_t (
    algo_comm_t iac,
    const std::string & configured_name )
```

### 5.137.2 Member Function Documentation

```
5.137.2.1 process() mha_wave_t * gtfb_simd_t::process (
    mha_wave_t * s )
```

```
5.137.2.2 prepare() void gtfb_simd_t::prepare (
    mhaconfig_t & tf ) [virtual]
```

Implements **MHAPlugin::plugin\_t< gtfb\_simd\_cfg\_t >** (p. [1149](#)).

```
5.137.2.3 update_cfg() void gtfb_simd_t::update_cfg ( ) [private]
```

### 5.137.3 Member Data Documentation

```
5.137.3.1 patchbay MHAEvents::patchbay_t< gtfb_simd_t > gtfb_simd_t::patchbay
[private]
```

**5.137.3.2 prepared** `bool gtfb_simd_t::prepared [private]`

**5.137.3.3 order** `MHAParser::int_t gtfb_simd_t::order [private]`

**5.137.3.4 coeff** `MHAParser::vcomplex_t gtfb_simd_t::coeff [private]`

**5.137.3.5 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`

## 5.138 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.*

### 5.138.1 Detailed Description

Runtime configuration class of gtfb\_simple\_bridge plugin.

### 5.138.2 Constructor & Destructor Documentation

---

**5.138.2.1 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 )
```

### 5.138.3 Member Function Documentation

**5.138.3.1 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

<code>s</code>	real-valued, broad-band input signal
----------------	--------------------------------------

#### Returns

real part of complex-valued output signal after `pre_stages` gammatone filter orders have been applied in each band. Order of output bands in real and imaginary output are: (channel0,band0), (channel0,band1), ..., (cannel1,band0), ...

**5.138.3.2 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), ...

### 5.138.3.3 **get\_gf()** const **MHAFilter::gamma\_flt\_t&** gtfb\_simple\_rt\_t::get\_gf ( ) const [inline]

Const-accessor to contained gammatone filterbank object.

### 5.138.3.4 **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.

## 5.138.4 Member Data Documentation

**5.138.4.1 \_order** `unsigned int gtfb_simple_rt_t::_order [private]`

Total number of gammatone filter orders.

**5.138.4.2 \_pre\_stages** `unsigned int gtfb_simple_rt_t::_pre_stages [private]`

Number of filter orders applied before the loaded plugin is invoked.

**5.138.4.3 nbands** `unsigned int gtfb_simple_rt_t::nbands [private]`

Number of frequency bands to produce = number of gammatone filters.

**5.138.4.4 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.

**5.138.4.5 accf** `MHA_AC::waveform_t gtfb_simple_rt_t::accf [private]`

AC variable to publish the center frequencies of the gammatone filters.

**5.138.4.6 acbw** `MHA_AC::waveform_t gtfb_simple_rt_t::acb [private]`

AC variable to publish the bandwidths of the gammatone filters.

**5.138.4.7 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.

**5.138.4.8 output** `MHASignal::waveform_t` `gtfb_simple_rt_t::output` [private]

Resynthesized broadband signal, used as the output signal of this plugin.

**5.138.4.9 gf** `MHAFilter::gamma_flt_t` `gtfb_simple_rt_t::gf` [private]

The gammatone filter bank implementation.

**5.138.4.10 cLTASS** `MHA_AC::waveform_t` `gtfb_simple_rt_t::cLTASS` [private]

AC variable to publish band-specific LTASS level correction values.

**5.138.4.11 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.

**5.138.4.12 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.

**5.138.4.13 \_ac** `algo_comm_t` `gtfb_simple_rt_t::_ac` [private]

Algorithm Communication Variable space.

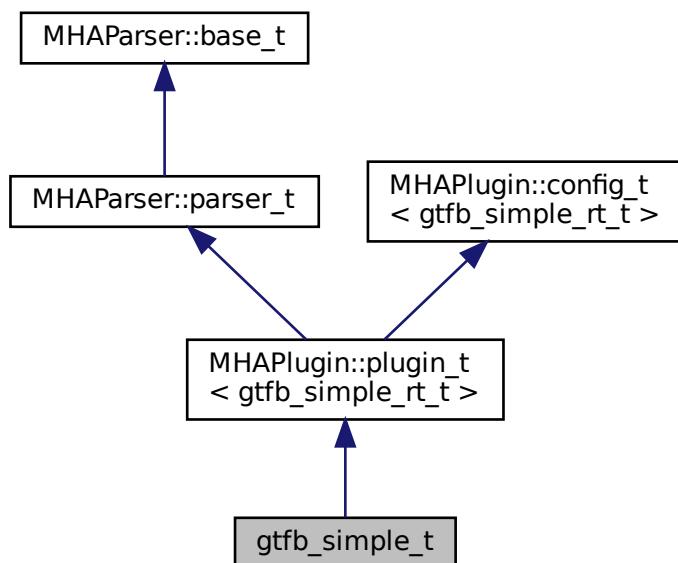
The documentation for this class was generated from the following file:

- `gtfb_simple_bridge.cpp`

## 5.139 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 iac, const std::string &configured_name)`**  
*Constructor.*
- **`void prepare ( mhaconfig_t &chcfg)`**  
*Prepare contained plugin for signal processing.*
- **`void release ()`**  
*Releases contained plugin.*
- **`mha_wave_t * process ( mha_wave_t *sln)`**  
*Process the input signal: Computes the filterbank with the pre-stages, calls the loaded plugin for processing of the filterbank signal, and resynthesizes the result to a modified broadband signal.*
- **`void setlock (bool b)`**  
*Locks / unlocks all configuration variables before / after signal processing.*

## Private Attributes

- **MHAParser::mhaplugloader\_t plug**  
*Handle to the loaded plugin.*
- **MHAOvlFilter::fscale\_bw\_t fscale**  
*Object determines the filterbank frequencies.*
- **MHAParser::int\_t order**  
*total order of gammatone filter*
- **MHAParser::int\_t prestages**  
*Number of gammatone filter order to process before the loaded plugin processes the filterbank signal.*
- **MHAParser::int\_t desired\_delay**  
*Desired group delay in audio samples.*
- **MHAParser::string\_t element\_gain\_name**  
*Number of AC variable to take element-wise gain factors from for resynthesis.*
- **MHAParser::vfloat\_mon\_t cLTASS**  
*Monitoring of LTASS correction values / dB.*
- **MHAParser::vfloat\_mon\_t resynthesis\_gain**  
*configured frequency-specific resynthesis gains*
- **MHAParser::string\_mon\_t gf\_internals**  
*For tests and debugging: a serialization of the gammatone filter internals.*
- std::string **name\_**  
*Configured algorithm name, used to name the AC variables.*

## Additional Inherited Members

### 5.139.1 Detailed Description

Interface class of gtfb\_simple\_bridge plugin.

### 5.139.2 Constructor & Destructor Documentation

#### 5.139.2.1 gtfb\_simple\_t() gtfb\_simple\_t::gtfb\_simple\_t (

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

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

## 5.139.3 Member Function Documentation

**5.139.3.1 `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. [1149](#)).

**5.139.3.2 `release()`** `void gtfb_simple_t::release ( ) [virtual]`

Releases contained plugin.

Unlocks all variables.

Reimplemented from `MHAPlugin::plugin_t< gtfb_simple_rt_t >` (p. [1150](#)).

**5.139.3.3 `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.

```
5.139.3.4 setlock() void gtfb_simple_t::setlock (
    bool b ) [inline]
```

Locks / unlocks all configuration variables before / after signal processing.

#### 5.139.4 Member Data Documentation

**5.139.4.1 plug** `MHAParser::mhapluginloader_t` `gtfb_simple_t::plug` [private]

Handle to the loaded plugin.

**5.139.4.2 fscale** `MHAOv1Filter::fscale_bw_t` `gtfb_simple_t::fscale` [private]

Object determines the filterbank frequencies.

**5.139.4.3 order** `MHAParser::int_t` `gtfb_simple_t::order` [private]

total order of gammatone filter

**5.139.4.4 prestages** `MHAParser::int_t` `gtfb_simple_t::prestages` [private]

Number of gammatone filter order to process before the loaded plugin processes the filterbank signal.

**5.139.4.5 desired\_delay** `MHAParser::int_t` `gtfb_simple_t::desired_delay` [private]

Desired group delay in audio samples.

**5.139.4.6 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.

**5.139.4.7 cLTASS** `MHAParser::vfloat_mon_t gtfb_simple_t::cLTASS` [private]

Monitoring of LTASS correction values / dB.

**5.139.4.8 resynthesis\_gain** `MHAParser::vfloat_mon_t gtfb_simple_t::resynthesis_` ↵  
gain [private]

configured frequency-specific resynthesis gains

**5.139.4.9 gf\_internals** `MHAParser::string_mon_t gtfb_simple_t::gf_internals` [private]

For tests and debugging: a serialization of the gammatone filter internals.

**5.139.4.10 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`

## 5.140 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**

### 5.140.1 Constructor & Destructor Documentation

**5.140.1.1 hanning\_ramps\_t()** hanning\_ramps\_t::hanning\_ramps\_t ( unsigned int *la*,  
unsigned int *lb* )

**5.140.1.2 ~hanning\_ramps\_t()** hanning\_ramps\_t::~hanning\_ramps\_t ( )

### 5.140.2 Member Function Documentation

**5.140.2.1 operator()()** void hanning\_ramps\_t::operator() ( MHASignal::waveform\_t & *b* )

### 5.140.3 Member Data Documentation

**5.140.3.1 len\_a** unsigned int hanning\_ramps\_t::len\_a [private]

**5.140.3.2 len\_b** unsigned int hanning\_ramps\_t::len\_b [private]

### 5.140.3.3 **ramp\_a** `mha_real_t* hanning_ramps_t::ramp_a` [private]

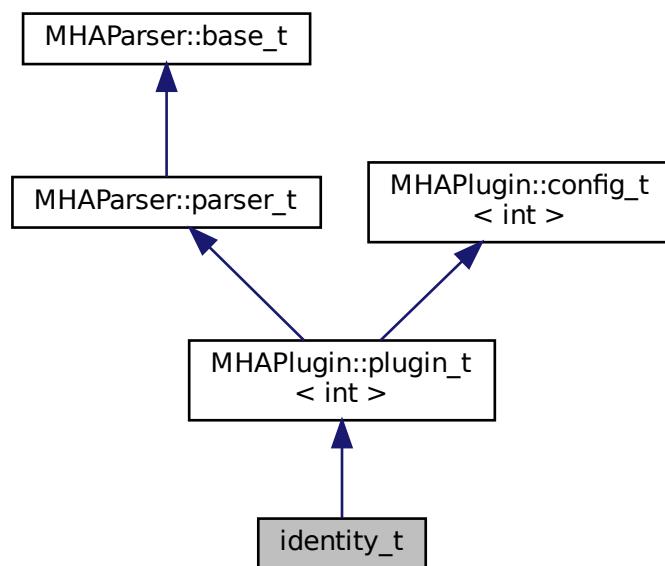
### 5.140.3.4 **ramp\_b** `mha_real_t* hanning_ramps_t::ramp_b` [private]

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

- `spec2wave.cpp`

## 5.141 **identity\_t** Class Reference

Inheritance diagram for `identity_t`:



### Public Member Functions

- `identity_t ( algo_comm_t iac, const std::string &configured_name)`
- `mha_wave_t * process ( mha_wave_t *)`
- `mha_spec_t * process ( mha_spec_t *)`
- `void prepare ( mhaconfig_t &)`
- `void release ()`

## Additional Inherited Members

### 5.141.1 Constructor & Destructor Documentation

```
5.141.1.1 identity_t() identity_t::identity_t (
    algo_comm_t iac,
    const std::string & configured_name )
```

### 5.141.2 Member Function Documentation

```
5.141.2.1 process() [1/2] mha_wave_t * identity_t::process (
    mha_wave_t * s )
```

```
5.141.2.2 process() [2/2] mha_spec_t * identity_t::process (
    mha_spec_t * s )
```

```
5.141.2.3 prepare() void identity_t::prepare (
    mhaconfig_t & ) [virtual]
```

Implements **MHAPlugin::plugin\_t< int >** (p. 1149).

```
5.141.2.4 release() void identity_t::release ( ) [virtual]
```

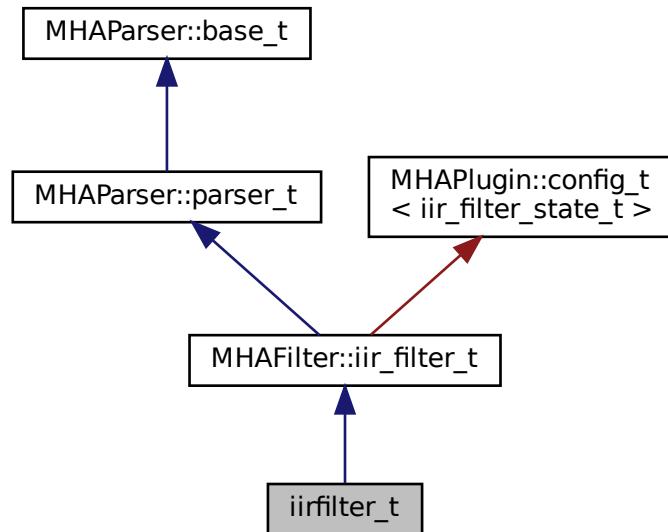
Reimplemented from **MHAPlugin::plugin\_t< int >** (p. 1150).

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

- **identity.cpp**

## 5.142 iirfilter\_t Class Reference

Inheritance diagram for iirfilter\_t:



### Public Member Functions

- `iirfilter_t ( algo_comm_t iac, const std::string &configured_name)`
- `void prepare_ ( mhaconfig_t &)`
- `void release_ ()`
- `mha_wave_t * process ( mha_wave_t *)`

### Additional Inherited Members

#### 5.142.1 Constructor & Destructor Documentation

**5.142.1.1 `iirfilter_t()`** `iirfilter_t::iirfilter_t (`  
`algo_comm_t iac,`  
`const std::string & configured_name )`

### 5.142.2 Member Function Documentation

**5.142.2.1 `prepare_()`** `void iirfilter_t::prepare_ (`  
`mhaconfig_t & tf )`

**5.142.2.2 `release_()`** `void iirfilter_t::release_ ( ) [inline]`

**5.142.2.3 `process()`** `mha_wave_t * iirfilter_t::process (`  
`mha_wave_t * s )`

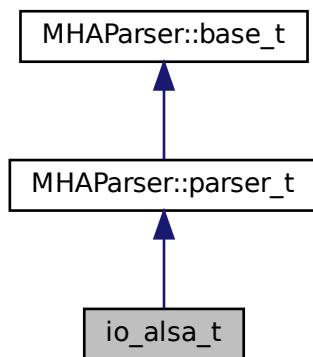
The documentation for this class was generated from the following file:

- `iirfilter.cpp`

## 5.143 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 alsal\_start\_counter**
- **MHAEvents::patchbay\_t< io\_alsa\_t > patchbay**

## Additional Inherited Members

### 5.143.1 Detailed Description

MHA IO interface class for ALSA IO.

### 5.143.2 Constructor & Destructor Documentation

```
5.143.2.1 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.

### 5.143.3 Member Function Documentation

```
5.143.3.1 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

```
5.143.3.2 release() void io_alsa_t::release ( )
```

MHA framework leaves prepared state.

**5.143.3.3 start()** void io\_alsa\_t::start ( )

MHA framework calls this function when signal processing should start.

**5.143.3.4 stop()** void io\_alsa\_t::stop ( )

MHA framework calls this function when signal processing should stop.

**5.143.3.5 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.

**5.143.3.6 process()** void io\_alsa\_t::process ( ) [private]**5.143.3.7 prepare() [2/2]** template<>  
void io\_alsa\_t::prepare ( int nch\_in, int nch\_out )**5.143.4 Member Data Documentation****5.143.4.1 b\_process** bool io\_alsa\_t::b\_process [private]**5.143.4.2 fw fragsize** unsigned int io\_alsa\_t::fw fragsize [private]

**5.143.4.3 fw\_samplerate** `unsigned int io_alsa_t::fw_samplerate [private]`

**5.143.4.4 proc\_event** `IOProcessEvent_t io_alsa_t::proc_event [private]`

**5.143.4.5 proc\_handle** `void* io_alsa_t::proc_handle [private]`

**5.143.4.6 start\_event** `IOSTartedEvent_t io_alsa_t::start_event [private]`

**5.143.4.7 start\_handle** `void* io_alsa_t::start_handle [private]`

**5.143.4.8 stop\_event** `IOSStoppedEvent_t io_alsa_t::stop_event [private]`

**5.143.4.9 stop\_handle** `void* io_alsa_t::stop_handle [private]`

**5.143.4.10 dev\_in** `alsa_base_t* io_alsa_t::dev_in [private]`

**5.143.4.11 dev\_out** `alsa_base_t* io_alsa_t::dev_out [private]`

**5.143.4.12 proc\_thread** pthread\_t io\_alsa\_t::proc\_thread [private]

**5.143.4.13 p\_in** alsa\_dev\_par\_parser\_t io\_alsa\_t::p\_in [private]

**5.143.4.14 p\_out** alsa\_dev\_par\_parser\_t io\_alsa\_t::p\_out [private]

**5.143.4.15 pcmlink** MHAParser::bool\_t io\_alsa\_t::pcmlink [private]

**5.143.4.16 priority** MHAParser::int\_t io\_alsa\_t::priority [private]

**5.143.4.17 format** MHAParser::kw\_t io\_alsa\_t::format [private]

**5.143.4.18 alsa\_start\_counter** MHAParser::int\_mon\_t io\_alsa\_t::alsa\_start\_counter [private]

**5.143.4.19 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**

## **5.144 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**

### 5.144.1 Detailed Description

TCP sound-io library's interface to the framework callbacks.

### 5.144.2 Constructor & Destructor Documentation

---

**5.144.2.1 `io_asterisk_fwcb_t()`** `io_asterisk_fwcb_t::io_asterisk_fwcb_t (`  
`IOPProcessEvent_t proc_event,`  
`void * proc_handle,`  
`IOStartedEvent_t start_event,`  
`void * start_handle,`  
`IOStoppedEvent_t stop_event,`  
`void * stop_handle )`

Constructor stores framework handles and initializes error numbers to 0.

**5.144.2.2 `~io_asterisk_fwcb_t()`** `virtual io_asterisk_fwcb_t::~io_asterisk_fwcb_t (`  
`) [inline], [virtual]`

Do-nothing destructor.

### 5.144.3 Member Function Documentation

**5.144.3.1 `start()`** `void io_asterisk_fwcb_t::start ( ) [virtual]`

Call the framework's start callback.

**5.144.3.2 `process()`** `int io_asterisk_fwcb_t::process (`  
`mha_wave_t * sIn,`  
`mha_wave_t *& sOut ) [virtual]`

Call the frameworks processing callback.

#### Parameters

<code>sIn</code>	The input sound data just received from TCP.
<code>sOut</code>	A pointer to output sound data. Will point to the output sound data storage when the callback finishes.

**Returns**

Status, an error number from the signal processing callback. If this is != 0, then the connection should be closed.

**5.144.3.3 `set_errnos()`** `void io_asterisk_fwcb_t::set_errnos ( int proc_err, int io_err ) [virtual]`

Save error numbers to use during.

**See also**

[stop](#) (p. 583)

**Parameters**

<code>proc_err</code>	The error number from the
-----------------------	---------------------------

**See also**

[process](#) (p. 582) callback.

**Parameters**

<code>io_err</code>	The error number from the io library itself.
---------------------	--

**5.144.3.4 `stop()`** `void io_asterisk_fwcb_t::stop ( ) [virtual]`

Call the frameworks stop callback.

Uses the error numbers set previously with

**See also**

[set\\_errnos](#) (p. 583).

## 5.144.4 Member Data Documentation

### 5.144.4.1 **proc\_event** `IOProcessEvent_t` `io_asterisk_fwcb_t::proc_event` [private]

Pointer to signal processing callback function.

### 5.144.4.2 **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.

### 5.144.4.3 **stop\_event** `IOSToppedEvent_t` `io_asterisk_fwcb_t::stop_event` [private]

Pointer to stop notification callback function.

Called when the connection is closed.

### 5.144.4.4 **proc\_handle** `void*` `io_asterisk_fwcb_t::proc_handle` [private]

Handles belonging to framework.

### 5.144.4.5 **start\_handle** `void *` `io_asterisk_fwcb_t::start_handle` [private]

### 5.144.4.6 **stop\_handle** `void *` `io_asterisk_fwcb_t::stop_handle` [private]

**5.144.4.7 proc\_err** int io\_asterisk\_fwcb\_t::proc\_err [private]

Errors from the processing callback and from the TCP IO itself are stored here before closing Network handles.

MHAIOTCP is notified by the server when the connection has been taken down, and calls

**See also**

**stop** (p. 583) from that callback. Within **stop** (p. 583), these error numbers are read again and transmitted to the framework.

**5.144.4.8 io\_err** int io\_asterisk\_fwcb\_t::io\_err [private]

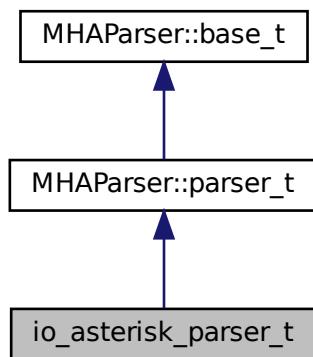
The documentation for this class was generated from the following file:

- **MHAIOAsterisk.cpp**

## 5.145 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

### 5.145.1 Detailed Description

The parser interface of the IOAsterisk library.

### 5.145.2 Constructor & Destructor Documentation

#### 5.145.2.1 **io\_asterisk\_parser\_t()** `io_asterisk_parser_t::io_asterisk_parser_t ( )`

Constructor initializes parser variables.

#### 5.145.2.2 **~io\_asterisk\_parser\_t()** `virtual io_asterisk_parser_t::~io_asterisk_parser_t ( ) [inline], [virtual]`

Do-nothing destructor.

### 5.145.3 Member Function Documentation

#### 5.145.3.1 **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.

---

**5.145.3.2 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.

**5.145.3.3 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

<i>port</i>	The TCP port number that is currently used. In the range [MIN_TCP_PORT, MAX_TCP_PORT], excluding 0.
-------------	---

#### Precondition

[get\\_local\\_port\(\)](#) (p. 587) currently returns 0.

**5.145.3.4 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. 588).

**5.145.3.5 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. 588) currently returns false.

#### Postcondition

#### See also

[get\\_server\\_port\\_open](#) (p. 588) returns the **value** (p. 49) of `open`.

**5.145.3.6 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. 589).

**5.145.3.7 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. 589).

## Postcondition

## See also

**get\_connected** (p. 589) returns the **value** (p. 49) of open.

**5.145.3.8 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. 589) currently returns false.

**Postcondition****See also**

**get\_connected** (p. 589) returns true.

**5.145.3.9 debug()** `virtual void io_asterisk_parser_t::debug ( const std::string & message ) [inline], [virtual]`

## 5.145.4 Member Data Documentation

**5.145.4.1 local\_address** `MHAParser::string_t io_asterisk_parser_t::local_address [private]`

Lets the user set the local network interface to listen on.

**5.145.4.2 local\_port** `MHAParser::int_t io_asterisk_parser_t::local_port [private]`

Lets the user choose the local tcp port to listen on.

**5.145.4.3 server\_port\_open** `MHAParser::int_mon_t io_asterisk_parser_t::server_port_open [private]`

Indicates whether the TCP server socket is currently open.

**5.145.4.4 connected** `MHAParser::int_mon_t io_asterisk_parser_t::connected [private]`

Indicator if there currently is a sound data connection over TCP.

**5.145.4.5 peer\_address** `MHAParser::string_mon_t io_asterisk_parser_t::peer_address [private]`

Display the ip address of the currently connected sound data client.

**5.145.4.6 peer\_port** `MHAParser::int_mon_t io_asterisk_parser_t::peer_port [private]`

Display the tcp port used by the current sound data client.

**5.145.4.7 debug\_filename** `MHAParser::string_t io_asterisk_parser_t::debug_filename [private]`

filename to write debugging info to (if non-empty)

**5.145.4.8 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**

## 5.146 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.*

### 5.146.1 Detailed Description

Sound data handling of io tcp library.

### 5.146.2 Constructor & Destructor Documentation

```
5.146.2.1 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.

**5.146.2.2 ~io\_asterisk\_sound\_t()** virtual io\_asterisk\_sound\_t::~io\_asterisk\_sound\_t ( ) [inline], [virtual]

Do-nothing destructor.

### 5.146.3 Member Function Documentation

**5.146.3.1 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.

**5.146.3.2 release()** void io\_asterisk\_sound\_t::release ( ) [virtual]

Called during release.

Deletes sound data storage.

**5.146.3.3 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.

**5.146.3.4 header()** `std::string io_asterisk_sound_t::header ( ) const [virtual]`

Create the tcp sound header lines.

**5.146.3.5 hton()** `std::string & io_asterisk_sound_t::hton ( const mha_wave_t * s_out )`

Serialize data for network transfer.

**5.146.3.6 ntoh()** `mha_wave_t * io_asterisk_sound_t::ntoh ( const std::string & data )`

Deserialize data from network.

**5.146.4 Member Data Documentation****5.146.4.1 fragsize** `int io_asterisk_sound_t::fragsize [private]`

Number of sound samples in each channel expected and returned from processing callback.

**5.146.4.2 samplerate** float io\_asterisk\_sound\_t::samplerate [private]

Sampling rate.

Number of samples per second in each channel.

**5.146.4.3 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.

**5.146.4.4 num\_outchannels** int io\_asterisk\_sound\_t::num\_outchannels [private]**5.146.4.5 s\_in** MHASignal::waveform\_t\* io\_asterisk\_sound\_t::s\_in [private]

Storage for input signal.

**5.146.4.6 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

## 5.147 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, **IOSoppedEvent\_t** stop\_event, void \*stop\_handle)
- void **prepare** (int num\_inchannels, int num\_outchannels)  
*Allocate server socket and start thread waiting for sound data exchange.*
- void **start** ()  
*Call frameworks start callback if there is a sound data connection at the moment.*
- void **stop** ()  
*Close the current connection if there is one.*
- void **release** ()  
*Close the current connection and close the server socket.*
- virtual void **accept\_loop** ()  
*IO thread executes this method.*
- virtual void **connection\_loop** (**MHA\_TCP::Connection** \*c)  
*IO thread executes this method for each connection.*
- virtual void **parse** (const char \*cmd, char \*retval, unsigned int len)  
*Parser interface.*
- virtual ~**io\_asterisk\_t** ()

## Private Attributes

- **io\_asterisk\_parser\_t** parser
- **io\_asterisk\_sound\_t** sound
- **io\_asterisk\_fwcb\_t** fwcb
- **MHA\_TCP::Server** \* server
- **MHA\_TCP::Thread** \* thread
- **MHA\_TCP::Async\_Notify** notify\_start
- **MHA\_TCP::Async\_Notify** notify\_stop
- **MHA\_TCP::Async\_Notify** notify\_release

### 5.147.1 Detailed Description

The tcp sound io library.

### 5.147.2 Constructor & Destructor Documentation

**5.147.2.1 `io_asterisk_t()`** `io_asterisk_t::io_asterisk_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 )`

**5.147.2.2 `~io_asterisk_t()`** `virtual io_asterisk_t::~io_asterisk_t ( ) [inline],`  
`[virtual]`

### 5.147.3 Member Function Documentation

**5.147.3.1 `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

**5.147.3.2 `start()`** `void io_asterisk_t::start ( )`

Call frameworks start callback if there is a sound data connection at the moment.

**5.147.3.3 `stop()`** `void io_asterisk_t::stop ( )`

Close the current connection if there is one.

stop IO thread

**5.147.3.4 release()** void io\_asterisk\_t::release ( )

Close the current connection and close the server socket.

Stop IO thread and close server socket.

**5.147.3.5 accept\_loop()** void io\_asterisk\_t::accept\_loop ( ) [virtual]

IO thread executes this method.

**5.147.3.6 connection\_loop()** void io\_asterisk\_t::connection\_loop ( MHA\_TCP::Connection \* c ) [virtual]

IO thread executes this method for each connection.

**Parameters**

<i>c</i>	pointer to connection. connection_loop deletes connection before exiting.
----------	---

**5.147.3.7 parse()** virtual void io\_asterisk\_t::parse ( const char \* cmd, char \* retval, unsigned int len ) [inline], [virtual]

Parser interface.

**5.147.4 Member Data Documentation****5.147.4.1 parser** io\_asterisk\_parser\_t io\_asterisk\_t::parser [private]

**5.147.4.2 sound** `io_asterisk_sound_t` `io_asterisk_t::sound` [private]

**5.147.4.3 fwcb** `io_asterisk_fwcb_t` `io_asterisk_t::fwcb` [private]

**5.147.4.4 server** `MHA_TCP::Server*` `io_asterisk_t::server` [private]

**5.147.4.5 thread** `MHA_TCP::Thread*` `io_asterisk_t::thread` [private]

**5.147.4.6 notify\_start** `MHA_TCP::Async_Notify` `io_asterisk_t::notify_start` [private]

**5.147.4.7 notify\_stop** `MHA_TCP::Async_Notify` `io_asterisk_t::notify_stop` [private]

**5.147.4.8 notify\_release** `MHA_TCP::Async_Notify` `io_asterisk_t::notify_release` [private]

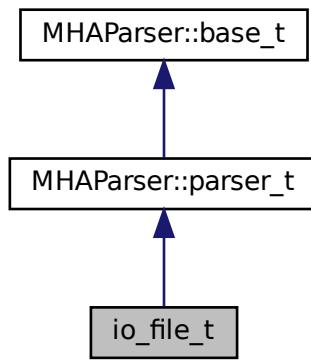
The documentation for this class was generated from the following file:

- **MHAIOAsterisk.cpp**

## 5.148 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**
- **MHAParser::string\_t filename\_input**
- **MHAParser::string\_t filename\_output**
- **MHAParser::kw\_t output\_sample\_format**
- **MHAParser::int\_t startsample**
- **MHAParser::int\_t length**
- **MHAParser::bool\_t strict\_channel\_match**
- **MHAParser::bool\_t strict\_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

### 5.148.1 Detailed Description

File IO.

### 5.148.2 Constructor & Destructor Documentation

```
5.148.2.1 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 )
```

5.148.2.2 **~io\_file\_t()** io\_file\_t::~io\_file\_t ( )

### 5.148.3 Member Function Documentation

```
5.148.3.1 prepare() void io_file_t::prepare (
    int nch_in,
    int nch_out )
```

Allocate buffers, activate FILE client and install internal ports.

5.148.3.2 **start()** void io\_file\_t::start ( )

5.148.3.3 **stop()** void io\_file\_t::stop ( )

5.148.3.4 **release()** void io\_file\_t::release ( )

Remove FILE client and deallocate internal ports and buffers.

**5.148.3.5 `stopped()`** void io\_file\_t::stopped ( int proc\_err, int io\_err ) [private]

**5.148.3.6 `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.
---------------	--

### 5.148.4 Member Data Documentation

**5.148.4.1 `fragsize`** int io\_file\_t::fragsize [private]

**5.148.4.2 `samplerate`** float io\_file\_t::samplerate [private]

**5.148.4.3 `nchannels_in`** int io\_file\_t::nchannels\_in [private]

**5.148.4.4 `nchannels_file_in`** int io\_file\_t::nchannels\_file\_in [private]

**5.148.4.5 nchannels\_out** int io\_file\_t::nchannels\_out [private]

**5.148.4.6 proc\_event** IOProcessEvent\_t io\_file\_t::proc\_event [private]

**5.148.4.7 proc\_handle** void\* io\_file\_t::proc\_handle [private]

**5.148.4.8 start\_event** IOStartedEvent\_t io\_file\_t::start\_event [private]

**5.148.4.9 start\_handle** void\* io\_file\_t::start\_handle [private]

**5.148.4.10 stop\_event** IOStoppedEvent\_t io\_file\_t::stop\_event [private]

**5.148.4.11 stop\_handle** void\* io\_file\_t::stop\_handle [private]

**5.148.4.12 filename\_input** MHAParser::string\_t io\_file\_t::filename\_input [private]

**5.148.4.13 filename\_output** MHAParser::string\_t io\_file\_t::filename\_output [private]

**5.148.4.14 output\_sample\_format** `MHAParser::kw_t io_file_t::output_sample_format` [private]

**5.148.4.15 startsample** `MHAParser::int_t io_file_t::startsample` [private]

**5.148.4.16 length** `MHAParser::int_t io_file_t::length` [private]

**5.148.4.17 strict\_channel\_match** `MHAParser::bool_t io_file_t::strict_channel_match` [private]

**5.148.4.18 strict\_srate\_match** `MHAParser::bool_t io_file_t::strict_srate_match` [private]

**5.148.4.19 s\_in** `MHASignal::waveform_t* io_file_t::s_in` [private]

**5.148.4.20 s\_file\_in** `MHASignal::waveform_t* io_file_t::s_file_in` [private]

**5.148.4.21 s\_out** `mha_wave_t* io_file_t::s_out` [private]

**5.148.4.22 b\_prepared** `bool io_file_t::b_prepared` [private]

**5.148.4.23 sf\_in** SNDFILE\* io\_file\_t::sf\_in [private]

**5.148.4.24 sf\_out** SNDFILE\* io\_file\_t::sf\_out [private]

**5.148.4.25 sfinf\_in** SF\_INFO io\_file\_t::sfinf\_in [private]

**5.148.4.26 sfinf\_out** SF\_INFO io\_file\_t::sfinf\_out [private]

**5.148.4.27 total\_read** sf\_count\_t io\_file\_t::total\_read [private]

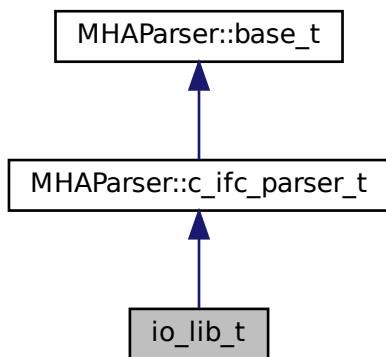
The documentation for this class was generated from the following file:

- **MHAIOfFile.cpp**

## 5.149 **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)
- **std::string get\_documentation () const**
- **std::vector< std::string > get\_categories () const**

## Protected Member Functions

- **void test\_error ()**

## Protected Attributes

- **int lib\_err**
- **pluginlib\_t lib\_handle**
- **void \* lib\_data**
- **IOInit\_t IOInit\_cb**
- **IOPrepare\_t IOPrepare\_cb**
- **IOStart\_t IOStart\_cb**
- **IOStop\_t IOStop\_cb**
- **IOResume\_t IOResume\_cb**
- **IOSetVar\_t IOSetVar\_cb**
- **IOStrError\_t IOStrError\_cb**
- **IODestroy\_t IODestroy\_cb**
- **std::string plugin\_documentation**
- **std::vector< std::string > plugin\_categories**

## Additional Inherited Members

### 5.149.1 Detailed Description

Class for loading MHA sound IO module.

## 5.149.2 Constructor & Destructor Documentation

```
5.149.2.1 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,
    IOStoppedEvent_t stop_event,
    void * stop_handle,
    std::string libname )
```

load and initialize MHA sound io module.

```
5.149.2.2 ~io_lib_t() io_lib_t::~io_lib_t ( )
```

Deinitialize and unload this MHA sound io module.

## 5.149.3 Member Function Documentation

```
5.149.3.1 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

**5.149.3.2 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.

**5.149.3.3 stop()** void io\_lib\_t::stop ( )**5.149.3.4 release()** void io\_lib\_t::release ( )**5.149.3.5 lib\_str\_error()** std::string io\_lib\_t::lib\_str\_error ( int err )**5.149.3.6 get\_documentation()** std::string io\_lib\_t::get\_documentation ( ) const [inline]**5.149.3.7 get\_categories()** std::vector<std::string> io\_lib\_t::get\_categories ( ) const [inline]**5.149.3.8 test\_error()** void io\_lib\_t::test\_error ( ) [protected]**5.149.4 Member Data Documentation**

**5.149.4.1 lib\_err** int io\_lib\_t::lib\_err [protected]

**5.149.4.2 lib\_handle** pluginlib\_t io\_lib\_t::lib\_handle [protected]

**5.149.4.3 lib\_data** void\* io\_lib\_t::lib\_data [protected]

**5.149.4.4 IOInit\_cb** IOInit\_t io\_lib\_t::IOInit\_cb [protected]

**5.149.4.5 IOPrepare\_cb** IOPrepare\_t io\_lib\_t::IOPrepare\_cb [protected]

**5.149.4.6 IOStart\_cb** IOStart\_t io\_lib\_t::IOStart\_cb [protected]

**5.149.4.7 IOStop\_cb** IOStop\_t io\_lib\_t::IOStop\_cb [protected]

**5.149.4.8 IOResume\_cb** IOResume\_t io\_lib\_t::IOResume\_cb [protected]

**5.149.4.9 IOSetVar\_cb** IOSetVar\_t io\_lib\_t::IOSetVar\_cb [protected]

**5.149.4.10 IOStrError\_cb** `iostrError_t io_lib_t::IOStrError_cb` [protected]

**5.149.4.11 IODestroy\_cb** `IODestroy_t io_lib_t::IODestroy_cb` [protected]

**5.149.4.12 plugin\_documentation** `std::string io_lib_t::plugin_documentation` [protected]

**5.149.4.13 plugin\_categories** `std::vector<std::string> io_lib_t::plugin_categories` [protected]

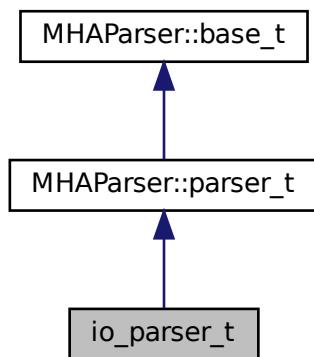
The documentation for this class was generated from the following files:

- `mhafw_lib.h`
- `mhafw_lib.cpp`

## 5.150 `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

### 5.150.1 Detailed Description

Main class for Parser IO.

## 5.150.2 Constructor & Destructor Documentation

**5.150.2.1 `io_parser_t()`** `io_parser_t::io_parser_t (`  
    `unsigned int fragsize,`  
    `IOPProcessEvent_t proc_event,`  
    `void * proc_handle,`  
    `IOStartedEvent_t start_event,`  
    `void * start_handle,`  
    `IOStoppedEvent_t stop_event,`  
    `void * stop_handle )`

**5.150.2.2 `~io_parser_t()`** `io_parser_t::~io_parser_t ( )`

## 5.150.3 Member Function Documentation

**5.150.3.1 `prepare()`** `void io_parser_t::prepare (`  
    `int nch_in,`  
    `int nch_out )`

Allocate buffers, activate JACK client and install internal ports.

**5.150.3.2 `start()`** `void io_parser_t::start ( )`

**5.150.3.3 `stop()`** `void io_parser_t::stop ( )`

**5.150.3.4 **release()**** void io\_parser\_t::release ( )

Remove JACK client and deallocate internal ports and buffers.

**5.150.3.5 **stopped()**** void io\_parser\_t::stopped ( int proc\_err, int io\_err ) [private]**5.150.3.6 **started()**** void io\_parser\_t::started ( ) [private]**5.150.3.7 **process\_frame()**** void io\_parser\_t::process\_frame ( ) [private]**5.150.4 Member Data Documentation****5.150.4.1 **fragsize**** unsigned int io\_parser\_t::fragsize [private]**5.150.4.2 **nchannels\_in**** unsigned int io\_parser\_t::nchannels\_in [private]**5.150.4.3 **nchannels\_out**** unsigned int io\_parser\_t::nchannels\_out [private]**5.150.4.4 **proc\_event**** IOProcessEvent\_t io\_parser\_t::proc\_event [private]

**5.150.4.5 proc\_handle** void\* io\_parser\_t::proc\_handle [private]

**5.150.4.6 start\_event** IOStartedEvent\_t io\_parser\_t::start\_event [private]

**5.150.4.7 start\_handle** void\* io\_parser\_t::start\_handle [private]

**5.150.4.8 stop\_event** IOSToppedEvent\_t io\_parser\_t::stop\_event [private]

**5.150.4.9 stop\_handle** void\* io\_parser\_t::stop\_handle [private]

**5.150.4.10 input** MHAParser::mfloat\_t io\_parser\_t::input [private]

**5.150.4.11 output** MHAParser::mfloat\_mon\_t io\_parser\_t::output [private]

**5.150.4.12 s\_in** MHASignal::waveform\_t\* io\_parser\_t::s\_in [private]

**5.150.4.13 s\_out** mha\_wave\_t\* io\_parser\_t::s\_out [private]

**5.150.4.14 b\_fw\_started** bool io\_parser\_t::b\_fw\_started [private]

**5.150.4.15 b\_stopped** bool io\_parser\_t::b\_stopped [private]

**5.150.4.16 b\_prepared** bool io\_parser\_t::b\_prepared [private]

**5.150.4.17 b\_starting** bool io\_parser\_t::b\_starting [private]

**5.150.4.18 patchbay** MHAEvents::patchbay\_t< io\_parser\_t> io\_parser\_t::patchbay [private]

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

- **MHAIOParser.cpp**

## 5.151 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**

### 5.151.1 Detailed Description

TCP sound-io library's interface to the framework callbacks.

### 5.151.2 Constructor & Destructor Documentation

```
5.151.2.1 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.

```
5.151.2.2 ~io_tcp_fwcb_t() virtual io_tcp_fwcb_t::~io_tcp_fwcb_t ( ) [inline],
[virtual]
```

Do-nothing destructor.

### 5.151.3 Member Function Documentation

#### 5.151.3.1 **start()** void io\_tcp\_fwcb\_t::start ( ) [virtual]

Call the framework's start callback.

#### 5.151.3.2 **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.

#### 5.151.3.3 **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. 620)

**Parameters**

<i>proc_err</i>	The error number from the
-----------------	---------------------------

**See also**

**process** (p. 619) callback.

**Parameters**

<i>io_err</i>	The error number from the io library itself.
---------------	--

**5.151.3.4 stop()** `void io_tcp_fwcb_t::stop ( ) [virtual]`

Call the frameworks stop callback.

Uses the error numbers set previously with

**See also**

**set\_errno** (p. 619).

**5.151.4 Member Data Documentation****5.151.4.1 proc\_event** `IOProcessEvent_t io_tcp_fwcb_t::proc_event [private]`

Pointer to signal processing callback function.

**5.151.4.2 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.

**5.151.4.3 stop\_event** `ioStoppedEvent_t io_tcp_fwcb_t::stop_event [private]`

Pointer to stop notification callback function.

Called when the connection is closed.

**5.151.4.4 proc\_handle** `void* io_tcp_fwcb_t::proc_handle [private]`

Handles belonging to framework.

**5.151.4.5 start\_handle** `void * io_tcp_fwcb_t::start_handle [private]`**5.151.4.6 stop\_handle** `void * io_tcp_fwcb_t::stop_handle [private]`**5.151.4.7 proc\_err** `int io_tcp_fwcb_t::proc_err [private]`

Errors from the processing callback and from the TCP IO itself are stored here before closing Network handles.

MHAIOTCP is notified by the server when the connection has been taken down, and calls

See also

[stop](#) (p. 620) from that callback. Within [stop](#) (p. 620), these error numbers are read again and transmitted to the framework.

**5.151.4.8 io\_err** `int io_tcp_fwcb_t::io_err [private]`

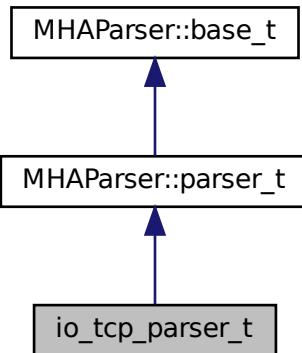
The documentation for this class was generated from the following file:

- [MHAIOTCP.cpp](#)

## 5.152 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

### 5.152.1 Detailed Description

The parser interface of the IOTCP library.

### 5.152.2 Constructor & Destructor Documentation

#### 5.152.2.1 **io\_tcp\_parser\_t()** `io_tcp_parser_t::io_tcp_parser_t ( )`

Constructor initializes parser variables.

#### 5.152.2.2 **~io\_tcp\_parser\_t()** `virtual io_tcp_parser_t::~io_tcp_parser_t ( ) [inline], [virtual]`

Do-nothing destructor.

### 5.152.3 Member Function Documentation

**5.152.3.1 `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.

**5.152.3.2 `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.

**5.152.3.3 `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. 624) currently returns 0.

**5.152.3.4 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. 625).

**5.152.3.5 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. 625) currently returns false.

**Postcondition****See also**

**get\_server\_port\_open** (p. 625) returns the **value** (p. 49) of `open`.

**5.152.3.6 `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. 626).

**5.152.3.7 `set_connected()`** `void io_tcp_parser_t::set_connected ( bool connected ) [virtual]`

Inform the parser about the existence of a sound data connection.

**Parameters**

<code>connected</code>	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. 625).

**Postcondition****See also**

**`get_connected`** (p. 625) returns the **value** (p. 49) of open.

**5.152.3.8 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. 625) currently returns false.

#### Postcondition

#### See also

**get\_connected** (p. 625) returns true.

**5.152.3.9 debug()** virtual void io\_tcp\_parser\_t::debug ( const std::string & *message* ) [inline], [virtual]

## 5.152.4 Member Data Documentation

**5.152.4.1 local\_address** **MHAParser::string\_t** io\_tcp\_parser\_t::local\_address [private]

Lets the user set the local network interface to listen on.

**5.152.4.2 local\_port** `MHAParser::int_t io_tcp_parser_t::local_port [private]`

Lets the user choose the local tcp port to listen on.

**5.152.4.3 server\_port\_open** `MHAParser::int_mon_t io_tcp_parser_t::server_port_open [private]`

Indicates wether the TCP server socket is currently open.

**5.152.4.4 connected** `MHAParser::int_mon_t io_tcp_parser_t::connected [private]`

Indicator if there currently is a sound data connection over TCP.

**5.152.4.5 peer\_address** `MHAParser::string_mon_t io_tcp_parser_t::peer_address [private]`

Display the ip address of the currently connected sound data client.

**5.152.4.6 peer\_port** `MHAParser::int_mon_t io_tcp_parser_t::peer_port [private]`

Display the tcp port used by the current sound data client.

**5.152.4.7 debug\_filename** `MHAParser::string_t io_tcp_parser_t::debug_filename [private]`

filename to write debugging info to (if non-empty)

#### 5.152.4.8 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**

## 5.153 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. 836) structure.*
- virtual std::string **hton (const mha\_wave\_t \*s\_out)**  
*Copy sound data from the output sound structure to a string.*

### Static Private Member Functions

- static void **check\_sound\_data\_type ()**  
*Check if **mha\_real\_t** is a usable 32-bit floating point type.*

## Private Attributes

- int **fragsize**  
*Number of sound samples in each channel expected and returned from processing callback.*
- float **samplerate**  
*Sampling rate.*
- int **num\_inchannels**  
*Number of input channels.*
- int **num\_outchannels**
- **MHASignal::waveform\_t \* s\_in**  
*Storage for input signal.*

### 5.153.1 Detailed Description

Sound data handling of io tcp library.

### 5.153.2 Constructor & Destructor Documentation

```
5.153.2.1 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. 631).

#### Parameters

<i>fragsize</i>	Number of sound samples in each channel expected and returned from processing callback.
<i>samplerate</i>	Number of samples per second in each channel.

**5.153.2.2 ~io\_tcp\_sound\_t()** virtual io\_tcp\_sound\_t::~io\_tcp\_sound\_t ( ) [inline], [virtual]

Do-nothing destructor.

### 5.153.3 Member Function Documentation

**5.153.3.1 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

<b>MHA_Error</b> (p. <a href="#">760</a> )	if mha_real_t is not compatible to 32-bit float.
--	--

**5.153.3.2 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.

**5.153.3.3 release()** void io\_tcp\_sound\_t::release ( ) [virtual]

Called during release.

Deletes sound data storage.

**5.153.3.4 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.

**5.153.3.5 header()** `std::string io_tcp_sound_t::header ( ) const [virtual]`

Create the tcp sound header lines.

**5.153.3.6 ntoh()** `mha_wave_t * io_tcp_sound_t::ntoh ( const std::string & data ) [virtual]`

Copy data received from tcp into **mha\_wave\_t** (p. 836) structure.

Doing network-to-host byte order swapping in the process.

**Parameters**

<code>data</code>	One chunk (
-------------------	-------------

**See also**

**chunkbytes\_in** (p. 631)) of sound data to process.

**Returns**

Pointer to the sound data storage.

**5.153.3.7 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.

**5.153.4 Member Data Documentation****5.153.4.1 fragsize int io\_tcp\_sound\_t::fragsize [private]**

Number of sound samples in each channel expected and returned from processing callback.

**5.153.4.2 samplerate float io\_tcp\_sound\_t::samplerate [private]**

Sampling rate.

Number of samples per second in each channel.

**5.153.4.3 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.

**5.153.4.4 num\_outchannels int io\_tcp\_sound\_t::num\_outchannels [private]****5.153.4.5 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**

## 5.154 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]

#### 5.154.1 Detailed Description

This union helps in conversion of floats from host byte order to network byte order and back again.

#### 5.154.2 Member Data Documentation

##### 5.154.2.1 **f** float io\_tcp\_sound\_t::float\_union::f

##### 5.154.2.2 **i** unsigned int io\_tcp\_sound\_t::float\_union::i

##### 5.154.2.3 **c** char io\_tcp\_sound\_t::float\_union::c[4]

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

- MHAIOTCP.cpp

## 5.155 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

### 5.155.1 Detailed Description

The tcp sound io library.

### 5.155.2 Constructor & Destructor Documentation

```
5.155.2.1 io_tcp_t() io_tcp_t::io_tcp_t (
    int fragsize,
    float samplerate,
    IOProcessEvent_t proc_event,
    void * proc_handle,
    IOStartedEvent_t start_event,
    void * start_handle,
    IOStoppedEvent_t stop_event,
    void * stop_handle )
```

5.155.2.2 **~io\_tcp\_t()** virtual io\_tcp\_t::~io\_tcp\_t ( ) [inline], [virtual]

### 5.155.3 Member Function Documentation

```
5.155.3.1 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

5.155.3.2 **start()** void io\_tcp\_t::start ( )

Call frameworks start callback if there is a sound data connection at the moment.

5.155.3.3 **stop()** void io\_tcp\_t::stop ( )

Close the current connection if there is one.

stop IO thread

**5.155.3.4 release()** void io\_tcp\_t::release ( )

Close the current connection and close the server socket.

Stop IO thread and close server socket.

**5.155.3.5 accept\_loop()** void io\_tcp\_t::accept\_loop ( ) [virtual]

IO thread executes this method.

**5.155.3.6 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.
----------	---

**5.155.3.7 parse()** virtual void io\_tcp\_t::parse (   
 const char \* cmd,   
 char \* retval,   
 unsigned int len ) [inline], [virtual]

Parser interface.

**5.155.4 Member Data Documentation****5.155.4.1 parser** io\_tcp\_parser\_t io\_tcp\_t::parser [private]

**5.155.4.2 sound** `io_tcp_sound_t` `io_tcp_t::sound` [private]

**5.155.4.3 fwcb** `io_tcp_fwcb_t` `io_tcp_t::fwcb` [private]

**5.155.4.4 server** `MHA_TCP::Server*` `io_tcp_t::server` [private]

**5.155.4.5 thread** `MHA_TCP::Thread*` `io_tcp_t::thread` [private]

**5.155.4.6 notify\_start** `MHA_TCP::Async_Notify` `io_tcp_t::notify_start` [private]

**5.155.4.7 notify\_stop** `MHA_TCP::Async_Notify` `io_tcp_t::notify_stop` [private]

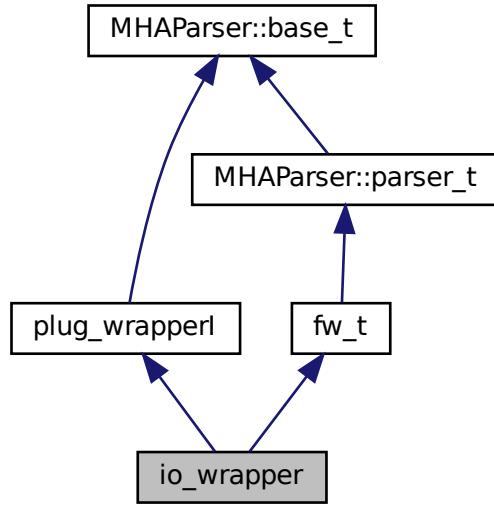
**5.155.4.8 notify\_release** `MHA_TCP::Async_Notify` `io_tcp_t::notify_release` [private]

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

- **MHAIOTCP.cpp**

## 5.156 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

#### 5.156.1 Constructor & Destructor Documentation

**5.156.1.1 `io_wrapper()`** `io_wrapper::io_wrapper (`  
    `algo_comm_t iac,`  
    `const std::string & libname ) [inline]`

**5.156.1.2 ~io\_wrapper()** virtual io\_wrapper::~io\_wrapper ( ) [virtual], [default]

## 5.156.2 Member Function Documentation

**5.156.2.1 get\_categories()** virtual std::vector<std::string> io\_wrapper::get\_←  
categories ( ) [inline], [virtual]

Implements **plug\_wrapperl** (p. 1349).

**5.156.2.2 parse()** virtual std::string io\_wrapper::parse ( const std::string & str ) [inline], [virtual]

Implements **plug\_wrapperl** (p. 1349).

**5.156.2.3 has\_parser()** virtual bool io\_wrapper::has\_parser ( ) [inline], [virtual]

Implements **plug\_wrapperl** (p. 1349).

**5.156.2.4 get\_documentation()** virtual std::string io\_wrapper::get\_documentation ( ) [inline], [virtual]

Implements **plug\_wrapperl** (p. 1349).

**5.156.2.5 has\_process()** virtual bool io\_wrapper::has\_process ( mha\_domain\_t in, mha\_domain\_t out ) [inline], [virtual]

Implements **plug\_wrapperl** (p. 1350).

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

- **generatemhaplugindoc.cpp**

## 5.157 `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 & **plugname**, const std::string & **plugin\_macro**)  
*Constructor loads the plugin into this process.*
- std::string `get_latex_doc` ()  
*This method accesses the compiled-in contents of the MHAPLUGIN\_DOCUMENTATION macro and the exported interface functions of the loaded plugin to produce latex 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\_wrapper** > **loader**
- const std::string **plugin\_macro**

### 5.157.1 Detailed Description

Class to access the information stored in the plugin source code's MHAPLUGIN\_DOCUMENTATION macro.

### 5.157.2 Constructor & Destructor Documentation

#### 5.157.2.1 `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>pluginname</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", ...)

## 5.157.3 Member Function Documentation

### 5.157.3.1 **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

### 5.157.3.2 **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

### 5.157.3.3 **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

**5.157.3.4 strdom()** std::string latex\_doc\_t::strdom (   
   **mha\_domain\_t** *d* ) const [private]

**5.157.3.5 get\_ac()** std::string latex\_doc\_t::get\_ac (   
   **MHAKernel::algo\_comm\_class\_t** & *ac*,   
   std::string *txt* ) const [private]

**5.157.3.6 parsername()** std::string latex\_doc\_t::parsername (   
   std::string *s* ) const [private]

**5.157.3.7 get\_parser\_var()** std::string latex\_doc\_t::get\_parser\_var (   
   **MHAParser::base\_t** \* *p*,   
   std::string *name* ) const [private]

**5.157.3.8 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]

## 5.157.4 Member Data Documentation

**5.157.4.1 plugname** const std::string latex\_doc\_t::plugname [private]

**5.157.4.2 latex\_plugname** const std::string latex\_doc\_t::latex\_plugname [private]

**5.157.4.3 ac** `MHAKernel::algo_comm_class_t` `latex_doc_t::ac` [private]

**5.157.4.4 loader** `std::unique_ptr< plug_wrapperI>` `latex_doc_t::loader` [private]

**5.157.4.5 plugin\_macro** `const std::string` `latex_doc_t::plugin_macro` [private]

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

- `generatemhaplugindoc.cpp`

## 5.158 `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::o1filt\_lowpass\_t mismatch**

*Low-pass filtered level mismatch.*

### 5.158.1 Constructor & Destructor Documentation

**5.158.1.1 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> $\leftarrow$ <i>constant_</i>	Time constant of low pass filter

**5.158.1.2 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> $\leftarrow$ <i>constant_</i>	Time constant of low pass filter

**5.158.2 Member Function Documentation****5.158.2.1 update\_mismatch() [1/2]** `mha_real_t level_matching::channel_pair::update $\leftarrow$   
_mismatch (`

```
    const mha_wave_t & signal_ )
```

Calculates the filtered rms level mismatch.

**Parameters**

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

**Returns**

Low-pass filtered mismatch ratio

```
5.158.2.2 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

```
5.158.2.3 get_mismatch() mha_real_t level_matching::channel_pair::get_mismatch (   
 ) const
```

Get last filter result.

**Returns**

Last filter result

```
5.158.2.4 get_idx() const std::pair<int,int>& level_matching::channel_pair::get_idx  
( ) const [inline]
```

## 5.158.3 Member Data Documentation

### 5.158.3.1 **idx** std::pair<int,int> level\_matching::channel\_pair::idx [private]

Indices of channels.

### 5.158.3.2 **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**

## 5.159 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*

## 5.159.1 Constructor & Destructor Documentation

```
5.159.1.1 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.

```
5.159.1.2 ~level_matching_config_t() level_matching::level_matching_config_t::~level←
_matching_config_t ( ) [default]
```

## 5.159.2 Member Function Documentation

**5.159.2.1 process() [1/2]** `mha_wave_t * level_matching::level_matching_config_t::process ( mha_wave_t * s )`

**5.159.2.2 process() [2/2]** `mha_spec_t * level_matching::level_matching_config_t::process ( mha_spec_t * s )`

### **5.159.3 Member Data Documentation**

**5.159.3.1 tmp** `MHASignal::waveform_t level_matching::level_matching_config_t::tmp` [private]

Temporary storage for input signal to use waveform\_t helper functions.

**5.159.3.2 pairings** `std::vector< channel_pair > level_matching::level_matching_config_t::pairings` [private]

Channel pairings.

**5.159.3.3 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.

**5.159.3.4 range** `mha_real_t level_matching::level_matching_config_t::range` [private]

Maximum matching range. Maximum adjustment done.

### 5.159.3.5 **ffflen** `unsigned level_matching::level_matching_config_t::ffflen [private]`

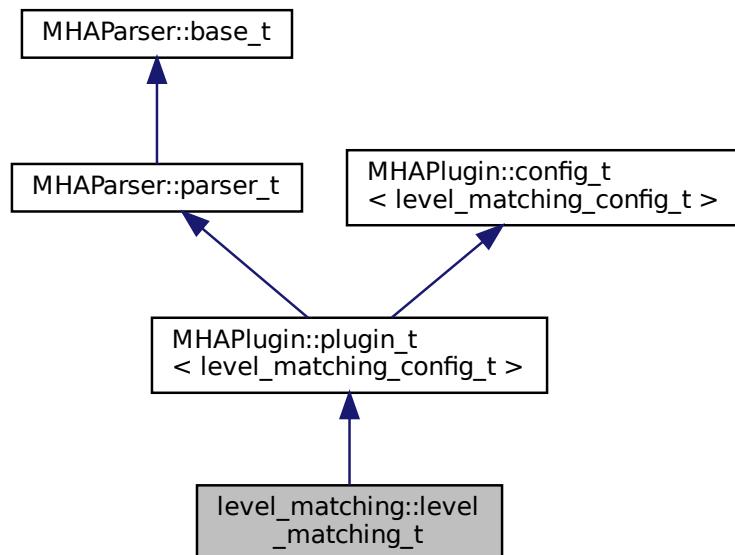
ffflen in spectral domain

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

- **level\_matching.hh**
- **level\_matching.cpp**

## 5.160 **level\_matching::level\_matching\_t Class Reference**

Inheritance diagram for `level_matching::level_matching_t`:



### Public Member Functions

- **level\_matching\_t ( algo\_comm\_t iac, const std::string &configured\_name)**  
*Plugin interface ctor.*
- **~level\_matching\_t ()**
- **mha\_wave\_t \* process ( mha\_wave\_t \*)**  
*Processing callback.*
- **mha\_spec\_t \* process ( mha\_spec\_t \*)**  
*Processing callback.*
- **void prepare ( mhaconfig\_t &)**  
*Plugin preparation callback.*
- **void release (void)**

## Private Member Functions

- void **update\_cfg ()**

## Private Attributes

- **MHAParser::mint\_t channels** ={"channels","[[0 1]]"}
- **MHAParser::float\_t range** ={"Maximum matching range in dB","4","[0,["]}
- **MHAParser::float\_t lp\_signal\_fpass** ={"Upper edge of the lp pass band for the signal in Hz","4000","[0,["]}
- **MHAParser::float\_t lp\_signal\_fstop** ={"Stop band lower edge frequency for the signal in Hz","8000","[0,["]}
- **MHAParser::float\_t lp\_signal\_order** ={"Signal lp order","24","[1,["]}
- **MHAParser::float\_t lp\_level\_tau** ={"Low pass time constant for the mismatch in s","1","[0,["]}
- **MHAEvents::patchbay\_t< level\_matching\_t > patchbay**

## Additional Inherited Members

### 5.160.1 Constructor & Destructor Documentation

**5.160.1.1 level\_matching\_t()** level\_matching::level\_matching\_t::level\_matching\_t (   
     algo\_comm\_t iac,  
     const std::string & configured\_name )

Plugin interface ctor.

#### Parameters

ac	
----	--

**5.160.1.2 ~level\_matching\_t()** level\_matching::level\_matching\_t::~level\_matching\_t ( )

### 5.160.2 Member Function Documentation

---

**5.160.2.1 `process()` [1/2]** `mha_wave_t * level_matching::level_matching_t::process ( mha_wave_t * signal_ )`

Processing callback.

**Parameters**

<code>signal</code>	Input signal
---------------------	--------------

**5.160.2.2 `process()` [2/2]** `mha_spec_t * level_matching::level_matching_t::process ( mha_spec_t * signal_ )`

Processing callback.

**Parameters**

<code>signal</code>	Input signal
---------------------	--------------

**5.160.2.3 `prepare()`** `void level_matching::level_matching_t::prepare ( mhaconfig_t & ) [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. [1149](#)).

**5.160.2.4 `release()`** `void level_matching::level_matching_t::release ( void ) [inline], [virtual]`

Reimplemented from `MHAPlugin::plugin_t< level_matching_config_t >` (p. [1150](#)).

**5.160.2.5 update\_cfg()** void level\_matching::level\_matching\_t::update\_cfg ( ) [private]

### 5.160.3 Member Data Documentation

**5.160.3.1 channels** **MHAParser::mint\_t** level\_matching::level\_matching\_t::channels  
= {"channels", "[[0 1]]"} [private]

**5.160.3.2 range** **MHAParser::float\_t** level\_matching::level\_matching\_t::range = {"Maximum matching range in dB", "4", "[0,["} [private]

**5.160.3.3 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]

**5.160.3.4 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]

**5.160.3.5 lp\_signal\_order** **MHAParser::float\_t** level\_matching::level\_matching\_t::lp\_signal\_order = {"Signal lp order", "24", "[1,["} [private]

**5.160.3.6 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]

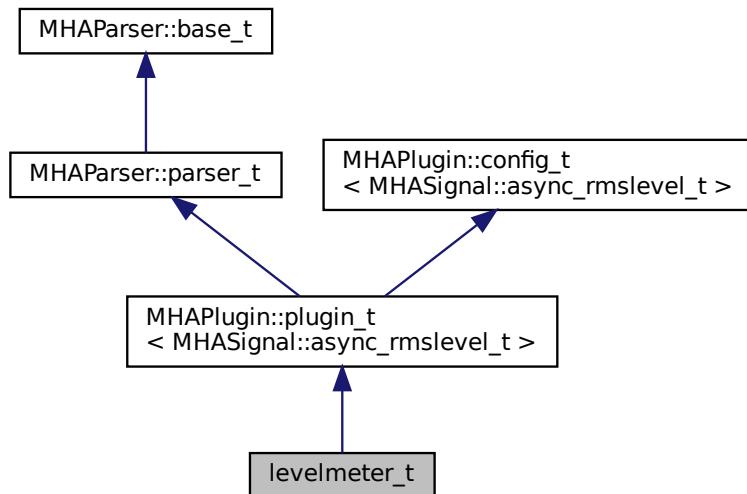
**5.160.3.7 patchbay** `MHAEEvents::patchbay_t< level_matching_t> level_matching< :>;`  
`::level_matching_t::patchbay [private]`

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

- `level_matching.hh`
- `level_matching.cpp`

## 5.161 levelmeter\_t Class Reference

Inheritance diagram for levelmeter\_t:



### Public Member Functions

- `levelemeter_t ( algo_comm_t iac, const std::string &configured_name )`
- `mha_wave_t * process ( mha_wave_t * )`
- `void prepare ( mhaconfig_t & )`

### Private Member Functions

- `void update_tau ()`
- `void query_rms ()`
- `void query_peak ()`

## Private Attributes

- `MHAParser::float_t tau`
- `MHAParser::kw_t mode`
- `MHAParser::vfloat_mon_t rms`
- `MHAParser::vfloat_mon_t peak`
- `MHAEvents::patchbay_t< levelmeter_t > patchbay`

## Additional Inherited Members

### 5.161.1 Constructor & Destructor Documentation

```
5.161.1.1 levelmeter_t() levelmeter_t::levelmeter_t (
    algo_comm_t iac,
    const std::string & configured_name )
```

### 5.161.2 Member Function Documentation

```
5.161.2.1 process() mha_wave_t * levelmeter_t::process (
    mha_wave_t * s )
```

```
5.161.2.2 prepare() void levelmeter_t::prepare (
    mhaconfig_t & cf ) [virtual]
```

Implements `MHAPlugin::plugin_t< MHASignal::async_rmslevel_t >` (p. [1149](#)).

```
5.161.2.3 update_tau() void levelmeter_t::update_tau ( ) [private]
```

**5.161.2.4 query\_rms()** void levelmeter\_t::query\_rms ( ) [private]

**5.161.2.5 query\_peak()** void levelmeter\_t::query\_peak ( ) [private]

### 5.161.3 Member Data Documentation

**5.161.3.1 tau** MHAParser::float\_t levelmeter\_t::tau [private]

**5.161.3.2 mode** MHAParser::kw\_t levelmeter\_t::mode [private]

**5.161.3.3 rms** MHAParser::vfloat\_mon\_t levelmeter\_t::rms [private]

**5.161.3.4 peak** MHAParser::vfloat\_mon\_t levelmeter\_t::peak [private]

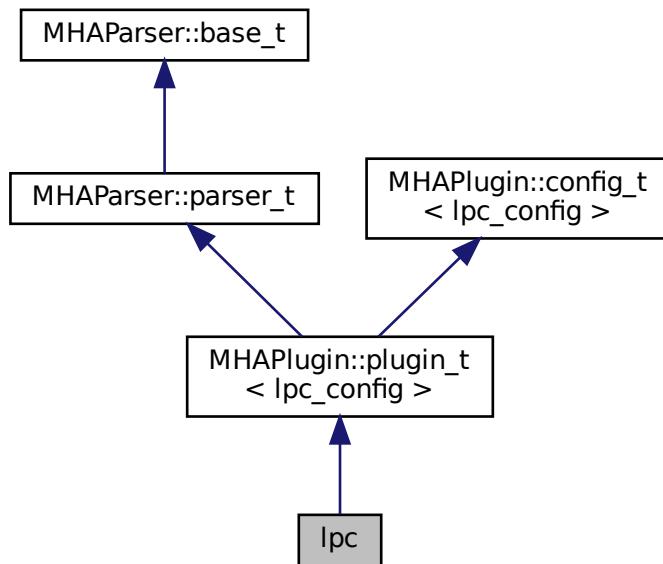
**5.161.3.5 patchbay** MHAEvents::patchbay\_t< levelmeter\_t> levelmeter\_t::patchbay [private]

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

- **levelmeter.cpp**

## 5.162 Ipc Class Reference

Inheritance diagram for Ipc:



### Public Member Functions

- **Ipc ( algo\_comm\_t iac, const std::string &configured\_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****5.162.1 Constructor & Destructor Documentation**

**5.162.1.1 `lpc()`** lpc::lpc (   
                   algo\_comm\_t iac,  
                   const std::string & configured\_name )

Constructs our plugin.

**5.162.1.2 `~lpc()`** ~lpc::~lpc ( )

**5.162.2 Member Function Documentation**

**5.162.2.1 `process()`** mha\_wave\_t \* lpc::process (   
                   mha\_wave\_t \* signal )

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

**5.162.2.2 `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. 1149).

**5.162.2.3 release()** void lpc::release (  
void ) [inline], [virtual]

Reimplemented from **MHAPlugIn::plugin\_t< lpc\_config >** (p. 1150).

**5.162.2.4 update\_cfg()** void lpc::update\_cfg ( ) [private]

### 5.162.3 Member Data Documentation

**5.162.3.1 algo\_name** std::string lpc::algo\_name [private]

**5.162.3.2 lpc\_order** MHAParser::int\_t lpc::lpc\_order [private]

**5.162.3.3 lpc\_buffer\_size** MHAParser::int\_t lpc::lpc\_buffer\_size [private]

**5.162.3.4 shift** MHAParser::bool\_t lpc::shift [private]

**5.162.3.5 comp\_each\_iter** `MHAParser::int_t lpc::comp_each_iter [private]`

**5.162.3.6 norm** `MHAParser::bool_t lpc::norm [private]`

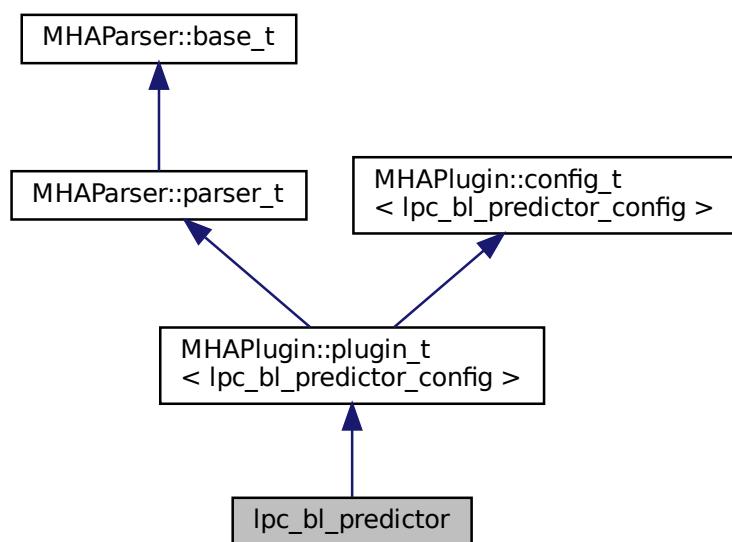
**5.162.3.7 patchbay** `MHAEEvents::patchbay_t< lpc> lpc::patchbay [private]`

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

- `lpc.h`
- `lpc.cpp`

## 5.163 lpc\_bl\_predictor Class Reference

Inheritance diagram for `lpc_bl_predictor`:



## Public Member Functions

- **lpc\_bl\_predictor ( algo\_comm\_t iac, const std::string &configured\_name)**  
*Constructs our plugin.*
- **~lpc\_bl\_predictor ()**
- **mha\_wave\_t \* process ( mha\_wave\_t \*)**  
*Checks for the most recent configuration and defers processing to it.*
- **void prepare ( mhaconfig\_t &)**  
*Plugin preparation.*
- **void release (void)**

## Public Attributes

- **MHAParser::int\_t lpc\_order**
- **MHAParser::string\_t name\_kappa**
- **MHAParser::string\_t name\_lpc\_f**
- **MHAParser::string\_t name\_lpc\_b**
- **MHAParser::string\_t name\_f**
- **MHAParser::string\_t name\_b**

## Private Member Functions

- **void update\_cfg ()**

## Private Attributes

- **MHAEvents::patchbay\_t< lpc\_bl\_predictor > patchbay**

## Additional Inherited Members

### 5.163.1 Constructor & Destructor Documentation

```
5.163.1.1 lpc_bl_predictor() lpc_bl_predictor::lpc_bl_predictor ( 
    algo_comm_t iac,
    const std::string & configured_name )
```

Constructs our plugin.

---

**5.163.1.2 ~lpc\_bl\_predictor()** `lpc_bl_predictor::~lpc_bl_predictor ()`

## 5.163.2 Member Function Documentation

**5.163.2.1 process()** `mha_wave_t * lpc_bl_predictor::process ( mha_wave_t * signal )`

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

**5.163.2.2 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. [1149](#)).

**5.163.2.3 release()** `void lpc_bl_predictor::release ( void ) [inline], [virtual]`

Reimplemented from **MHAPlugin::plugin\_t< lpc\_bl\_predictor\_config >** (p. [1150](#)).

**5.163.2.4 update\_cfg()** `void lpc_bl_predictor::update_cfg ( ) [private]`

### 5.163.3 Member Data Documentation

**5.163.3.1 lpc\_order** `MHAParser::int_t lpc_bl_predictor::lpc_order`

**5.163.3.2 name\_kappa** `MHAParser::string_t lpc_bl_predictor::name_kappa`

**5.163.3.3 name\_lpc\_f** `MHAParser::string_t lpc_bl_predictor::name_lpc_f`

**5.163.3.4 name\_lpc\_b** `MHAParser::string_t lpc_bl_predictor::name_lpc_b`

**5.163.3.5 name\_f** `MHAParser::string_t lpc_bl_predictor::name_f`

**5.163.3.6 name\_b** `MHAParser::string_t lpc_bl_predictor::name_b`

**5.163.3.7 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`

## 5.164 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`

### 5.164.1 Constructor & Destructor Documentation

**5.164.1.1 `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 )`

**5.164.1.2 `~lpc_bl_predictor_config()`** `lpc_bl_predictor_config::~lpc_bl_predictor_config ( )`

### 5.164.2 Member Function Documentation

**5.164.2.1 process()** `mha_wave_t * lpc_bl_predictor_config::process (`  
`mha_wave_t * wave )`

### 5.164.3 Member Data Documentation

**5.164.3.1 ac** `algo_comm_t lpc_bl_predictor_config::ac` [private]

**5.164.3.2 f\_est** `MHA_AC::waveform_t lpc_bl_predictor_config::f_est` [private]

**5.164.3.3 b\_est** `MHA_AC::waveform_t lpc_bl_predictor_config::b_est` [private]

**5.164.3.4 forward** `MHASignal::waveform_t lpc_bl_predictor_config::forward` [private]

**5.164.3.5 backward** `MHASignal::waveform_t lpc_bl_predictor_config::backward` [private]

**5.164.3.6 lpc\_order** `int lpc_bl_predictor_config::lpc_order` [private]

**5.164.3.7 name\_km** `std::string lpc_bl_predictor_config::name_km` [private]

**5.164.3.8 name\_f** std::string lpc\_bl\_predictor\_config::name\_f [private]

**5.164.3.9 name\_b** std::string lpc\_bl\_predictor\_config::name\_b [private]

**5.164.3.10 km** mha\_wave\_t lpc\_bl\_predictor\_config::km [private]

**5.164.3.11 s\_f** mha\_wave\_t lpc\_bl\_predictor\_config::s\_f [private]

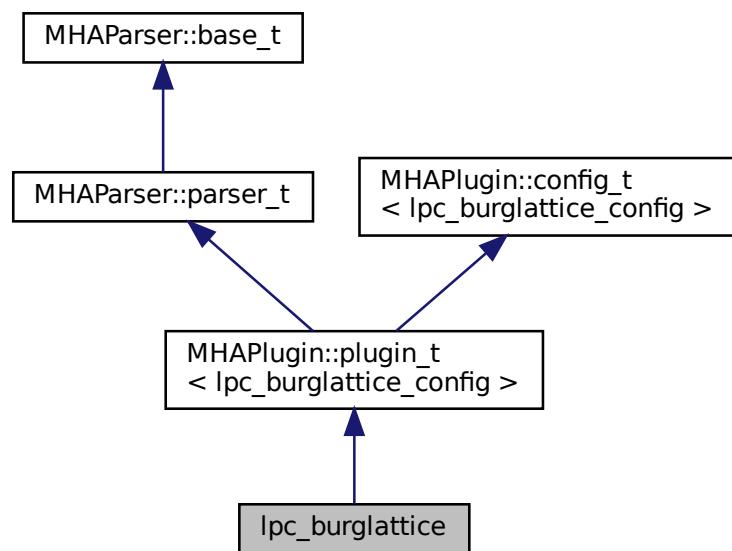
**5.164.3.12 s\_b** mha\_wave\_t lpc\_bl\_predictor\_config::s\_b [private]

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

- **lpc\_bl\_predictor.h**
- **lpc\_bl\_predictor.cpp**

## 5.165 lpc\_burglattice Class Reference

Inheritance diagram for lpc\_burglattice:



## Public Member Functions

- **lpc\_burglattice ( algo\_comm\_t iac, const std::string &configured\_name)**  
*Constructs our plugin.*
- **~lpc\_burglattice ()**
- **mha\_wave\_t \* process ( mha\_wave\_t \*)**  
*Checks for the most recent configuration and defers processing to it.*
- **void prepare ( mhaconfig\_t &)**  
*Plugin preparation.*
- **void release (void)**

## Public Attributes

- **MHAParser::int\_t lpc\_order**
- **MHAParser::string\_t name\_kappa**
- **MHAParser::string\_t name\_f**
- **MHAParser::string\_t name\_b**
- **MHAParser::float\_t lambda**

## Private Member Functions

- **void update\_cfg ()**

## Private Attributes

- **MHAEvents::patchbay\_t< lpc\_burglattice > patchbay**

## Additional Inherited Members

### 5.165.1 Constructor & Destructor Documentation

**5.165.1.1 lpc\_burglattice()** `lpc_burglattice::lpc_burglattice ( algo_comm_t iac, const std::string & configured_name )`

Constructs our plugin.

**5.165.1.2 ~lpc\_burglattice()** `lpc_burglattice::~lpc_burglattice ( )`

## 5.165.2 Member Function Documentation

**5.165.2.1 process()** `mha_wave_t * lpc_burglattice::process ( mha_wave_t * signal )`

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

**5.165.2.2 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::MHAPeripheral ( )` (p. [1149](#)).

**5.165.2.3 release()** `void lpc_burglattice::release ( void ) [inline], [virtual]`

Reimplemented from `MHAPeripheral::MHAPeripheral ( )` (p. [1150](#)).

**5.165.2.4 update\_cfg()** `void lpc_burglattice::update_cfg ( ) [private]`

### 5.165.3 Member Data Documentation

**5.165.3.1 lpc\_order** `MHAParser::int_t lpc_burglattice::lpc_order`

**5.165.3.2 name\_kappa** `MHAParser::string_t lpc_burglattice::name_kappa`

**5.165.3.3 name\_f** `MHAParser::string_t lpc_burglattice::name_f`

**5.165.3.4 name\_b** `MHAParser::string_t lpc_burglattice::name_b`

**5.165.3.5 lambda** `MHAParser::float_t lpc_burglattice::lambda`

**5.165.3.6 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`

## 5.166 **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`

### 5.166.1 Constructor & Destructor Documentation

**5.166.1.1 `lpc_burglattice_config()`** `lpc_burglattice_config::lpc_burglattice_config (`  
`algo_comm_t & iac,`  
`const mhaconfig_t in_cfg,`  
`lpc_burglattice * _lpc )`

**5.166.1.2 `~lpc_burglattice_config()`** `lpc_burglattice_config::~lpc_burglattice_config`  
( )

### 5.166.2 Member Function Documentation

**5.166.2.1 `process()`** `mha_wave_t * lpc_burglattice_config::process (`  
`mha_wave_t * wave )`

### 5.166.3 Member Data Documentation

**5.166.3.1 ac algo\_comm\_t** lpc\_burglattice\_config::ac [private]

**5.166.3.2 forward MHASignal::waveform\_t** lpc\_burglattice\_config::forward [private]

**5.166.3.3 backward MHASignal::waveform\_t** lpc\_burglattice\_config::backward [private]

**5.166.3.4 kappa MHASignal::waveform\_t** lpc\_burglattice\_config::kappa [private]

**5.166.3.5 kappa\_block MHA\_AC::waveform\_t** lpc\_burglattice\_config::kappa\_block [private]

**5.166.3.6 dm MHASignal::waveform\_t** lpc\_burglattice\_config::dm [private]

**5.166.3.7 nm MHASignal::waveform\_t** lpc\_burglattice\_config::nm [private]

**5.166.3.8 lambda mha\_real\_t** lpc\_burglattice\_config::lambda [private]

**5.166.3.9 lpc\_order int** lpc\_burglattice\_config::lpc\_order [private]

---

**5.166.3.10 name\_f** std::string lpc\_burglattice\_config::name\_f [private]

**5.166.3.11 name\_b** std::string lpc\_burglattice\_config::name\_b [private]

**5.166.3.12 s\_f** mha\_wave\_t lpc\_burglattice\_config::s\_f [private]

**5.166.3.13 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**

## 5.167 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**

**5.167.1 Constructor & Destructor Documentation**

**5.167.1.1 `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 )
```

**5.167.1.2 `~lpc_config()`** `lpc_config::~lpc_config ( )`

**5.167.2 Member Function Documentation**

**5.167.2.1 `process()`** `mha_wave_t * lpc_config::process (`

```
    mha_wave_t * wave )
```

**5.167.2.2 `insert()`** `void lpc_config::insert ( )`

**5.167.3 Member Data Documentation**

**5.167.3.1 `norm`** `bool lpc_config::norm [private]`

**5.167.3.2 shift** bool lpc\_config::shift [private]

**5.167.3.3 comp\_each\_iter** unsigned int lpc\_config::comp\_each\_iter [private]

**5.167.3.4 order** unsigned int lpc\_config::order [private]

**5.167.3.5 lpc\_buffer\_size** unsigned int lpc\_config::lpc\_buffer\_size [private]

**5.167.3.6 N** unsigned int lpc\_config::N [private]

**5.167.3.7 comp\_iter** unsigned int lpc\_config::comp\_iter [private]

**5.167.3.8 sample** mha\_wave\_t lpc\_config::sample [private]

**5.167.3.9 R** std::vector< mha\_real\_t > lpc\_config::R [private]

**5.167.3.10 A** std::vector< mha\_real\_t > lpc\_config::A [private]

**5.167.3.11 inwave** `MHASignal::ringbuffer_t lpc_config::inwave` [private]

**5.167.3.12 lpc\_out** `MHA_AC::waveform_t lpc_config::lpc_out` [private]

**5.167.3.13 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`

## 5.168 **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.*

### 5.168.1 Detailed Description

Runtime configuration class of the **Isl2ac** (p. 96) plugin.

## 5.168.2 Constructor & Destructor Documentation

```
5.168.2.1 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.

## 5.168.3 Member Function Documentation

```
5.168.3.1 process() void cfg_t::process ( )
```

## 5.168.4 Member Data Documentation

**5.168.4.1 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.

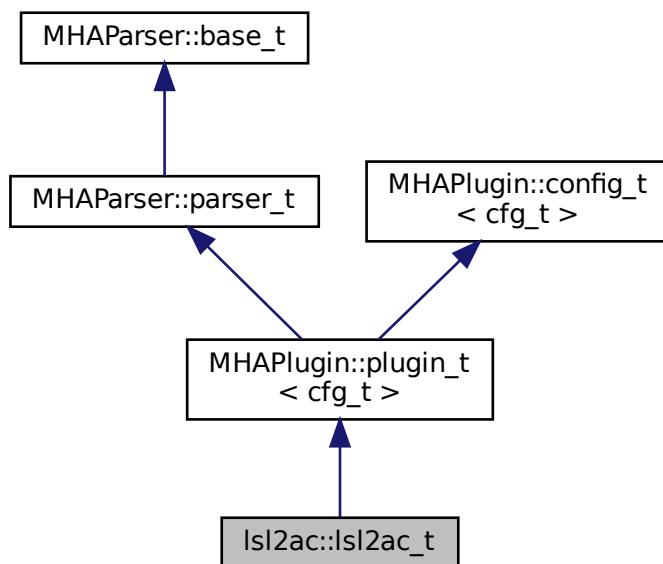
The documentation for this class was generated from the following files:

- **lsl2ac.hh**
- **lsl2ac.cpp**

## 5.169 **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 std::string &configured\_name)**
- **void prepare ( mhaconfig\_t &)**  
*Prepare constructs the vector of bridge variables and locks the configuration, then calls **update()** (p. 680).*
- **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

### 5.169.1 Detailed Description

Plugin class of **Isl2ac** (p. 96).

## 5.169.2 Constructor & Destructor Documentation

**5.169.2.1 `lsl2ac_t()`** `lsl2ac::lsl2ac_t::lsl2ac_t ( algo_comm_t iac, const std::string & configured_name )`

## 5.169.3 Member Function Documentation

**5.169.3.1 `prepare()`** `void lsl2ac::lsl2ac_t::prepare ( mhaconfig_t & ) [virtual]`

Prepare constructs the vector of bridge variables and locks the configuration, then calls **update()** (p. 680).

Implements **MHAPlugin::plugin\_t< cfg\_t >** (p. 1149).

**5.169.3.2 `process() [1/3]`** `mha_wave_t* lsl2ac::lsl2ac_t::process ( mha_wave_t * s ) [inline]`

Processing fct for waveforms.

Calls **process(void)** (p. 679).

**5.169.3.3 `process() [2/3]`** `mha_spec_t* lsl2ac::lsl2ac_t::process ( mha_spec_t * s ) [inline]`

Processing fct for spectra.

Calls **process(void)** (p. 679).

**5.169.3.4 `process() [3/3]`** `void lsl2ac::lsl2ac_t::process ( )`

Process function.

---

**5.169.3.5 `release()`** void lsl2ac::lsl2ac\_t::release ( ) [virtual]

Release fct.

Unlocks variable name list

Reimplemented from **MHAPlugIn::plugin\_t< cfg\_t >** (p. [1150](#)).

**5.169.3.6 `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`.

**5.169.3.7 `update()`** void lsl2ac::lsl2ac\_t::update ( ) [private]

Construct new runtime configuration.

**5.169.3.8 `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
_	

## 5.169.4 Member Data Documentation

**5.169.4.1 `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.

**5.169.4.2 activate** `MHAParser::bool_t lsl2ac::lsl2ac_t::activate = {"Receive from network?", "yes"} [private]`

Config variable for activation/deactivation of plugin.

**5.169.4.3 overrun\_behavior** `MHAParser::kw_t lsl2ac::lsl2ac_t::overrun_behavior = {"How to handle overrun", "discard", "[discard ignore]"} [private]`

Config variable for overrun behavior.

**5.169.4.4 bufsize** `MHAParser::int_t lsl2ac::lsl2ac_t::bufsize [private]`

Config variable for maximum buffer size of LSL.

**5.169.4.5 chunksize** `MHAParser::int_t lsl2ac::lsl2ac_t::chunksize [private]`

Config variable for maximum chunk size of LSL.

**5.169.4.6 nchannels** `MHAParser::int_t lsl2ac::lsl2ac_t::nchannels [private]`

Config variable for number of channels to expect from LSL stream.

**5.169.4.7 nsamples** `MHAParser::int_t lsl2ac::lsl2ac_t::nsamples [private]`

Config variable for maximum chunk size of LSL.

**5.169.4.8 patchbay** `MHAEEvents::patchbay_t< Isl2ac_t> Isl2ac::Isl2ac_t::patchbay`  
[private]

Patchbay for configuration callbacks.

**5.169.4.9 available\_streams** `MHAParser::vstring_mon_t Isl2ac::Isl2ac_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:

- `Isl2ac.hh`
- `Isl2ac.cpp`

## 5.170 Isl2ac::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.*

### 5.170.1 Detailed Description

LSL to AC bridge variable.

## 5.170.2 Constructor & Destructor Documentation

**5.170.2.1 `save_var_t()` [1/3]** `lsl2ac::save_var_t::save_var_t (`  
`const save_var_t & ) [delete]`

**5.170.2.2 `save_var_t()` [2/3]** `lsl2ac::save_var_t::save_var_t (`  
`save_var_t && ) [delete]`

**5.170.2.3 `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

**5.170.2.4 ~save\_var\_t()** lsl2ac::save\_var\_t::~save\_var\_t ( ) [default]

### 5.170.3 Member Function Documentation

**5.170.3.1 operator=() [1/2]** save\_var\_t& lsl2ac::save\_var\_t::operator= (const save\_var\_t &) [delete]

**5.170.3.2 operator=() [2/2]** save\_var\_t& lsl2ac::save\_var\_t::operator= (save\_var\_t &&) [delete]

**5.170.3.3 info()** lsl::stream\_info lsl2ac::save\_var\_t::info ( )

Get stream info object from stream inlet.

**5.170.3.4 receive\_frame()** void lsl2ac::save\_var\_t::receive\_frame ( )

Receive a samples from lsl and copy to AC space.

Handling of underrun is configuration-dependent

**5.170.3.5 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.

**5.170.3.6 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.

**5.170.3.7 get\_time\_correction()** void lsl2ac::save\_var\_t::get\_time\_correction ( )  
[private]

Refresh time correction value every 5s.

**5.170.3.8 insert\_vars()** void lsl2ac::save\_var\_t::insert\_vars ( ) [private]

Insert stream value, time stamp and time offset into ac space.

## 5.170.4 Member Data Documentation

**5.170.4.1 stream** lsl::stream\_inlet lsl2ac::save\_var\_t::stream [private]

LSL stream outlet.

Interface to Isl

**5.170.4.2 buf** std::vector< mha\_real\_t> lsl2ac::save\_var\_t::buf [private]

Data buffer of the ac variable.

**5.170.4.3 ts\_buf** std::vector<double> lsl2ac::save\_var\_t::ts\_buf [private]

Timestamp buffer.

**5.170.4.4 ac** const algo\_comm\_t& lsl2ac::save\_var\_t::ac [private]

Handle to AC space.

**5.170.4.5 cv** comm\_var\_t lsl2ac::save\_var\_t::cv [private]

Timeseries AC variable.

**5.170.4.6 ts** comm\_var\_t lsl2ac::save\_var\_t::ts [private]

Timestamp AC variable.

**5.170.4.7 tc** double lsl2ac::save\_var\_t::tc =0.0 [private]

Current time correction.

**5.170.4.8 ts\_name** std::string lsl2ac::save\_var\_t::ts\_name [private]

Timestamp AC variable name.

**5.170.4.9 tc\_name** std::string lsl2ac::save\_var\_t::tc\_name [private]

Time correction AC variable name.

**5.170.4.10 new\_name** std::string lsl2ac::save\_var\_t::new\_name [private]

Number of new samples AC variable name.

**5.170.4.11 tic** std::chrono::time\_point<std::chrono::steady\_clock> lsl2ac::save\_↔  
var\_t::tic [private]

time point of last time correction pull

**5.170.4.12 skip** bool lsl2ac::save\_var\_t::skip =false [private]

Should the variable be skipped in future process calls? Only true when error occurred.

**5.170.4.13 ob overrun\_behavior** lsl2ac::save\_var\_t::ob [private]

Behavior on stream overrun.

**5.170.4.14 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

**5.170.4.15 nchannels** int32\_t lsl2ac::save\_var\_t::nchannels [private]

Number of channels.

**5.170.4.16 nsamples** std::size\_t lsl2ac::save\_var\_t::nsamples [private]

Number of samples per channel in the AC variable.

Zero means resize as needed.

**5.170.4.17 chunksize** std::size\_t lsl2ac::save\_var\_t::chunksize [private]

Maximal chunk size of lsl stream.

**5.170.4.18 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.

**5.170.4.19 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**

## 5.171 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 $\leftarrow$  ::mfloat\_t \* **var**  
*Ptr to parser.*
- matlab\_wrapper\_t \* **parent**  
*Ptr to parent plugin.*

### 5.171.1 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.

### 5.171.2 Constructor & Destructor Documentation

#### 5.171.2.1 `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_

### 5.171.3 Member Function Documentation

#### 5.171.3.1 `on_writeaccess()`

Write access callback.

To be called on every writeaccess of \*var. Synchronises \*var and \*user\_config.

### 5.171.4 Member Data Documentation

## **5.172 matlab\_wrapper::matlab\_wrapper\_rt\_cfg\_t Class Reference**

### **5.171.4.1 user\_config user\_config\_t\* matlab\_wrapper::callback::user\_config [private]**

Ptr to user\_config\_t.

### **5.171.4.2 var MHPARSER::mfloat\_t\* matlab\_wrapper::callback::var [private]**

Ptr to parser.

### **5.171.4.3 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**

## **5.172 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.*

### 5.172.1 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.

### 5.172.2 Constructor & Destructor Documentation

**5.172.2.1 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
---------------------	---

**5.172.2.2 ~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

### 5.172.3 Member Data Documentation

**5.172.3.1 user\_config** emxArray\_user\_config\_t\* matlab\_wrapper::matlab\_wrapper\_rt\_cfg\_t::user\_config

User configuration to be used in the process callback.

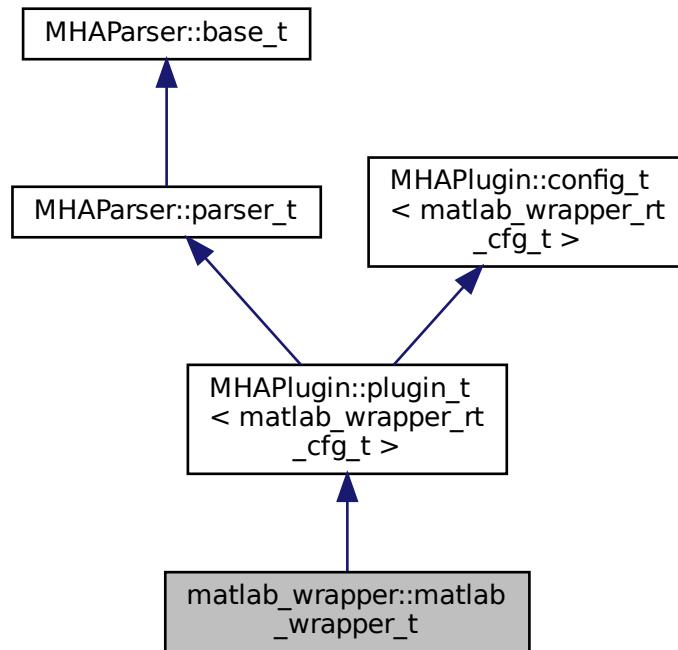
The documentation for this class was generated from the following files:

- **matlab\_wrapper.hh**
- **matlab\_wrapper.cpp**

## 5.173 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 &configured\_name)  
*Ctor of the plugin interface class.*
- **void process ( mha\_wave\_t \*sin, mha\_wave\_t \*\*sout)**  
*Pure waveform processing.*
- **void process ( mha\_spec\_t \*sin, mha\_spec\_t \*\*sout)**  
*Pure spectrum processing.*
- **void process ( mha\_wave\_t \*sin, mha\_spec\_t \*\*sout)**  
*Signal processing with domain transformation from waveform to spectrum.*
- **void process ( mha\_spec\_t \*sin, mha\_wave\_t \*\*sout)**  
*Signal processing with domain transformation from spectrum to waveform.*
- **void prepare ( mhaconfig\_t &signal\_info)**  
*Prepare callback.*
- **void release ()**  
*Release callback.*

## Private Member Functions

- **void insert\_monitors ()**
- **void update\_monitors ()**
- **void insert\_config\_vars ()**
- **void load\_lib ()**  
*Create new library wrapper and load library.*
- **void update ()**  
*Create new real time safe user config from user config.*

## Private Attributes

- **friend callback**
- **MHAParser::string\_t library\_name {"Name of matlab generated library",""}**  
*Configuration variable holding the file name of the matlab generated library.*
- **MHAEvents::patchbay\_t< matlab\_wrapper\_t > patchbay**  
*Patchbay for the interface plugins.*
- **MHAEvents::patchbay\_t< callback > cb\_patchbay**  
*Patchbay for the custom callbacks.*
- **std::unique\_ptr< wrapped\_plugin\_t > plug**  
*Unique ptr holding the instance of the plugin wrapper class.*
- **std::deque< MHAParser::mfloat\_t > vars**  
*Deque holding the user defined configuration variables.*
- **std::deque< callback > callbacks**  
*Deque holding the callbacks for the user defined variables' write access.*
- **std::deque< MHAParser::mfloat\_mon\_t > monitors**  
*Deque holding the monitors corresponding to user state.*

## Additional Inherited Members

### 5.173.1 Detailed Description

Matlab wraper plugin interface class.

### 5.173.2 Constructor & Destructor Documentation

```
5.173.2.1 matlab_wrapper_t() matlab_wrapper::matlab_wrapper_t::matlab_wrapper_t (
    const algo_comm_t iac,
    const std::string & configured_name )
```

Ctor of the plugin interface class.

#### Parameters

<i>iac</i>	Handle to ac space
<i>configured_name</i>	Configured name of this plugin

### 5.173.3 Member Function Documentation

```
5.173.3.1 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

---

**5.173.3.2 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

**5.173.3.3 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

**5.173.3.4 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

**5.173.3.5 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. [1149](#)).

### **5.173.3.6 `release()`** void matlab\_wrapper::matlab\_wrapper\_t::release ( ) [virtual]

Release callback.

Reimplemented from `MHAPlugIn::plugin_t< matlab_wrapper_rt_cfg_t >` (p. [1150](#)).

### **5.173.3.7 `insert_monitors()`** void matlab\_wrapper::matlab\_wrapper\_t::insert\_monitors ( ) [private]

### **5.173.3.8 `update_monitors()`** void matlab\_wrapper::matlab\_wrapper\_t::update\_monitors ( ) [private]

### **5.173.3.9 `insert_config_vars()`** void matlab\_wrapper::matlab\_wrapper\_t::insert\_config\_vars ( ) [private]

### **5.173.3.10 `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`.

### **5.173.3.11 `update()`** void matlab\_wrapper::matlab\_wrapper\_t::update ( ) [private]

Create new real time safe user config from user config.

Called by the custom callbacks.

## 5.173.4 Member Data Documentation

**5.173.4.1 callback** friend matlab\_wrapper::matlab\_wrapper\_t::callback [private]

**5.173.4.2 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.

**5.173.4.3 patchbay** `MHAEEvents::patchbay_t< matlab_wrapper_t> matlab_wrapper::matlab_wrapper_t::patchbay [private]`

Patchbay for the interface plugins.

**5.173.4.4 cb\_patchbay** `MHAEEvents::patchbay_t< callback> matlab_wrapper::matlab_wrapper_t::cb_patchbay [private]`

Patchbay for the custom callbacks.

Can use normal patchbay bc/ of the interface of patchbay\_t

**5.173.4.5 plug** `std::unique_ptr< wrapped_plugin_t> matlab_wrapper::matlab_wrapper_t::plug [private]`

Unique ptr holding the instance of the plugin wrapper class.

**5.173.4.6 vars** `std::deque< MHAParser::mfloat_t> matlab_wrapper::matlab_wrapper_t::vars [private]`

Deque holding the user defined configuration variables.

Deque is used because we need an indexable container that does not invalidate pointers

**5.173.4.7 callbacks** std::deque< `callback`> matlab\_wrapper::matlab\_wrapper\_t::callbacks  
[private]

Deque holding the callbacks for the user defined variables' write access.

Deque is used because we need an indexable container that does not invalidate pointers

**5.173.4.8 monitors** std::deque< `MHAParser::mfloate_mon_t`> matlab\_wrapper::matlab\_wrapper\_t::monitors [private]

Deque holding the monitors corresponding to user state.

Deque is used because we need an indexable container that does not invalidate pointers.

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

- `matlab_wrapper.hh`
- `matlab_wrapper.cpp`

## **5.174 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.*

### **5.174.1 Detailed Description**

Wrapper class around the matlab-generated library.

### **5.174.2 Constructor & Destructor Documentation**

**5.174.2.1 wrapped\_plugin\_t() [1/2]** `matlab_wrapper::matlab_wrapper_t::wrapped_plugin_t::wrapped_plugin_t( ) [delete]`

**5.174.2.2 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
_	

**5.174.2.3 ~wrapped\_plugin\_t()** `matlab_wrapper::matlab_wrapper_t::wrapped_plugin_t::~wrapped_plugin_t( ) [virtual]`

Dtor.

### **5.174.3 Member Function Documentation**

---

**5.174.3.1 process\_ww()** `mha_wave_t * matlab_wrapper::matlab_wrapper_t::wrapped_`  $\leftarrow$   
`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

<code>s</code>	input/output signal
<code>user_</code> $\leftarrow$ <code>config_</code>	Ptr to user configuration array

**5.174.3.2 process\_ss()** `mha_spec_t * matlab_wrapper::matlab_wrapper_t::wrapped_`  $\leftarrow$   
`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

<code>s</code>	input/output signal
<code>user_</code> $\leftarrow$ <code>config_</code>	Ptr to user configuration array

**5.174.3.3 process\_ws()** `mha_spec_t * matlab_wrapper::matlab_wrapper_t::wrapped_`  $\leftarrow$   
`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_config_</i>	Ptr to user configuration array

```
5.174.3.4 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_config_</i>	Ptr to user configuration array

```
5.174.3.5 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

```
5.174.3.6 release() void matlab_wrapper::matlab_wrapper_t::wrapped_plugin_t::release
( )
```

Release callback.

Cleans up io arrays and calls wrapped release if necessary.

## 5.174.4 Member Data Documentation

**5.174.4.1 user\_config** emxArray\_user\_config\_t\* matlab\_wrapper::matlab\_wrapper\_t::wrapped\_plugin\_t::user\_config =nullptr

Ptr to user config array.

**5.174.4.2 state** emxArray\_user\_config\_t\* matlab\_wrapper::matlab\_wrapper\_t::wrapped\_plugin\_t::state =nullptr

Ptr to state array.

**5.174.4.3 library\_handle** dynamiclib\_t matlab\_wrapper::matlab\_wrapper\_t::wrapped\_plugin\_t::library\_handle [private]

Handle to matlab generated shared library.

**5.174.4.4 fcn\_terminate** void(\* matlab\_wrapper::matlab\_wrapper\_t::wrapped\_plugin\_t::fcn\_terminate) () =nullptr [private]

Handle to matlab generated cleanup function.

**5.174.4.5 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.

**5.174.4.6 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.

**5.174.4.7 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.

**5.174.4.8 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.

**5.174.4.9 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.

**5.174.4.10 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.

**5.174.4.11 fcn\_release** void(\* matlab\_wrapper::matlab\_wrapper\_t::wrapped\_plugin\_t::fcn\_release) () =nullptr [private]

**5.174.4.12 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

**5.174.4.13 wave\_in** emxArray\_real\_T\* matlab\_wrapper::matlab\_wrapper\_t::wrapped\_plugin\_t::wave\_in =nullptr [private]

Ptr to emxArray holding the input signal.

**5.174.4.14 wave\_out** emxArray\_real\_T\* matlab\_wrapper::matlab\_wrapper\_t::wrapped\_plugin\_t::wave\_out =nullptr [private]

Ptr to emxArray holding the output signal.

**5.174.4.15 spec\_in** emxArray\_creal\_T\* matlab\_wrapper::matlab\_wrapper\_t::wrapped\_plugin\_t::spec\_in =nullptr [private]

Ptr to emxArray holding the input signal.

**5.174.4.16 spec\_out** emxArray\_creal\_T\* matlab\_wrapper::matlab\_wrapper\_t::wrapped\_plugin\_t::spec\_out =nullptr [private]

Ptr to emxArray holding the output signal.

**5.174.4.17 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.

**5.174.4.18 mha\_spec\_out** std::unique\_ptr< **MHASignal::spectrum\_t::matlab\_wrapper\_t::wrapped\_plugin\_t::mha\_spec\_out [private]**

MHA waveform holding the output signal.

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

- **matlab\_wrapper.hh**
- **matlab\_wrapper.cpp**

## **5.175 matlab\_wrapper::types< T > Struct Template Reference**

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

- **matlab\_wrapper.hh**

## **5.176 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**

### **5.176.1 Member Typedef Documentation**

**5.176.1.1 array\_type** **typedef emxArray\_creal\_T matlab\_wrapper::types< MHA\_SPECTRUM >::array\_type**

**5.176.1.2 signal\_type** `typedef mha_spec_t matlab_wrapper::types< MHA_SPECTRUM >:: signal_type`

**5.176.1.3 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`

## 5.177 `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`

### 5.177.1 Member Typedef Documentation

**5.177.1.1 array** `typedef emxArray_real_T matlab_wrapper::types< MHA_WAVEFORM >:: array`

**5.177.1.2 signal\_type** `typedef mha_wave_t matlab_wrapper::types< MHA_WAVEFORM >:: signal_type`

**5.177.1.3 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`

## 5.178 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**

#### 5.178.1 Constructor & Destructor Documentation

```
5.178.1.1 cfg_t() cfg_t::cfg_t (
    std::vector< std::vector< float > > imixer,
    unsigned int ci,
    unsigned int co,
    unsigned int fragsize,
    unsigned int nfft )
```

#### 5.178.2 Member Function Documentation

```
5.178.2.1 process() [1/2] mha_wave_t * cfg_t::process (
    mha_wave_t * s )
```

```
5.178.2.2 process() [2/2] mha_spec_t * cfg_t::process (
    mha_spec_t * s )
```

### 5.178.3 Member Data Documentation

**5.178.3.1 m** `MHASignal::waveform_t` `matrixmixer::cfg_t::m` [private]

**5.178.3.2 wout** `MHASignal::waveform_t` `matrixmixer::cfg_t::wout` [private]

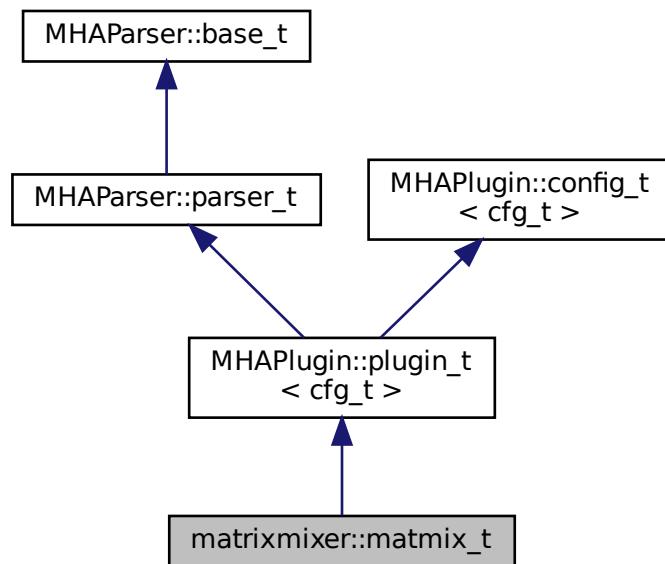
**5.178.3.3 sout** `MHASignal::spectrum_t` `matrixmixer::cfg_t::sout` [private]

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

- `matrixmixer.cpp`

### 5.179 matrixmixer::matmix\_t Class Reference

Inheritance diagram for matrixmixer::matmix\_t:



## Public Member Functions

- `matmix_t ( algo_comm_t iac, const std::string &configured_name)`
- `void prepare ( mhaconfig_t &)`
- `mha_wave_t * process ( mha_wave_t *)`
- `mha_spec_t * process ( mha_spec_t *)`

## Private Member Functions

- `void update_m ()`

## Private Attributes

- `MHAEvents::patchbay_t< matmix_t > patchbay`
- `MHAParser::mfloat_t mixer`
- `unsigned int ci`
- `unsigned int co`

## Additional Inherited Members

### 5.179.1 Constructor & Destructor Documentation

```
5.179.1.1 matmix_t() matrixmixer::matmix_t::matmix_t (
    algo_comm_t iac,
    const std::string & configured_name )
```

### 5.179.2 Member Function Documentation

```
5.179.2.1 prepare() void matrixmixer::matmix_t::prepare (
    mhaconfig_t & tf ) [virtual]
```

Implements `MHAPlugin::plugin_t< cfg_t >` (p. 1149).

**5.179.2.2 process()** [1/2] `mha_wave_t * matrixmixer::matmix_t::process ( mha_wave_t * s )`

**5.179.2.3 process()** [2/2] `mha_spec_t * matrixmixer::matmix_t::process ( mha_spec_t * s )`

**5.179.2.4 update\_m()** `void matrixmixer::matmix_t::update_m ( ) [private]`

### 5.179.3 Member Data Documentation

**5.179.3.1 patchbay** `MHAEvents::patchbay_t< matmix_t> matrixmixer::matmix_t::patchbay [private]`

**5.179.3.2 mixer** `MHAParser::mfloat_t matrixmixer::matmix_t::mixer [private]`

**5.179.3.3 ci** `unsigned int matrixmixer::matmix_t::ci [private]`

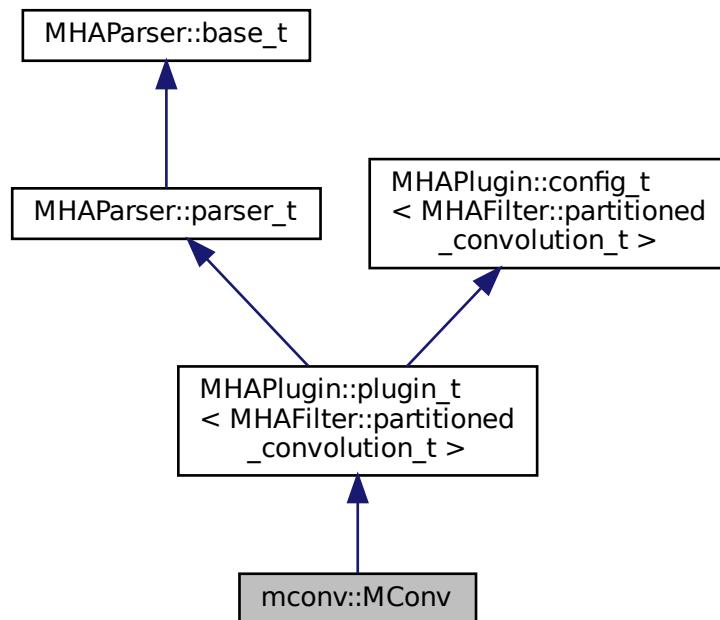
**5.179.3.4 co** `unsigned int matrixmixer::matmix_t::co [private]`

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

- **matrixmixer.cpp**

## 5.180 mconv::MConv Class Reference

Inheritance diagram for mconv::MConv:



### Public Member Functions

- **MConv** (const **algo\_comm\_t** &iac, const std::string &algoname)  
*Plugin constructor.*
- void **prepare** ( **mhaconfig\_t** &mhaconfig)  
*Prepare this plugin for processing.*
- void **release** ()
- **mha\_wave\_t** \* **process** ( **mha\_wave\_t** \*)

### Private Member Functions

- void **update** ()  
*Update is needed only once, since this plugin allows only change of irs after **prepare()** (p. 715).*
- void **update\_irs** ()  
*This function updates the irs without allowing a change of its size after **prepare()** (p. 715).*

## Private Attributes

- **MHAParser::int\_t nchannels\_out**  
*Number of output channels to produce.*
- **MHAParser::vint\_t inch**  
*Vector of input channel indices.*
- **MHAParser::vint\_t outch**  
*Vector of output channel indices.*
- **MHAParser::mfloat\_t irs**  
*Impulse responses, one per row.*
- **unsigned int nchannels\_in**  
*Number of input channels, set during prepare.*
- **unsigned int fragsize**  
*Fragsize, set during prepare, is used as the partition length in the partitioned convolution.*
- **MHAEvents::patchbay\_t< MConv > patchbay**

## Additional Inherited Members

### 5.180.1 Detailed Description

class implements plugin for partitioned convolution. A matrix of impulse responses, filtering n input channels to m output channels, is supported.

### 5.180.2 Constructor & Destructor Documentation

```
5.180.2.1 MConv() mconv::MConv::MConv (
    const algo_comm_t & iac,
    const std::string & algoname )
```

Plugin constructor.

#### Parameters

<i>iac</i>	handle and function pointers for algorithm communication
<i>configured_name</i>	The name by which e.g an mhachain refers to this instance of the mconv plugin

### 5.180.3 Member Function Documentation

**5.180.3.1 `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. [1149](#)).

**5.180.3.2 `release()`** `void mconv::MConv::release ( ) [virtual]`

Reimplemented from `MHAPlugin::plugin_t< MHAFilter::partitioned_convolution_t >` (p. [1150](#)).

**5.180.3.3 `process()`** `mha_wave_t * mconv::MConv::process ( mha_wave_t * s_in )`

**5.180.3.4 `update()`** `void mconv::MConv::update ( ) [private]`

Update is needed only once, since this plugin allows only change of irs after `prepare()` (p. [715](#)).

**5.180.3.5 `update_ir()`** `void mconv::MConv::update_ir ( ) [private]`

This function updates the irs without allowing a change of its size after `prepare()` (p. [715](#)).

## 5.180.4 Member Data Documentation

### 5.180.4.1 **nchannels\_out** `MHAParser::int_t mconv::MConv::nchannels_out [private]`

Number of output channels to produce.

### 5.180.4.2 **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.

### 5.180.4.3 **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.

### 5.180.4.4 **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.

### 5.180.4.5 **nchannels\_in** `unsigned int mconv::MConv::nchannels_in [private]`

Number of input channels, set during prepare.

### 5.180.4.6 **fragsize** `unsigned int mconv::MConv::fragsize [private]`

Fragsize, set during prepare, is used as the partition length in the partitioned convolution.

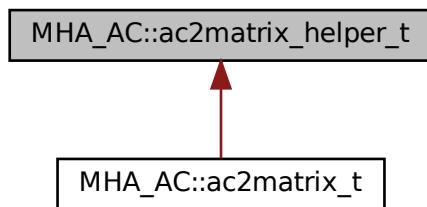
### 5.180.4.7 patchbay MHAEvents::patchbay\_t < MConv> mconv::MConv::patchbay [private]

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

- `mconv.cpp`

## 5.181 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`

## 5.181.1 Constructor & Destructor Documentation

**5.181.1.1 ac2matrix\_helper\_t()** MHA\_AC::ac2matrix\_helper\_t::ac2matrix\_helper\_t ( algo\_comm\_t iac, const std::string & iname )

## 5.181.2 Member Function Documentation

**5.181.2.1 getvar()** void MHA\_AC::ac2matrix\_helper\_t::getvar ( )

## 5.181.3 Member Data Documentation

**5.181.3.1 ac** algo\_comm\_t MHA\_AC::ac2matrix\_helper\_t::ac

**5.181.3.2 name** std::string MHA\_AC::ac2matrix\_helper\_t::name

**5.181.3.3 username** std::string MHA\_AC::ac2matrix\_helper\_t::username

**5.181.3.4 size** MHASignal::uint\_vector\_t MHA\_AC::ac2matrix\_helper\_t::size

### 5.181.3.5 **is\_complex** `bool MHA_AC::ac2matrix_helper_t::is_complex`

### 5.181.3.6 **acvar** `comm_var_t MHA_AC::ac2matrix_helper_t::acvar [protected]`

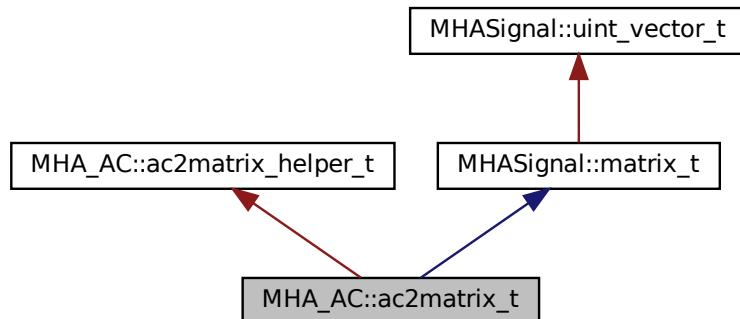
The documentation for this class was generated from the following files:

- `mha_algo_comm.h`
- `mha_algo_comm.cpp`

## 5.182 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

### 5.182.1 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. 720) function is real-time safe.

### 5.182.2 Constructor & Destructor Documentation

```
5.182.2.1 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

### 5.182.3 Member Function Documentation

```
5.182.3.1 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.

```
5.182.3.2 getname() const std::string& MHA_AC::ac2matrix_t::getname ( ) const [inline]
```

Return name of AC variable/matrix.

**5.182.3.3 getusername()** const std::string& MHA\_AC::ac2matrix\_t::getusername ( )  
const [inline]

Return user specified name of AC variable/matrix.

**5.182.3.4 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](#)

## 5.183 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**

### 5.183.1 Detailed Description

Copy all or a subset of all numeric AC variables into an array of matrixes.

### 5.183.2 Constructor & Destructor Documentation

```
5.183.2.1 acspace2matrix_t() [1/2] MHA_AC::acspace2matrix_t::acspace2matrix_t (
    algo_comm_t ac,
    const std::vector< std::string > & names )
```

Constructor.

Scan all given AC variables and allocate corresponding matrixes.

#### Parameters

<b>ac</b>	AC handle.
<b>names</b>	Names of AC variables, or empty for all.

**5.183.2.2 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.
------------------	------------------------

**5.183.2.3 ~acspace2matrix\_t()** `MHA_AC::acspace2matrix_t::~acspace2matrix_t ( )`

### 5.183.3 Member Function Documentation

**5.183.3.1 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.
------------------	---------------------------------

**5.183.3.2 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 <b>size()</b> (p. <a href="#">724</a> )-1.
----------------	--

### Return values

<i>Reference</i>	to matrix.
------------------	------------

**5.183.3.3 operator[]( ) [2/2]** const **MHA\_AC::ac2matrix\_t&** MHA\_AC::acspace2matrix\_t $\leftarrow$   
**::operator[] (**  
  **unsigned int k ) const** [inline]

Constant access operator.

### Parameters

<i>k</i>	index into array; should not exceed <b>size()</b> (p. 724)-1.
----------	---

### Return values

<i>Constant</i>	reference to matrix.
-----------------	----------------------

**5.183.3.4 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.

**5.183.3.5 size()** unsigned int MHA\_AC::acspace2matrix\_t::size ( ) const [inline]

Number of matrixes in AC space.

**5.183.3.6 frame()** unsigned int MHA\_AC::acspace2matrix\_t::frame ( ) const [inline]

Actual frame number.

**5.183.3.7 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
-----------------	--------------------------------

**5.183.4 Member Data Documentation****5.183.4.1 len** `unsigned int MHA_AC::acspace2matrix_t::len [private]`**5.183.4.2 data** `MHA_AC::ac2matrix_t** MHA_AC::acspace2matrix_t::data [private]`**5.183.4.3 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`

**5.184 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`

### 5.184.1 Detailed Description

Insert a double precision floating point variable into the AC space.

The variable is automatically removed on destruction.

### 5.184.2 Constructor & Destructor Documentation

**5.184.2.1 `double_t()`** `MHA_AC::double_t::double_t (`  
`algo_comm_t ac,`  
`std::string name,`  
`double val = 0 )`

**5.184.2.2 `~double_t()`** `MHA_AC::double_t::~double_t ( )`

### 5.184.3 Member Function Documentation

**5.184.3.1 `insert()`** `void MHA_AC::double_t::insert ( )`

### 5.184.4 Member Data Documentation

**5.184.4.1 data** double MHA\_AC::double\_t::data

Floating point (double precision) value variable.

**5.184.4.2 ac** algo\_comm\_t MHA\_AC::double\_t::ac [private]**5.184.4.3 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](#)

## 5.185 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**

### 5.185.1 Detailed Description

Insert a float point variable into the AC space.

The variable is automatically removed on destruction.

### 5.185.2 Constructor & Destructor Documentation

```
5.185.2.1 float_t() MHA_AC::float_t::float_t (
    algo_comm_t ac,
    std::string name,
    float val = 0 )
```

Constructor.

```
5.185.2.2 ~float_t() MHA_AC::float_t::~float_t ( )
```

### 5.185.3 Member Function Documentation

```
5.185.3.1 insert() void MHA_AC::float_t::insert ( )
```

### 5.185.4 Member Data Documentation

```
5.185.4.1 data float MHA_AC::float_t::data
```

Floating point value variable.

**5.185.4.2 ac algo\_comm\_t MHA\_AC::float\_t::ac [private]**

**5.185.4.3 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**

## 5.186 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**

## 5.186.1 Detailed Description

Insert a integer variable into the AC space.

The variable is automatically removed on destruction.

## 5.186.2 Constructor & Destructor Documentation

**5.186.2.1 `int_t()`** `MHA_AC::int_t::int_t ( algo_comm_t ac, std::string name, int val = 0 )`

**5.186.2.2 `~int_t()`** `MHA_AC::int_t::~int_t ( )`

## 5.186.3 Member Function Documentation

**5.186.3.1 `insert()`** `void MHA_AC::int_t::insert ( )`

## 5.186.4 Member Data Documentation

**5.186.4.1 `data`** `int MHA_AC::int_t::data`

Integer value variable.

**5.186.4.2 `ac`** `algo_comm_t MHA_AC::int_t::ac [private]`

### 5.186.4.3 name std::string MHA\_AC::int\_t::name [private]

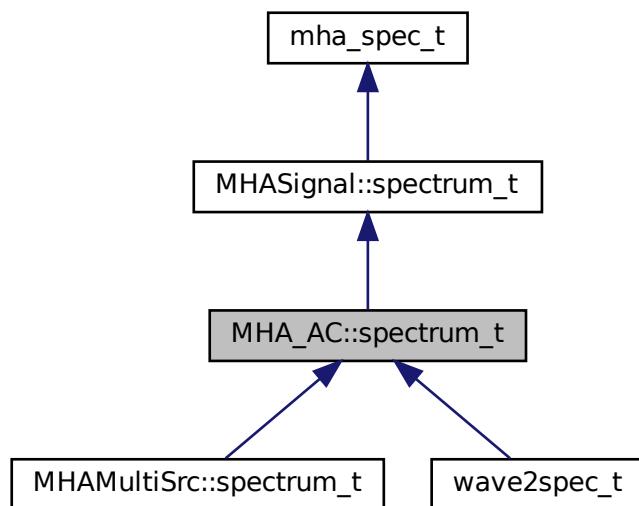
The documentation for this class was generated from the following files:

- [mha\\_algo\\_comm.h](#)
- [mha\\_algo\\_comm.cpp](#)

## 5.187 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

### 5.187.1 Detailed Description

Insert a `MHASignal::spectrum_t` (p. 1244) class into the AC space.

The variable is automatically removed on destruction.

### 5.187.2 Constructor & Destructor Documentation

```
5.187.2.1 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. 735) class
<code>channels</code>	Number of audio channels in the <code>waveform_t</code> (p. 735) class
<code>insert_now</code>	Insert implicitly in the constructor (true) or explicitly in the <code>insert()</code> (p. 733) function (false)

**5.187.2.2 ~spectrum\_t()** MHA\_AC::spectrum\_t::~spectrum\_t ( ) [virtual]

Reimplemented from `MHASignal::spectrum_t` (p. 1246).

### 5.187.3 Member Function Documentation

#### 5.187.3.1 **insert()** void MHA\_AC::spectrum\_t::insert ( )

Insert AC variable into AC space.

### 5.187.4 Member Data Documentation

#### 5.187.4.1 **ac algo\_comm\_t** MHA\_AC::spectrum\_t::ac [protected]

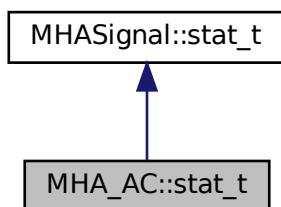
#### 5.187.4.2 **name std::string** MHA\_AC::spectrum\_t::name [protected]

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

- **mha\_algo\_comm.h**
- **mha\_algo\_comm.cpp**

## 5.188 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**

### 5.188.1 Constructor & Destructor Documentation

```
5.188.1.1 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 )
```

### 5.188.2 Member Function Documentation

```
5.188.2.1 update() void MHA_AC::stat_t::update ( )
```

```
5.188.2.2 insert() void MHA_AC::stat_t::insert ( )
```

### 5.188.3 Member Data Documentation

### 5.188.3.1 mean MHA\_AC::waveform\_t MHA\_AC::stat\_t::mean [private]

### 5.188.3.2 std MHA\_AC::waveform\_t MHA\_AC::stat\_t::std [private]

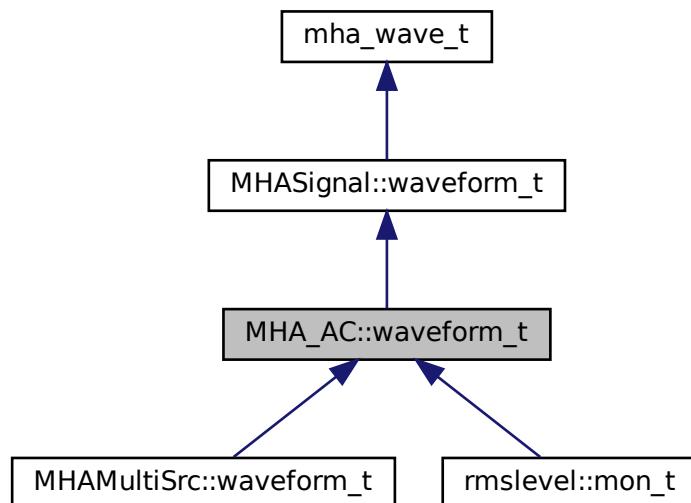
The documentation for this class was generated from the following files:

- `mha_algo_comm.h`
- `mha_algo_comm.cpp`

## 5.189 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

### 5.189.1 Detailed Description

Insert a `MHASignal::waveform_t` (p. 1259) class into the AC space.

The variable is automatically removed on destruction.

### 5.189.2 Constructor & Destructor Documentation

```
5.189.2.1 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. 735) class
<code>channels</code>	Number of audio channels in the <code>waveform_t</code> (p. 735) class
<code>insert_now</code>	Insert implicitly in the constructor (true) or explicitly in the <code>insert()</code> (p. 737) function (false)

**5.189.2.2 ~waveform\_t()** MHA\_AC::waveform\_t::~waveform\_t ( ) [virtual]

Reimplemented from `MHASignal::waveform_t` (p. 1262).

### 5.189.3 Member Function Documentation

#### 5.189.3.1 **insert()** void MHA\_AC::waveform\_t::insert ( )

Insert AC variable into AC space.

### 5.189.4 Member Data Documentation

#### 5.189.4.1 **ac** algo\_comm\_t MHA\_AC::waveform\_t::ac [protected]

#### 5.189.4.2 **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**

## 5.190 mha\_audio\_descriptor\_t Struct Reference

Description of an audio fragment (planned as a replacement of **mhaconfig\_t** (p. 847)).

### Public Attributes

- unsigned int **n\_samples**  
*Number of samples.*
- unsigned int **n\_channels**  
*Number of audio channels.*
- unsigned int **n\_freqs**  
*Number of frequency bands.*
- unsigned int **is\_complex**  
*Flag about sample type.*
- **mha\_real\_t dt**  
*Time distance between samples (only equidistant samples allowed)*
- **mha\_real\_t \* cf**  
*Center frequencies of frequency bands.*
- **mha\_real\_t \* chdir**  
*Hint on source direction of channel, values below zero is left, values above zero is right, zero means unknown.*

### 5.190.1 Detailed Description

Description of an audio fragment (planned as a replacement of **mhaconfig\_t** (p. 847)).

### 5.190.2 Member Data Documentation

#### 5.190.2.1 **n\_samples** `unsigned int mha_audio_descriptor_t::n_samples`

Number of samples.

#### 5.190.2.2 **n\_channels** `unsigned int mha_audio_descriptor_t::n_channels`

Number of audio channels.

#### 5.190.2.3 **n\_freqs** `unsigned int mha_audio_descriptor_t::n_freqs`

Number of frequency bands.

#### 5.190.2.4 **is\_complex** `unsigned int mha_audio_descriptor_t::is_complex`

Flag about sample type.

#### 5.190.2.5 **dt** `mha_real_t mha_audio_descriptor_t::dt`

Time distance between samples (only equidistant samples allowed)

**5.190.2.6 cf mha\_real\_t\* mha\_audio\_descriptor\_t::cf**

Center frequencies of frequency bands.

**5.190.2.7 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**

## 5.191 mha\_audio\_t Struct Reference

An audio fragment in the openMHA (planned as a replacement of **mha\_wave\_t** (p. 836) and **mha\_spec\_t** (p. 790)).

### 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. 738) is unset.*
- **mha\_complex\_t \* cdata**  
*Data pointer if flag **mha\_audio\_descriptor\_t::is\_complex** (p. 738) is set.*

### 5.191.1 Detailed Description

An audio fragment in the openMHA (planned as a replacement of **mha\_wave\_t** (p. 836) and **mha\_spec\_t** (p. 790)).

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. 836) and **mha\_spec\_t** (p. 790) data pointers into corresponding **mha\_audio\_t** (p. 739) objects.

### 5.191.2 Member Data Documentation

**5.191.2.1 descriptor** `mha_audio_descriptor_t mha_audio_t::descriptor`

Dimension and description of the data.

**5.191.2.2 rdata** `mha_real_t* mha_audio_t::rdata`

Data pointer if flag `mha_audio_descriptor_t::is_complex` (p. 738) is unset.

**5.191.2.3 cdata** `mha_complex_t* mha_audio_t::cdata`

Data pointer if flag `mha_audio_descriptor_t::is_complex` (p. 738) is set.

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

- `mha.hh`

## 5.192 `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.*

### 5.192.1 Detailed Description

Channel information structure.

### 5.192.2 Member Data Documentation

**5.192.2.1 id** int mha\_channel\_info\_t::id

channel id

**5.192.2.2 idstr** char mha\_channel\_info\_t::idstr[32]

channel id

**5.192.2.3 side** unsigned int mha\_channel\_info\_t::side

side (left/right)

**5.192.2.4 dir** mha\_direction\_t mha\_channel\_info\_t::dir

source direction

**5.192.2.5 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**

## 5.193 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.*

### 5.193.1 Detailed Description

Type for complex floating point values.

### 5.193.2 Member Data Documentation

#### 5.193.2.1 **re** `mha_real_t mha_complex_t::re`

Real part.

#### 5.193.2.2 **im** `mha_real_t mha_complex_t::im`

Imaginary part.

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

- **mha.hh**

## 5.194 **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. 741) `c[]` to an array of `mha_real_t r[]` with `r[0] == c[0].re` `r[1] == c[0].im` `r[2] == c[1].re` ...

## Public Attributes

- **mha\_complex\_t c [2]**

### 5.194.1 Detailed Description

Several places in MHA rely on the fact that you can cast an array of **mha\_complex\_t** (p. 741) `c[]` to an array of **mha\_real\_t** `r[]` with `r[0] == c[0].re` `r[1] == c[0].im` `r[2] == c[1].re` ...

Check these expectations in static asserts.

### 5.194.2 Member Data Documentation

#### 5.194.2.1 C mha\_complex\_t mha\_complex\_test\_array\_t::c[2]

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

- **mha.hh**

## 5.195 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.*

### 5.195.1 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.

### 5.195.2 Member Typedef Documentation

```
5.195.2.1 value_type template<class FIFO >
typedef FIFO::value_type mha dblbuf_t< FIFO >:: value_type
```

The datatype exchanged by the FIFO and this doublebuffer.

### 5.195.3 Constructor & Destructor Documentation

```
5.195.3.1 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.

**5.195.3.2 ~mha\_dblbuf\_t()** template<class FIFO >  
**mha\_dblbuf\_t< FIFO >::~ mha\_dblbuf\_t** [virtual]

## 5.195.4 Member Function Documentation

**5.195.4.1 get\_inner\_size()** template<class FIFO >  
**virtual unsigned mha\_dblbuf\_t< FIFO >::get\_inner\_size ( ) const [inline], [virtual]**

**5.195.4.2 get\_outer\_size()** template<class FIFO >  
**virtual unsigned mha\_dblbuf\_t< FIFO >::get\_outer\_size ( ) const [inline], [virtual]**

**5.195.4.3 get\_delay()** template<class FIFO >  
**virtual unsigned mha\_dblbuf\_t< FIFO >::get\_delay ( ) const [inline], [virtual]**

**5.195.4.4 get\_fifo\_size()** template<class FIFO >  
**virtual unsigned mha\_dblbuf\_t< FIFO >::get\_fifo\_size ( ) const [inline], [virtual]**

**5.195.4.5 get\_input\_channels()** template<class FIFO >  
virtual unsigned mha\_dblbuf\_t< FIFO >::get\_input\_channels ( ) const [inline],  
[virtual]

**5.195.4.6 get\_output\_channels()** template<class FIFO >  
virtual unsigned mha\_dblbuf\_t< FIFO >::get\_output\_channels ( ) const [inline],  
[virtual]

**5.195.4.7 get\_input\_fifo\_fill\_count()** template<class FIFO >  
virtual unsigned mha\_dblbuf\_t< FIFO >::get\_input\_fifo\_fill\_count ( ) const [inline],  
[virtual]

**5.195.4.8 get\_output\_fifo\_fill\_count()** template<class FIFO >  
virtual unsigned mha\_dblbuf\_t< FIFO >::get\_output\_fifo\_fill\_count ( ) const [inline],  
[virtual]

**5.195.4.9 get\_input\_fifo\_space()** template<class FIFO >  
virtual unsigned mha\_dblbuf\_t< FIFO >::get\_input\_fifo\_space ( ) const [inline],  
[virtual]

**5.195.4.10 get\_output\_fifo\_space()** template<class FIFO >  
virtual unsigned mha\_dblbuf\_t< FIFO >::get\_output\_fifo\_space ( ) const [inline],  
[virtual]

**5.195.4.11 get\_inner\_error()** template<class FIFO >  
virtual MHA\_Error\* mha\_dblbuf\_t< FIFO >::get\_inner\_error ( ) const [inline],  
[virtual]

**5.195.4.12 `provoke_inner_error()`** template<class FIFO >  
 void **mha\_dblbuf\_t**< FIFO >::provoke\_inner\_error (  
 const **MHA\_Error** & error ) [virtual]

**5.195.4.13 `provoke_outer_error()`** template<class FIFO >  
 void **mha\_dblbuf\_t**< FIFO >::provoke\_outer\_error (  
 const **MHA\_Error** & error ) [virtual]

**5.195.4.14 `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 <i>inner_size</i> given to constructor.

#### Exceptions

<b>MHA_Error</b> (p. 760)	When count is > outer_size as given to constructor or the underlying fifo implementation detects an error.
---------------------------	--

**5.195.4.15 `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**

<b>MHA_Error</b> (p. 760)	When the underlying fifo implementation detects an error.
---------------------------	---

```
5.195.4.16 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**

<b>MHA_Error</b> (p. 760)	When the underlying fifo implementation detects an error.
---------------------------	---

**5.195.5 Member Data Documentation**

```
5.195.5.1 outer_size template<class FIFO >
unsigned mha_dblbuf_t< FIFO >::outer_size [private]
```

The block size used by the outer process.

```
5.195.5.2 inner_size template<class FIFO >
unsigned mha_dblbuf_t< FIFO >::inner_size [private]
```

The block size used by the inner process.

**5.195.5.3 `delay`** template<class FIFO >  
unsigned `mha_dblbuf_t`< FIFO >::`delay` [private]

The delay introduced by bidirectional buffer size adaptation.

**5.195.5.4 `fifo_size`** template<class FIFO >  
unsigned `mha_dblbuf_t`< FIFO >::`fifo_size` [private]

The size of each of the FIFOs.

**5.195.5 `input_channels`** template<class FIFO >  
unsigned `mha_dblbuf_t`< FIFO >::`input_channels` [private]

The number of input channels.

**5.195.6 `output_channels`** template<class FIFO >  
unsigned `mha_dblbuf_t`< FIFO >::`output_channels` [private]

The number of output channels.

**5.195.7 `input_fifo`** template<class FIFO >  
FIFO `mha_dblbuf_t`< FIFO >::`input_fifo` [private]

The FIFO for transporting the input signal from the outer process to the inner process.

**5.195.8 `output_fifo`** template<class FIFO >  
FIFO `mha_dblbuf_t`< FIFO >::`output_fifo` [private]

The FIFO for transporting the output signal from the inner process to the outer process.

**5.195.5.9 inner\_error** template<class FIFO >  
`MHA_Error* mha_dbdbuf_t< FIFO >::inner_error [private]`

Owned copy of exception to be thrown in inner thread.

**5.195.5.10 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](#)

## 5.196 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*

### 5.196.1 Detailed Description

Channel source direction structure.

### 5.196.2 Member Data Documentation

**5.196.2.1 azimuth** `mha_real_t mha_direction_t::azimuth`

azimuth in radians

**5.196.2.2 elevation** `mha_real_t mha_direction_t::elevation`

elevation in radians

**5.196.2.3 distance** `mha_real_t mha_direction_t::distance`

distance in meters

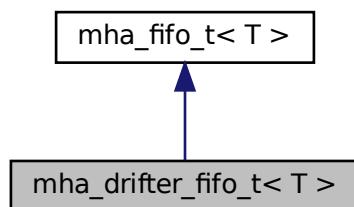
The documentation for this struct was generated from the following file:

- `mha.hh`

**5.197 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. 760) 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. 760) 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. 755) or **mha\_drifter\_fifo\_t< T >::write** (p. 755) when their xrun in succession counter exceeds its limit.*
- virtual void **starting** ()  
*Called by **mha\_drifter\_fifo\_t< T >::read** (p. 755) or **mha\_drifter\_fifo\_t< T >::write** (p. 755) when the respective flag (**mha\_drifter\_fifo\_t< T >::reader\_started** (p. 758) or **mha\_drifter\_fifo\_t< T >::writer\_started** (p. 758)) 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**

- **unsigned `reader_xruns_since_start`**

*The number of xruns seen by the writer since the last start of processing.*
- **unsigned `writer_xruns_in_succession`**

*The number of xruns seen by the reader since the last start of processing.*
- **unsigned `reader_xruns_in_succession`**

*The number of xruns seen by the writer 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. 758) and `mha_drifter_fifo_t< T >::writer_started` (p. 758) are true, then first `mha_drifter_fifo_t< T >::desired_fill_count` (p. 758) instances of `mha_drifter_fifo_t< T >::null_data` (p. 759) are delivered to the reader.*

## Additional Inherited Members

### 5.197.1 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.

### 5.197.2 Constructor & Destructor Documentation

```
5.197.2.1 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.

```
5.197.2.2 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.

### 5.197.3 Member Function Documentation

```
5.197.3.1 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. 758) to true.

When processing has started, i.e. both **reader\_started** (p. 758) and **writer\_started** (p. 758) 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. 759) exceeds **maximum\_writer\_xruns\_in\_succession\_before\_stop** (p. 759).

#### Parameters

<i>data</i>	Pointer to source data.
<i>count</i>	Number of instances to copy

Reimplemented from **mha\_fifo\_t< T >** (p. 775).

```
5.197.3.2 read() template<class T >
void mha_drifter_fifo_t< T >::read (
    T * buf,
    unsigned count ) [virtual]
```

Read data from fifo.

Sets **reader\_started** (p. 758) to true.

When processing has started, i.e. both **reader\_started** (p. 758) and **writer\_started** (p. 758) are true, then read specified amount of data from the fifo. As long as **startup\_zeros** (p. 760) is  $> 0$ , **null\_data** (p. 759) is delivered to the reader and **startup\_zeros** (p. 760) is diminished. Only when **startup\_zeros** (p. 760) 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. 757), then only so much data are read that **minimum\_fill\_count** (p. 757) entries remain in the fifo, the missing data is replaced with **null\_data** (p. 759), and the reader xrun counters are increased.

Processing is stopped when **reader\_xruns\_in\_succession** (p. 759) exceeds **maximum\_← reader\_xruns\_in\_succession\_before\_stop** (p. 759).

#### Parameters

<i>buf</i>	Pointer to the target buffer
<i>count</i>	Number of instances to copy

Reimplemented from **mha\_fifo\_t< T >** (p. 776).

```
5.197.3.3 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. 760) to the number of samples actually in the fifo's buffer.

Reimplemented from **mha\_fifo\_t< T >** (p. 776).

```
5.197.3.4 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. 760) 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. 776).

```
5.197.3.5 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.

```
5.197.3.6 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.

```
5.197.3.7 stop() template<class T >
void mha_drifter_fifo_t< T >::stop [virtual]
```

Called by **mha\_drifter\_fifo\_t<T>::read** (p. 755) or **mha\_drifter\_fifo\_t<T>::write** (p. 755) when their xrun in succession counter exceeds its limit.

Called by **read** (p. 755) or **write** (p. 755) when their xrun in succession counter exceeds its limit.

May also be called explicitly.

```
5.197.3.8 starting() template<class T >
void mha_drifter_fifo_t< T >::starting [virtual]
```

Called by **mha\_drifter\_fifo\_t<T>::read** (p. 755) or **mha\_drifter\_fifo\_t<T>::write** (p. 755) when the respective flag (**mha\_drifter\_fifo\_t<T>::reader\_started** (p. 758) or **mha\_drifter\_fifo\_t<T>::writer\_started** (p. 758)) is about to be toggled from false to true.

The fifo's buffer is emptied, this method resets **startup\_zeros** (p. 760) to **desired\_fill\_count** (p. 758), and it also resets **reader\_xruns\_since\_start** (p. 759) and **writer\_xruns\_since\_start** (p. 758) to 0.

## 5.197.4 Member Data Documentation

**5.197.4.1 minimum\_fill\_count** template<class T >  
const unsigned **mha\_drifter\_fifo\_t**< T >::minimum\_fill\_count [private]

The minimum fill count of this fifo.

**5.197.4.2 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.

**5.197.4.3 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.

**5.197.4.4 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.

**5.197.4.5 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.

**5.197.4.6 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.

**5.197.4.7 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.

**5.197.4.8 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.

**5.197.4.9 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.

**5.197.4.10 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.

**5.197.4.11 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.

**5.197.4.12 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.

```
5.197.4.13 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.

```
5.197.4.14 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. 758) and **mha\_drifter\_fifo\_t<T>::writer\_started** (p. 758) are true, then first **mha\_drifter\_fifo\_t< T >::desired\_fill\_count** (p. 758) instances of **mha\_drifter\_fifo\_t<T>::null\_data** (p. 759) are delivered to the reader.

These **null\_data** (p. 759) instances are not transmitted through the fifo because filling the fifo with enough **null\_data** (p. 759) might not be realtime safe and this filling has to be initiated by **starting** (p. 757) or **stop** (p. 757) (this implementation: **starting** (p. 757)) which are be called with realtime constraints.

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

- **mha\_fifo.h**

## 5.198 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. 760).*)
- **MHA\_Error** (const **MHA\_Error** &)
- **MHA\_Error** & **operator=** (const **MHA\_Error** &)
- **~MHA\_Error** () throw ()
- const char \* **get\_msg** () const  
*Return the error message without source position.*
- const char \* **get\_longmsg** () const  
*Return the error message with source position.*
- const char \* **what** () const throw ()  
*overwrite std::exception::what()*

## Private Attributes

- char \* **msg**
- char \* **longmsg**

### 5.198.1 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. 760) class holds source file name, line number and an error message.

### 5.198.2 Constructor & Destructor Documentation

```
5.198.2.1 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. 760).

#### Parameters

<i>s_file</i>	source file name ( <b>FILE</b> )
<i>l</i>	source line ( <b>LINE</b> )
<i>fmt</i>	format string for error message (as in printf)

```
5.198.2.2 MHA_Error() [2/2] MHA_Error::MHA_Error (
    const MHA_Error & p )
```

**5.198.2.3 ~MHA\_Error()** `MHA_Error::~MHA_Error ( ) throw ( )`

### 5.198.3 Member Function Documentation

**5.198.3.1 operator=()** `MHA_Error & MHA_Error::operator= ( const MHA_Error & p )`

**5.198.3.2 get\_msg()** `const char* MHA_Error::get_msg ( ) const [inline]`

Return the error message without source position.

**5.198.3.3 get\_longmsg()** `const char* MHA_Error::get_longmsg ( ) const [inline]`

Return the error message with source position.

**5.198.3.4 what()** `const char* MHA_Error::what ( ) const throw ( ) [inline]`

overwrite std::exception::what()

### 5.198.4 Member Data Documentation

**5.198.4.1 msg** `char* MHA_Error::msg [private]`

### 5.198.4.2 longmsg char\* MHA\_Error::longmsg [private]

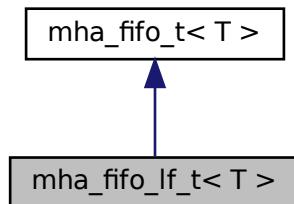
The documentation for this class was generated from the following files:

- `mha_error.hh`
- `mha_error.cpp`

## 5.199 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. 765) must only be called by the reader thread*
- virtual unsigned **get\_available\_space** () const override  
*get\_available\_space() (p. 765) 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

### 5.199.1 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. 772), adds release-acquire semantics to ensure consumer that the fill count or free space deduced from read and write pointers is consistent with the actual data. Copying, moving, and assignment of FIFO are forbidden by base class.

### 5.199.2 Constructor & Destructor Documentation

```
5.199.2.1 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

### 5.199.3 Member Function Documentation

```
5.199.3.1 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**

<b>MHA_Error</b> (p. 760)	when there is not enough space available.
---------------------------	---

Reimplemented from **mha\_fifo\_t< T >** (p. 775).

```
5.199.3.2 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**

<b>MHA_Error</b> (p. 760)	when there is not enough data available.
---------------------------	--

Reimplemented from **mha\_fifo\_t< T >** (p. 776).

```
5.199.3.3 get_fill_count() template<class T >
virtual unsigned mha_fifo_lf_t< T >::get_fill_count ( ) const [inline], [override],
[virtual]
```

**get\_fill\_count()** (p. 765) must only be called by the reader thread

Reimplemented from **mha\_fifo\_t< T >** (p. 776).

**5.199.3.4 get\_available\_space()** template<class T >  
 virtual unsigned mha\_fifo\_lf\_t< T >::get\_available\_space ( ) const [inline], [override],  
 [virtual]

**get\_available\_space()** (p. 765) must only be called by the writer thread

Reimplemented from **mha\_fifo\_t< T >** (p. 776).

#### 5.199.4 Member Data Documentation

**5.199.4.1 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

**5.199.4.2 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

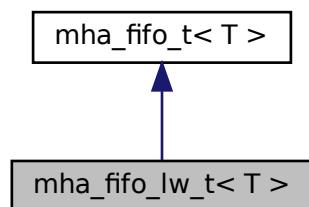
The documentation for this class was generated from the following file:

- **mha\_fifo.h**

### 5.200 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

### 5.200.1 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.

### 5.200.2 Constructor & Destructor Documentation

```
5.200.2.1 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.

**5.200.2.2 ~mha\_fifo\_lw\_t()** template<class T >  
`mha_fifo_lw_t< T >::~ mha_fifo_lw_t [virtual]`

release synchronization object

### 5.200.3 Member Function Documentation

**5.200.3.1 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

<b>MHA_Error</b> (p. <a href="#">760</a> )	when detecting a deadlock situation.
--	--------------------------------------

Reimplemented from `mha_fifo_t< T >` (p. [775](#)).

**5.200.3.2 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

<b>MHA_Error</b> (p. 760)	when detecting a deadlock situation.
---------------------------	--------------------------------------

Reimplemented from **mha\_fifo\_t< T >** (p. 776).

```
5.200.3.3 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>	<b>MHA_Error</b> (p. 760) to be thrown

## 5.200.4 Member Data Documentation

```
5.200.4.1 sync template<class T >
mha_fifo_thread_platform_t* mha_fifo_lw_t< T >::sync [private]
platform specific thread synchronization
```

```
5.200.4.2 error template<class T >
MHA_Error* mha_fifo_lw_t< T >::error[2] [private]
```

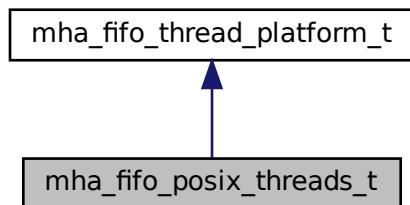
If waiting for synchronization should be aborted then exception to be thrown by reader process (index 0) or writer process (index 1) has to be placed here.

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

- **mha\_fifo.h**
- **mha\_fifo.cpp**

## 5.201 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**

### 5.201.1 Constructor & Destructor Documentation

#### 5.201.1.1 mha\_fifo\_posix\_threads\_t() mha\_fifo\_posix\_threads\_t::mha\_fifo\_posix\_posix\_threads\_t ( ) [inline]

**5.201.1.2 ~mha\_fifo\_posix\_threads\_t()** virtual mha\_fifo\_posix\_threads\_t::~mha\_fifo\_posix\_threads\_t ( ) [inline], [virtual]

## 5.201.2 Member Function Documentation

**5.201.2.1 acquire\_mutex()** virtual void mha\_fifo\_posix\_threads\_t::acquire\_mutex ( ) [inline], [virtual]

Implements **mha\_fifo\_thread\_platform\_t** (p. [782](#)).

**5.201.2.2 release\_mutex()** virtual void mha\_fifo\_posix\_threads\_t::release\_mutex ( ) [inline], [virtual]

Implements **mha\_fifo\_thread\_platform\_t** (p. [782](#)).

**5.201.2.3 wait\_for\_decrease()** virtual void mha\_fifo\_posix\_threads\_t::wait\_for\_decrease ( ) [inline], [virtual]

Implements **mha\_fifo\_thread\_platform\_t** (p. [782](#)).

**5.201.2.4 wait\_for\_increase()** virtual void mha\_fifo\_posix\_threads\_t::wait\_for\_increase ( ) [inline], [virtual]

Implements **mha\_fifo\_thread\_platform\_t** (p. [783](#)).

**5.201.2.5 increment()** virtual void mha\_fifo\_posix\_threads\_t::increment ( ) [inline], [virtual]

Implements **mha\_fifo\_thread\_platform\_t** (p. [783](#)).

**5.201.2.6 decrement()** virtual void mha\_fifo\_posix\_threads\_t::decrement ( ) [inline], [virtual]

Implements **mha\_fifo\_thread\_platform\_t** (p. 783).

### 5.201.3 Member Data Documentation

**5.201.3.1 mutex** pthread\_mutex\_t mha\_fifo\_posix\_threads\_t::mutex [private]

**5.201.3.2 decrease\_condition** pthread\_cond\_t mha\_fifo\_posix\_threads\_t::decrease\_← condition [private]

**5.201.3.3 increase\_condition** pthread\_cond\_t mha\_fifo\_posix\_threads\_t::increase\_← condition [private]

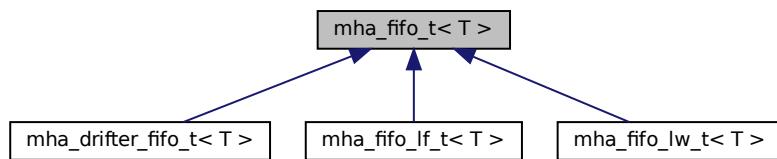
The documentation for this class was generated from the following file:

- **mha\_fifo.h**

## 5.202 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*

### 5.202.1 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.

### 5.202.2 Member Typedef Documentation

```
5.202.2.1 value_type template<class T >
typedef std::vector<T>:: value_type mha_fifo_t< T >:: value_type
```

The data type exchanged by this fifo.

### 5.202.3 Constructor & Destructor Documentation

```
5.202.3.1 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.

**5.202.3.2 ~mha\_fifo\_t()** template<class T >  
 virtual **mha\_fifo\_t< T >::~ mha\_fifo\_t ( )** [virtual], [default]

Make destructor virtual.

**5.202.3.3 mha\_fifo\_t() [2/3]** template<class T >  
**mha\_fifo\_t< T >:: mha\_fifo\_t (**  
 const **mha\_fifo\_t< T > & )** [delete]

Copy constructor.

**5.202.3.4 mha\_fifo\_t() [3/3]** template<class T >  
**mha\_fifo\_t< T >:: mha\_fifo\_t (**  
**mha\_fifo\_t< T > && )** [delete]

Move constructor.

## 5.202.4 Member Function Documentation

**5.202.4.1 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

<b>MHA_Error</b> (p. 760)	when there is not enough space available.
---------------------------	---

Reimplemented in `mha_fifo_if_t< T >` (p. 764), `mha_fifo_lw_t< T >` (p. 768), `mha_drifter_fifo_t< T >` (p. 755), `mha_fifo_if_t< mha_real_t >` (p. 764), `mha_fifo_lw_t< mha_real_t >` (p. 768), and `mha_fifo_if_t< MHA_AC::acspace2matrix_t >` (p. 764).

```
5.202.4.2 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

<b>MHA_Error</b> (p. 760)	when there is not enough data available.
---------------------------	--

Reimplemented in `mha_fifo_if_t< T >` (p. 765), `mha_fifo_lw_t< T >` (p. 768), `mha_drifter_fifo_t< T >` (p. 755), `mha_fifo_if_t< mha_real_t >` (p. 765), `mha_fifo_lw_t< mha_real_t >` (p. 768), and `mha_fifo_if_t< MHA_AC::acspace2matrix_t >` (p. 765).

```
5.202.4.3 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_if_t< T >` (p. 765), `mha_fifo_if_t< mha_real_t >` (p. 765), `mha_fifo_if_t< MHA_AC::acspace2matrix_t >` (p. 765), and `mha_drifter_fifo_t< T >` (p. 756).

```
5.202.4.4 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. 765), **mha\_fifo\_if\_t< mha\_real\_t >** (p. 765), **mha\_fifo\_if\_t< MHA\_AC::acspace2matrix\_t >** (p. 765), and **mha\_drifter\_fifo\_t< T >** (p. 756).

```
5.202.4.5 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.

```
5.202.4.6 operator=() [1/2] template<class T >
mha_fifo_t<T>& mha_fifo_t< T >::operator= (
    const mha_fifo_t< T > & ) [delete]
```

Assignment operator.

```
5.202.4.7 operator=() [2/2] template<class T >
mha_fifo_t<T>& mha_fifo_t< T >::operator= (
    mha_fifo_t< T > && ) [delete]
```

Move assignment operator.

```
5.202.4.8 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.

**5.202.4.9 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

**5.202.4.10 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

**5.202.4.11 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.

## 5.202.5 Member Data Documentation

**5.202.5.1 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.

**5.202.5.2 write\_ptr** template<class T >  
T\* mha\_fifo\_t< T >::write\_ptr [private]

points to location where to write next

**5.202.5.3 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**

## 5.203 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**

### 5.203.1 Detailed Description

Simple Mutex Guard Class.

### 5.203.2 Constructor & Destructor Documentation

**5.203.2.1 `mha_fifo_thread_guard_t()`** `mha_fifo_thread_guard_t::mha_fifo_thread_guard_t( mha_fifo_thread_platform_t * sync )` [inline]

**5.203.2.2 `~mha_fifo_thread_guard_t()`** `mha_fifo_thread_guard_t::~mha_fifo_thread_guard_t( )` [inline]

### 5.203.3 Member Data Documentation

**5.203.3.1 `sync`** `mha_fifo_thread_platform_t* mha_fifo_thread_guard_t::sync` [private]

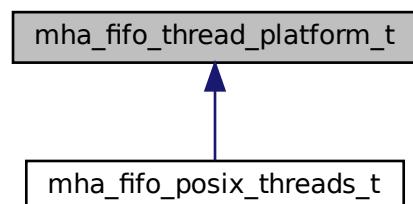
The documentation for this class was generated from the following file:

- `mha_fifo.h`

## 5.204 `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** &)

### 5.204.1 Detailed Description

Abstract base class for synchronizing multithreaded (producer/consumer) fifo operations.

Works only with single producer and single consumer.

### 5.204.2 Constructor & Destructor Documentation

#### 5.204.2.1 ~mha\_fifo\_thread\_platform\_t()

```
virtual mha_fifo_thread_platform_t::~mha_fifo_thread_platform_t ( ) [inline], [virtual]
```

Make destructor virtual.

**5.204.2.2 mha\_fifo\_thread\_platform\_t() [1/2]** `mha_fifo_thread_platform_t::mha_fifo->`  
`_thread_platform_t (`  
    `const mha_fifo_thread_platform_t & ) [private]`

**5.204.2.3 mha\_fifo\_thread\_platform\_t() [2/2]** `mha_fifo_thread_platform_t::mha_fifo->`  
`_thread_platform_t ( ) [inline]`

Make default constructor accessible.

### 5.204.3 Member Function Documentation

**5.204.3.1 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. [771](#)).

**5.204.3.2 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. [771](#)).

**5.204.3.3 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. [771](#)).

**5.204.3.4 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. [771](#)).

**5.204.3.5 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. [771](#)).

**5.204.3.6 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. [771](#)).

```
5.204.3.7 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](#)

## 5.205 mha\_real\_test\_array\_t Struct Reference

### Public Attributes

- [mha\\_real\\_t r \[4\]](#)

### 5.205.1 Member Data Documentation

#### 5.205.1.1 r mha\_real\_t mha\_real\_test\_array\_t::r[4]

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

- [mha.hh](#)

## 5.206 mha\_rt\_fifo\_element\_t< T > Class Template Reference

Object wrapper for [mha\\_rt\\_fifo\\_t](#) (p. 786).

### Public Member Functions

- [mha\\_rt\\_fifo\\_element\\_t \(T \\* data\)](#)  
*Constructor.*
- [~mha\\_rt\\_fifo\\_element\\_t \(\)](#)

## Public Attributes

- **mha\_rt\_fifo\_element\_t< T > \* next**  
*Pointer to next fifo element. NULL for the last (newest) fifo element.*
- **bool abandonned**  
*Indicates that this element will no longer be used and may be deleted.*
- **T \* data**  
*Pointer to user data.*

### 5.206.1 Detailed Description

```
template<class T>
class mha_rt_fifo_element_t< T >
```

Object wrapper for **mha\_rt\_fifo\_t** (p. 786).

### 5.206.2 Constructor & Destructor Documentation

```
5.206.2.1 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

<b>data</b>	User data. Has to be allocated on the heap with standard operator new, because it will be deleted in this element's destructor.
-------------	---

```
5.206.2.2 ~mha_rt_fifo_element_t() template<class T >
mha_rt_fifo_element_t< T >::~ mha_rt_fifo_element_t ( ) [inline]
```

### 5.206.3 Member Data Documentation

**5.206.3.1 `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.

**5.206.3.2 `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.

**5.206.3.3 `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`

## 5.207 `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. 788) or **poll\_1** (p. 788).*

### 5.207.1 Detailed Description

```
template<class T>
class mha_rt_fifo_t< T >
```

Template class for thread safe, half real time safe fifo without explicit locks.

Reading from this fifo is realtime safe, writing to it is not. This fifo is designed for objects that were constructed on the heap. It assumes ownership of these objects and calls delete on them when they are no longer used. Objects remain inside the Fifo while being used by the reader.

A new fifo element is inserted by using **push** (p. 788). The push operation is not real time safe, it allocates and deallocates memory. The latest element is retrieved by calling **poll** (p. 788). This operation will skip fifo elements if more than one **push** (p. 788) has been occurred since the last poll. To avoid skipping, call the **poll\_1** (p. 788) operation instead.

### 5.207.2 Constructor & Destructor Documentation

```
5.207.2.1 mha_rt_fifo_t() template<class T >
mha_rt_fifo_t< T >:: mha_rt_fifo_t ( ) [inline]
```

Construct empty fifo.

---

**5.207.2.2 ~mha\_rt\_fifo\_t()** template<class T >  
`mha_rt_fifo_t< T >::~ mha_rt_fifo_t ( ) [inline]`

Destructor will delete all data currently in the fifo.

### 5.207.3 Member Function Documentation

**5.207.3.1 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.

**5.207.3.2 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. 788) or **poll\_1()** (p. 788) invocation.

#### Returns

The next element in this Fifo, if there is one, or the same as before. Returns NULL if the Fifo is empty.

**5.207.3.3 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.
-------------	---

#### 5.207.3.4 remove\_abandonned() template<class T >

```
void mha_rt_fifo_t< T >::remove_abandonned ( ) [inline], [private]
```

Deletes abandonned elements.

#### 5.207.3.5 remove\_all() template<class T >

```
void mha_rt_fifo_t< T >::remove_all ( ) [inline], [private]
```

Deletes all elements.

### 5.207.4 Member Data Documentation

#### 5.207.4.1 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.

#### 5.207.4.2 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. 788) or **poll\_1** (p. 788).

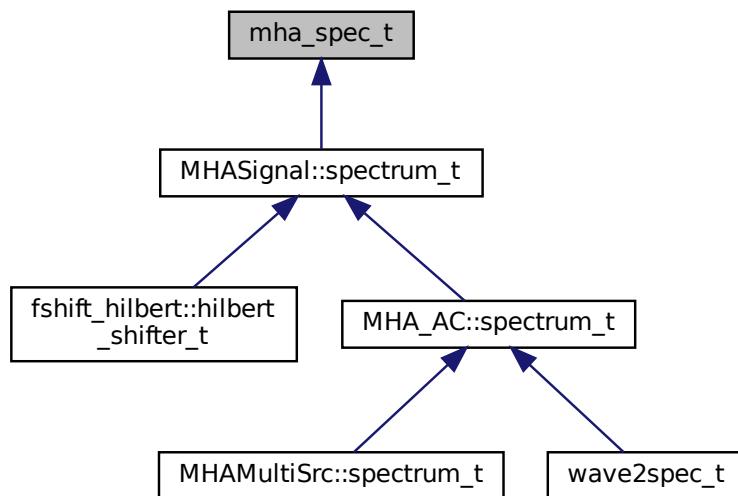
Searching for new elements starts here.

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

- **mha\_fifo.h**

## 5.208 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*

### 5.208.1 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.

<b>buf[k].re</b>	$Re(0)$	$Re(1)$	$Re(2)$	$Re(3)$	$Re(4)$	$\cdots$	$Re(n/2-1)$	$Re(n/2)$
<b>buf[k].im</b>		$Im(1)$	$Im(2)$	$Im(3)$	$Im(4)$	$\cdots$	$Im(n/2-1)$	
<b>k</b>	0	1	2	3	4		$n/2-1$	$n/2$

**Figure 4 Data order of FFT spectrum.**

Only the FFT bins for the positive frequencies, 0, and the Nyquist frequency are stored in this structure. The negative frequencies are not stored, because for a real-valued time signal they are the complex conjugates of the positive frequencies.

The negative frequencies still contribute to the signal's level. Refer to **Central Calibration** (p. 3) for a description of the scaling and how the level would be computed from the spectrum. It is recommended to use the library function **MHASignal::rmslevel** (p. 53) to compute the unweighted level correctly in Pascal, or **MHASignal::colored\_intensity** (p. 54) to compute a possibly weighted intensity.

## 5.208.2 Member Data Documentation

### 5.208.2.1 buf `mha_complex_t* mha_spec_t::buf`

signal buffer

### 5.208.2.2 num\_channels `unsigned int mha_spec_t::num_channels`

number of channels

### 5.208.2.3 num\_frames `unsigned int mha_spec_t::num_frames`

number of frames in each channel

### 5.208.2.4 **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**

## 5.209 **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.*

### 5.209.1 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.

### 5.209.2 Constructor & Destructor Documentation

```
5.209.2.1 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]
```

```
5.209.2.2 ~mha_stash_environment_variable_t() mha_stash_environment_variable_t::  
::~mha_stash_environment_variable_t ( ) [inline]
```

### 5.209.3 Member Data Documentation

```
5.209.3.1 existed_before const bool mha_stash_environment_variable_t::existed_<-  
before [private]
```

Flag indicates if the environment variable existed before constructor.

```
5.209.3.2 variable_name const std::string mha_stash_environment_variable_t::variable-<-  
_name [private]
```

Name of environment variable.

```
5.209.3.3 original_content const std::string mha_stash_environment_variable_t::  
::original_content [private]
```

Content of environment variable before constructor executed.

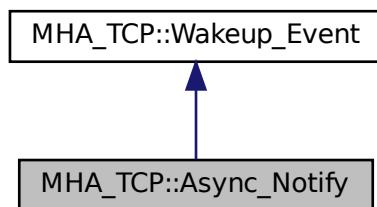
The documentation for this class was generated from the following file:

- [mha\\_os.h](#)

## 5.210 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

#### 5.210.1 Detailed Description

Portable Multiplexable cross-thread notification.

#### 5.210.2 Constructor & Destructor Documentation

**5.210.2.1 Async\_Notify()** `Async_Notify::Async_Notify ( )`**5.210.2.2 ~Async\_Notify()** `Async_Notify::~Async_Notify ( ) [virtual]`**5.210.3 Member Function Documentation****5.210.3.1 reset()** `void Async_Notify::reset ( ) [virtual]`

Reimplemented from **MHA\_TCP::Wakeup\_Event** (p. 834).

**5.210.3.2 set()** `void Async_Notify::set ( ) [virtual]`**5.210.4 Member Data Documentation****5.210.4.1 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**

**5.211 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.*

### 5.211.1 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.

### 5.211.2 Member Function Documentation

#### 5.211.2.1 `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

#### 5.211.2.2 `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.
----------------	---

**5.211.3 Member Data Documentation****5.211.3.1 streambuf** `asio::streambuf mha_tcp::buffered_socket_t::streambuf [private]`

associated streambuf object to collect received pieces into lines

**5.211.3.2 current\_message** `std::string mha_tcp::buffered_socket_t::current_message [private]`

The message that is currently sent back to the client.

**5.211.3.3 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.

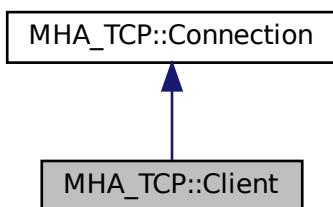
The documentation for this class was generated from the following files:

- `mha_tcp_server.hh`
- `mha_tcp_server.cpp`

**5.212 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

### 5.212.1 Detailed Description

A portable class for a tcp client connections.

### 5.212.2 Constructor & Destructor Documentation

**5.212.2.1 Client() [1/2]** Client::Client (

const std::string & host,
unsigned short port )

Constructor connects to host, port via TCP.

#### Parameters

<i>host</i>	The hostname of the TCP <b>Server</b> (p. 811).
<i>port</i>	The port or the TCP <b>Server</b> (p. 811).

**5.212.2.2 Client() [2/2]** Client::Client (

const std::string & host,
unsigned short port,
<b>Timeout_Watcher</b> & timeout_watcher )

Constructor connects to host, port via TCP, using a timeout.

### Parameters

<i>host</i>	The hostname of the TCP <b>Server</b> (p. 811).
<i>port</i>	The port or the TCP <b>Server</b> (p. 811).
<i>timeout_watcher</i>	an Event watcher that implements a timeout.

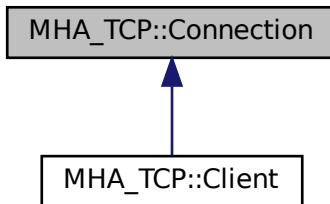
The documentation for this class was generated from the following files:

- **mha\_tcp.hh**
- **mha\_tcp.cpp**

## 5.213 MHA\_TCP::Connection Class Reference

**Connection** (p. 799) 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 TCP port.*
- **unsigned short get\_peer\_port ()**  
*Return the (protected) file descriptor of the connection.*
- **SOCKET get\_fd () const**  
*Destructor closes the underlying file descriptor.*
- **virtual ~Connection ()**
- **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**

### 5.213.1 Detailed Description

**Connection** (p. 799) handles Communication between client and server, is used on both sides.

### 5.213.2 Constructor & Destructor Documentation

**5.213.2.1 Connection()** `MHA_TCP::Connection::Connection (`  
`SOCKET _fd ) [protected]`

Create a connection instance from a socket filedescriptor.

#### Parameters

←  
↙ ↘  
fd

The file descriptor of the TCP Socket. This file descriptor is closed again in the destructor.

#### Exceptions

**MHA\_Error** (p. 760) If the file descriptor is < 0.

**5.213.2.2 ~Connection()** `Connection::~Connection ( ) [virtual]`

Destructor closes the underlying file descriptor.

### 5.213.3 Member Function Documentation

**5.213.3.1 init\_peer\_data()** void MHA\_TCP::Connection::init\_peer\_data ( ) [private]

determine peer address and port

**5.213.3.2 can\_sysread()** bool Connection::can\_sysread ( ) [private]

Determine whether at least 1 byte can be read without blocking.

**5.213.3.3 can\_syswrite()** bool Connection::can\_syswrite ( ) [private]

Determine whether at least 1 byte can be written without blocking.

**5.213.3.4 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.

**5.213.3.5 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**

<code>data</code>	A string of characters to write to the socket.
-------------------	--

**Returns**

The rest of the characters that have not yet been written.

**5.213.3.6 `get_read_event()`** `Sockread_Event * Connection::get_read_event ( )`**5.213.3.7 `get_write_event()`** `Sockwrite_Event * Connection::get_write_event ( )`**5.213.3.8 `get_peer_address()`** `std::string Connection::get_peer_address ( )`

Get peer's IP Address.

**5.213.3.9 `get_peer_port()`** `unsigned short Connection::get_peer_port ( )`

Get peer's TCP port.

**5.213.3.10 `get_fd()`** `SOCKET MHA_TCP::Connection::get_fd ( ) const [inline]`

Return the (protected) file descriptor of the connection.

Will be required for SSL.

**5.213.3.11 `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.

**5.213.3.12 `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.

**5.213.3.13 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

**5.213.3.14 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.

**5.213.3.15 `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.

**5.213.3.16 `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.
-------------------	-------------------------------

**5.213.3.17 `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.
-------------	-------------------------------

**5.213.3.18 needs\_write()** `bool Connection::needs_write ()`

Checks if the internal "outgoing" buffer contains data.

**5.213.3.19 buffered\_incoming\_bytes()** `unsigned Connection::buffered_incoming_bytes () const`

Returns the number of bytes in the internal "incoming" buffer.

**5.213.3.20 buffered\_outgoing\_bytes()** `unsigned Connection::buffered_outgoing_bytes () const`

Returns the number of bytes in the internal "outgoing" buffer.

**5.213.4 Member Data Documentation****5.213.4.1 outbuf** `std::string MHA_TCP::Connection::outbuf [private]`**5.213.4.2 inbuf** `std::string MHA_TCP::Connection::inbuf [private]`**5.213.4.3 read\_event** `Sockread_Event* MHA_TCP::Connection::read_event [private]`

**5.213.4.4 write\_event** `Sockwrite_Event* MHA_TCP::Connection::write_event [private]`

**5.213.4.5 closed** `bool MHA_TCP::Connection::closed [private]`

**5.213.4.6 peer\_addr** `struct sockaddr_in MHA_TCP::Connection::peer_addr [private]`

**5.213.4.7 fd** `SOCKET MHA_TCP::Connection::fd [protected]`

The file descriptor of the TCP Socket.

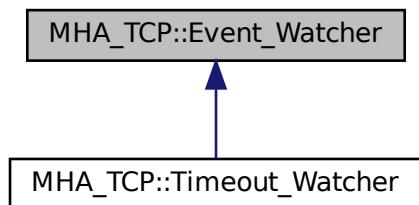
The documentation for this class was generated from the following files:

- **mha\_tcp.hh**
- **mha\_tcp.cpp**

## 5.214 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.*

### 5.214.1 Detailed Description

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

### 5.214.2 Member Typedef Documentation

#### 5.214.2.1 Events `typedef std::set< Wakeup_Event*> MHA_TCP::Event_Watcher::Events`

#### 5.214.2.2 iterator `typedef std::set< Wakeup_Event*>:: iterator MHA_TCP::Event_Watcher::iterator`

### 5.214.3 Constructor & Destructor Documentation

**5.214.3.1 ~Event\_Watcher()** Event\_Watcher::~Event\_Watcher ( ) [virtual]

### 5.214.4 Member Function Documentation

**5.214.4.1 observe()** void Event\_Watcher::observe (   
      Wakeup\_Event \* event )

Add an event to this observer.

**5.214.4.2 ignore()** void Event\_Watcher::ignore (   
      Wakeup\_Event \* event )

Remove an event from this observer.

**5.214.4.3 wait()** std::set< Wakeup\_Event \* > Event\_Watcher::wait ( )

\ Wait for some event to occur.

Return all events that are ready

### 5.214.5 Member Data Documentation

**5.214.5.1 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`

**5.215 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`};

**5.215.1 Member Enumeration Documentation****5.215.1.1 anonymous enum**   `anonymous enum`**Enumerator**

<code>R</code>	
<code>W</code>	
<code>X</code>	
<code>T</code>	

### 5.215.2 Member Data Documentation

**5.215.2.1 mode** enum { ... } MHA\_TCP::OS\_EVENT\_TYPE::mode

**5.215.2.2 fd** int MHA\_TCP::OS\_EVENT\_TYPE::fd

**5.215.2.3 timeout** double MHA\_TCP::OS\_EVENT\_TYPE::timeout

**5.215.2.4 "@5** union { ... }

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

- **mha\_tcp.hh**

## 5.216 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. 807).*
- **Connection** \* **accept** ()  
*Accept an incoming connection.*
- **Connection** \* **try\_accept** ()  
*Accept an incoming connection if it can be done without blocking.*

## Private Member Functions

- void **initialize** (const std::string & **iface**, unsigned short **port**)

## Private Attributes

- sockaddr\_in **sock\_addr**
- SOCKET **serversocket**
- std::string **iface**
- unsigned short **port**
- Sockaccept\_Event \* **accept\_event**

### 5.216.1 Constructor & Destructor Documentation

**5.216.1.1 Server() [1/2]** Server::Server (

<i>unsigned short port = 0,</i>
<i>const std::string &amp; iface = "0.0.0.0" )</i>

Create a TCP server socket.

#### Parameters

<i>port</i>	The TCP port to listen to.
<i>iface</i>	The network interface to bind to.

**5.216.1.2 Server() [2/2]** Server::Server (

<i>const std::string &amp; iface,</i>
<i>unsigned short port = 0 )</i>

Create a TCP server socket.

#### Parameters

<i>port</i>	The TCP port to listen to.
<i>iface</i>	The network interface to bind to.

**5.216.1.3 ~Server()** Server::~Server ( )

Close the TCP server socket.

**5.216.2 Member Function Documentation****5.216.2.1 initialize()** void Server::initialize ( const std::string & iface, unsigned short port ) [private]**5.216.2.2 get\_interface()** std::string Server::get\_interface ( ) const

Get the name given in the constructor for the network interface.

**5.216.2.3 get\_port()** unsigned short Server::get\_port ( ) const

Get the port that the TCP server socket currently listens to.

**5.216.2.4 get\_accept\_event()** Sockaccept\_Event \* Server::get\_accept\_event ( )

Produces an event that can be observed by an **Event\_Watcher** (p. 807).

This event signals incoming connections that can be accepted.

**5.216.2.5 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.

**5.216.2.6 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.

**5.216.3 Member Data Documentation****5.216.3.1 sock\_addr** `sockaddr_in MHA_TCP::Server::sock_addr [private]`**5.216.3.2 serversocket** `SOCKET MHA_TCP::Server::serversocket [private]`**5.216.3.3 iface** `std::string MHA_TCP::Server::iface [private]`**5.216.3.4 port** `unsigned short MHA_TCP::Server::port [private]`**5.216.3.5 accept\_event** `Sockaccept_Event* MHA_TCP::Server::accept_event [private]`

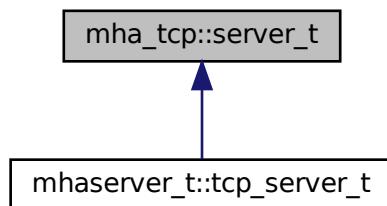
The documentation for this class was generated from the following files:

- **mha\_tcp.hh**
- **mha\_tcp.cpp**

## 5.217 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. 818) method 1 second in the future, giving us reasonably enough time for the pending responses to be sent out.*
- virtual ~**server\_t** ()=default
 

*Make destructor virtual.*
- asio::io\_context & **get\_context** ()

## Private Member Functions

- **void trigger\_accept ()**  
*Triggers the acceptance of the next connection.*
- **void post\_trigger\_read\_line (std::shared\_ptr< buffered\_socket\_t > c)**  
*Call trigger\_read\_line with a single detour through asio's event loop to avoid starving other connections when one connection floods us.*
- **void trigger\_read\_line (std::shared\_ptr< buffered\_socket\_t > c)**  
*Triggers the reading of the next line from a connection.*
- **void add\_connection (std::shared\_ptr< buffered\_socket\_t > connection)**  
*Add new connection to the list of connections, retire stale pointers.*

## Private Attributes

- **asio::io\_context io\_context**  
*The io context used to run event loops.*
- **std::shared\_ptr< asio::ip::tcp::acceptor > acceptor**  
*The underlying asio object used to accept incoming TCP connections.*
- **bool async\_accept\_has\_been\_triggered = false**  
*Set to true when async\_acceptance is triggered in trigger\_accept (Only one accept can be in process at any time).*
- **size\_t num\_accepted\_connections = 0U**  
*Number of accepted connections (not necessarily still existing)*
- **std::vector< std::weak\_ptr< buffered\_socket\_t > > connections**  
*Weak pointers to the existing connections.*

### 5.217.1 Detailed Description

Class for accepting TCP connections from clients.

### 5.217.2 Constructor & Destructor Documentation

```
5.217.2.1 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 port==0, then the operating system will select a free port.

**Exceptions**

<i>system_error</i>	if the name given in interface cannot be resolved
<i>system_error</i>	if we cannot bind to the requested interface

**5.217.2.2 ~server\_t()** `virtual mha_tcp::server_t::~server_t ( ) [virtual], [default]`

Make destructor virtual.

**5.217.3 Member Function Documentation****5.217.3.1 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.

**5.217.3.2 get\_endpoint()** `asio::ip::tcp::endpoint mha_tcp::server_t::get_endpoint ( ) const`**Returns**

The local endpoint of the acceptor.

**5.217.3.3 get\_address()** `asio::ip::address mha_tcp::server_t::get_address ( ) const`

**Returns**

The ip address that the server is bound to.

**5.217.3.4 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

**5.217.3.5 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.

**5.217.3.6 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**

<code>c</code>	the connection that has received this line
<code>l</code>	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).

### 5.217.3.7 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. 818) method 1 second in the future, giving us reasonably enough time for the pending responses to be sent out.

### 5.217.3.8 get\_context() asio::io\_context & mha\_tcp::server\_t::get\_context ( )

#### Returns

the asio io context used to run the event loop

### 5.217.3.9 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.

### 5.217.3.10 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.

### 5.217.3.11 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

<b>c</b>	The connection where the incoming data is expected from.
----------	--

**5.217.3.12 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.

## 5.217.4 Member Data Documentation

**5.217.4.1 io\_context** asio::io\_context mha\_tcp::server\_t::io\_context [private]

The io context used to run event loops.

**5.217.4.2 acceptor** std::shared\_ptr<asio::ip::tcp::acceptor> mha\_tcp::server\_t::acceptor [private]

The underlying asio object used to accept incoming TCP connections.

**5.217.4.3 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).

**5.217.4.4 num\_accepted\_connections** size\_t mha\_tcp::server\_t::num\_accepted\_connections = 0U [private]

Number of accepted connections (not necessarily still existing)

**5.217.4.5 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**

## 5.218 MHA\_TCP::sock\_init\_t Class Reference

### Public Member Functions

- **sock\_init\_t ()**

#### 5.218.1 Constructor & Destructor Documentation

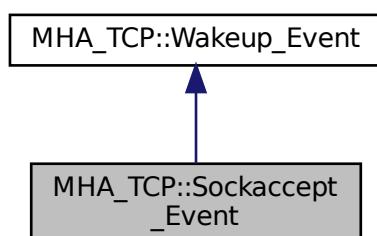
**5.218.1.1 sock\_init\_t()** MHA\_TCP::sock\_init\_t::sock\_init\_t ( ) [inline]

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

- **mha\_tcp.cpp**

## 5.219 MHA\_TCP::Sockaccept\_Event Class Reference

Inheritance diagram for MHA\_TCP::Sockaccept\_Event:



## Public Member Functions

- **Sockaccept\_Event ( SOCKET )**

## Additional Inherited Members

### 5.219.1 Constructor & Destructor Documentation

**5.219.1.1 Sockaccept\_Event()** `MHA_TCP::Sockaccept_Event::Sockaccept_Event ( SOCKET s )`

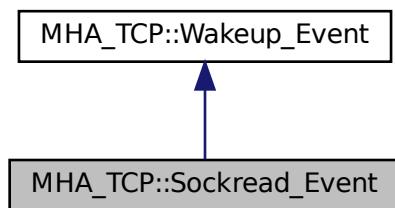
The documentation for this class was generated from the following files:

- **mha\_tcp.hh**
- **mha\_tcp.cpp**

## 5.220 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

### 5.220.1 Detailed Description

Watch socket for incoming data.

### 5.220.2 Constructor & Destructor Documentation

**5.220.2.1 Sockread\_Event()** `MHA_TCP::Sockread_Event::Sockread_Event (`  
`SOCKET s )`

Set socket to watch for.

#### Parameters

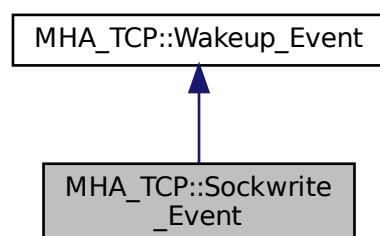
<code>s</code>	The socket to observe incoming data on.
----------------	---

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

- `mha_tcp.hh`
- `mha_tcp.cpp`

## 5.221 MHA\_TCP::Sockwrite\_Event Class Reference

Inheritance diagram for MHA\_TCP::Sockwrite\_Event:



## Public Member Functions

- **Sockwrite\_Event ( SOCKET s)**

## Additional Inherited Members

### 5.221.1 Constructor & Destructor Documentation

**5.221.1.1 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**

## 5.222 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. 824) is finished.*
- **virtual void run ()**  
*The internal method that delegated the new thread to the registered **Thread** (p. 824) 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. 760) that caused the thread to abort, if any.*

## Protected Member Functions

- **Thread ()**  
*Default constructor may only be used by derived classes that want to start the thread themselves.*

## Protected Attributes

- void \* **arg**  
*The argument for the client's thread function.*
- void \* **return\_value**  
*The return value from the client's thread function is stored here When that function returns.*

## Private Attributes

- pthread\_t **thread\_handle**  
*The posix thread handle.*
- pthread\_attr\_t **thread\_attr**  
*The posix thread attribute structure.*

### 5.222.1 Detailed Description

A very simple class for portable threads.

### 5.222.2 Member Typedef Documentation

**5.222.2.1 `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.

**5.222.3 Member Enumeration Documentation****5.222.3.1 `anonymous enum`** `anonymous enum`

The current state of the thread.

Enumerator

PREPARED	
RUNNING	
FINISHED	

**5.222.4 Constructor & Destructor Documentation****5.222.4.1 `Thread()` [1/2]** `MHA_TCP::Thread::Thread ( )` [protected]

Default constructor may only be used by derived classes that want to start the thread themselves.

**5.222.4.2 `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.

#### 5.222.4.3 ~Thread() Thread::~Thread ( ) [virtual]

The destructor should only be called when the **Thread** (p. 824) is finished.

There is preliminary support for forceful thread cancellation in the destructor, but probably not very robust or portable..

### 5.222.5 Member Function Documentation

#### 5.222.5.1 run() void Thread::run ( ) [virtual]

The internal method that delegated the new thread to the registered **Thread** (p. 824) function.

### 5.222.6 Member Data Documentation

#### 5.222.6.1 thread\_handle pthread\_t MHA\_TCP::Thread::thread\_handle [private]

The posix thread handle.

#### 5.222.6.2 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.

**5.222.6.3 arg** void\* MHA\_TCP::Thread::arg [protected]

The argument for the client's thread function.

**5.222.6.4 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.

**5.222.6.5 thread\_finish\_event** **Async\_Notify** MHA\_TCP::Thread::thread\_finish\_event

Event will be triggered when the thread exits.

**5.222.6.6 state** enum { ... } MHA\_TCP::Thread::state

The current state of the thread.

**5.222.6.7 thread\_func** **thr\_f** MHA\_TCP::Thread::thread\_func

The thread function that the client has registered.

**5.222.6.8 thread\_arg** void\* MHA\_TCP::Thread::thread\_arg

The argument that the client wants to be handed through to the thread function.

**5.222.6.9 error** **MHA\_Error\*** MHA\_TCP::Thread::error

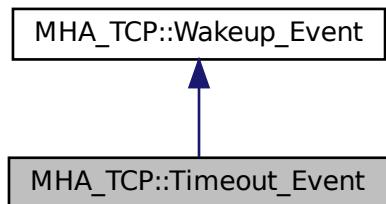
The **MHA\_Error** (p. 760) that caused the thread to abort, if any.

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

- **mha\_tcp.hh**
- **mha\_tcp.cpp**

## 5.223 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

#### 5.223.1 Constructor & Destructor Documentation

**5.223.1.1 Timeout\_Event()** `Timeout_Event::Timeout_Event (`  
`double interval )`

#### 5.223.2 Member Function Documentation

**5.223.2.1 `get_os_event()`** `os_EVENT_TYPE Timeout_Event::get_os_event () [virtual]`

Reimplemented from **MHA\_TCP::Wakeup\_Event** (p. 834).

### 5.223.3 Member Data Documentation

**5.223.3.1 `end_time`** `double MHA_TCP::Timeout_Event::end_time [private]`

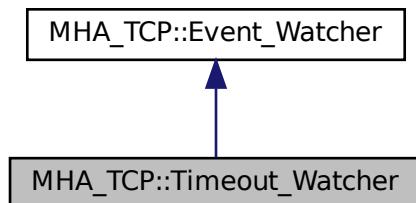
The documentation for this class was generated from the following files:

- **mha\_tcp.hh**
- **mha\_tcp.cpp**

## 5.224 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

### 5.224.1 Detailed Description

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

### 5.224.2 Constructor & Destructor Documentation

**5.224.2.1 `Timeout_Watcher()`** `Timeout_Watcher::Timeout_Watcher ( double interval ) [explicit]`

**5.224.2.2 `~Timeout_Watcher()`** `Timeout_Watcher::~Timeout_Watcher ( ) [virtual]`

### 5.224.3 Member Data Documentation

**5.224.3.1 `timeout`** `Timeout_Event MHA_TCP::Timeout_Watcher::timeout [private]`

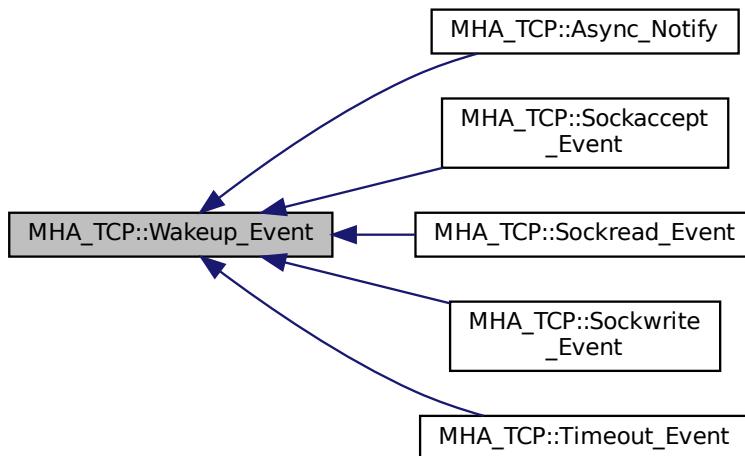
The documentation for this class was generated from the following files:

- `mha_tcp.hh`
- `mha_tcp.cpp`

## 5.225 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. 807) when this event is added to its list of observed events.*
- virtual void **ignored\_by ( Event\_Watcher \*observer)**  
*Called by the **Event\_Watcher** (p. 807) 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. 807) instances that this **Wakeup\_Event** (p. 832) is observed by (stored here for proper deregistering).*

### 5.225.1 Detailed Description

A base class for asynchronous wakeup events.

### 5.225.2 Constructor & Destructor Documentation

#### 5.225.2.1 **Wakeup\_Event()** `Wakeup_Event::Wakeup_Event ( )`

Event Constructor.

The new event has invalid state.

#### 5.225.2.2 **~Wakeup\_Event()** `Wakeup_Event::~Wakeup_Event ( ) [virtual]`

Destructor deregisters from observers.

### 5.225.3 Member Function Documentation

#### 5.225.3.1 **observed\_by()** `void Wakeup_Event::observed_by ( Event_Watcher * observer ) [virtual]`

Called by the **Event\_Watcher** (p. 807) when this event is added to its list of observed events.

---

**5.225.3.2 ignored\_by()** void Wakeup\_Event::ignored\_by ( Event\_Watcher \* observer ) [virtual]

Called by the **Event\_Watcher** (p. 807) when this event is removed from its list of observed events.

**5.225.3.3 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. 829).

**5.225.3.4 reset()** void Wakeup\_Event::reset ( ) [virtual]

For pure notification events, reset the "signalled" status.

Reimplemented in **MHA\_TCP::Async\_Notify** (p. 795).

**5.225.3.5 status()** bool Wakeup\_Event::status ( ) [virtual]

Query whether the event is in signalled state now.

## 5.225.4 Member Data Documentation

**5.225.4.1 observers** std::set<class Event\_Watcher \*> MHA\_TCP::Wakeup\_Event::observers [private]

A list of all **Event\_Watcher** (p. 807) instances that this **Wakeup\_Event** (p. 832) is observed by (stored here for proper deregistering).

**5.225.4.2 os\_event** `os_EVENT_TYPE MHA_TCP::Wakeup_Event::os_event` [protected]

**5.225.4.3 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`

## 5.226 mha\_tictoc\_t Struct Reference

### Public Attributes

- `struct timeval tv1`
- `struct timeval tv2`
- `struct timezone tz`
- `float t`

### 5.226.1 Member Data Documentation

**5.226.1.1 tv1** `struct timeval mha_tictoc_t::tv1`

**5.226.1.2 tv2** `struct timeval mha_tictoc_t::tv2`

**5.226.1.3 tz** `struct timezone mha_tictoc_t::tz`

### 5.226.1.4 **t float mha\_tictoc\_t::t**

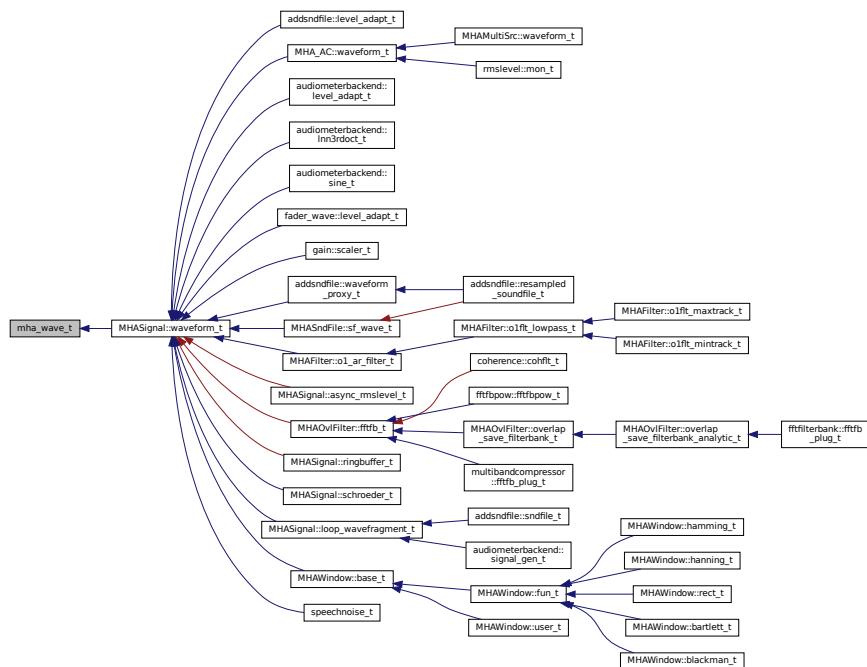
The documentation for this struct was generated from the following file:

- **mha\_profiling.h**

## 5.227 **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*

### 5.227.1 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.

### 5.227.2 Member Data Documentation

#### 5.227.2.1 buf `mha_real_t* mha_wave_t::buf`

signal buffer

#### 5.227.2.2 num\_channels `unsigned int mha_wave_t::num_channels`

number of channels

#### 5.227.2.3 num\_frames `unsigned int mha_wave_t::num_frames`

number of frames in each channel

#### 5.227.2.4 channel\_info `mha_channel_info_t* mha_wave_t::channel_info`

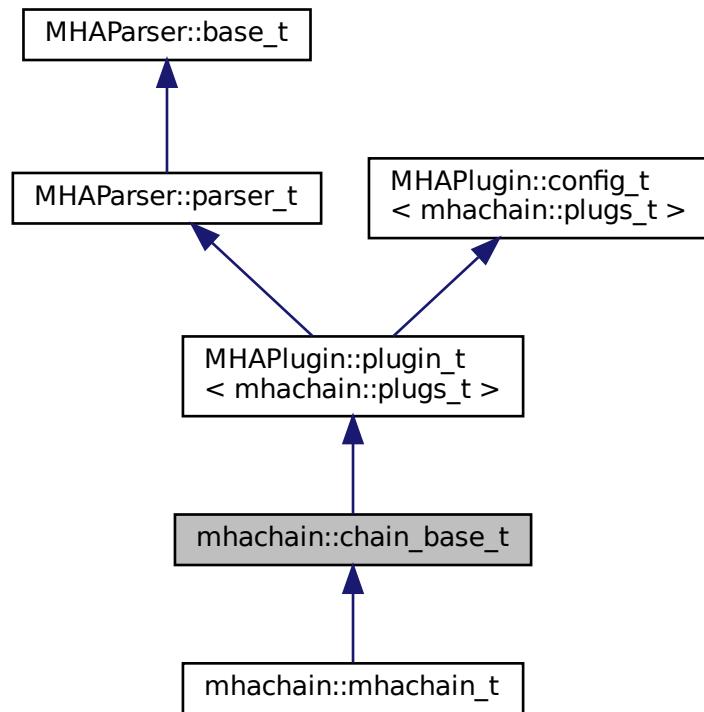
detailed channel description

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

- `mha.hh`

## 5.228 mhachain::chain\_base\_t Class Reference

Inheritance diagram for mhachain::chain\_base\_t:



### Public Member Functions

- **chain\_base\_t ( algo\_comm\_t iac, const std::string &configured\_name)**
- **void process ( mha\_wave\_t \*, mha\_wave\_t \*\*)**
- **void process ( mha\_spec\_t \*, mha\_wave\_t \*\*)**
- **void process ( mha\_wave\_t \*, mha\_spec\_t \*\*)**
- **void process ( mha\_spec\_t \*, mha\_spec\_t \*\*)**
- **void prepare ( mhaconfig\_t &)**
- **void release ()**

### Protected Attributes

- **MHAParser::bool\_t bprofiling**
- **MHAParser::vstring\_t algos**

## Private Member Functions

- void **update ()**

## Private Attributes

- std::vector< std::string > **old\_algos**
- MHAEvents::patchbay\_t< mhachain::chain\_base\_t > **patchbay**
- mhaconfig\_t **cfin**
- mhaconfig\_t **cfout**
- bool **b\_prepared**

## Additional Inherited Members

### 5.228.1 Constructor & Destructor Documentation

```
5.228.1.1 chain_base_t() mhachain::chain_base_t::chain_base_t (
    algo_comm_t iac,
    const std::string & configured_name )
```

### 5.228.2 Member Function Documentation

```
5.228.2.1 process() [1/4] void mhachain::chain_base_t::process (
    mha_wave_t * sin,
    mha_wave_t ** sout )
```

```
5.228.2.2 process() [2/4] void mhachain::chain_base_t::process (
    mha_spec_t * sin,
    mha_wave_t ** sout )
```

**5.228.2.3 process() [3/4]** void mhachain::chain\_base\_t::process (

```
mha_wave_t * sin,
mha_spec_t ** sout )
```

**5.228.2.4 process() [4/4]** void mhachain::chain\_base\_t::process (

```
mha_spec_t * sin,
mha_spec_t ** sout )
```

**5.228.2.5 prepare()** void mhachain::chain\_base\_t::prepare (

```
mhaconfig_t & cf ) [virtual]
```

Implements **MHAPlugIn::plugin\_t< mhachain::plugs\_t >** (p. 1149).

**5.228.2.6 release()** void mhachain::chain\_base\_t::release ( ) [virtual]

Reimplemented from **MHAPlugIn::plugin\_t< mhachain::plugs\_t >** (p. 1150).

**5.228.2.7 update()** void mhachain::chain\_base\_t::update ( ) [private]

### 5.228.3 Member Data Documentation

**5.228.3.1 bprofiling** MHAParser::bool\_t mhachain::chain\_base\_t::bprofiling [protected]

**5.228.3.2 algos** MHAParser::vstring\_t mhachain::chain\_base\_t::algos [protected]

**5.228.3.3 old\_algos** std::vector<std::string> mhachain::chain\_base\_t::old\_algos  
[private]

**5.228.3.4 patchbay** MHAEvents::patchbay\_t< mhachain::chain\_base\_t > mhachain->  
::chain\_base\_t::patchbay [private]

**5.228.3.5 cfin** mhaconfig\_t mhachain::chain\_base\_t::cfin [private]

**5.228.3.6 cfout** mhaconfig\_t mhachain::chain\_base\_t::cfout [private]

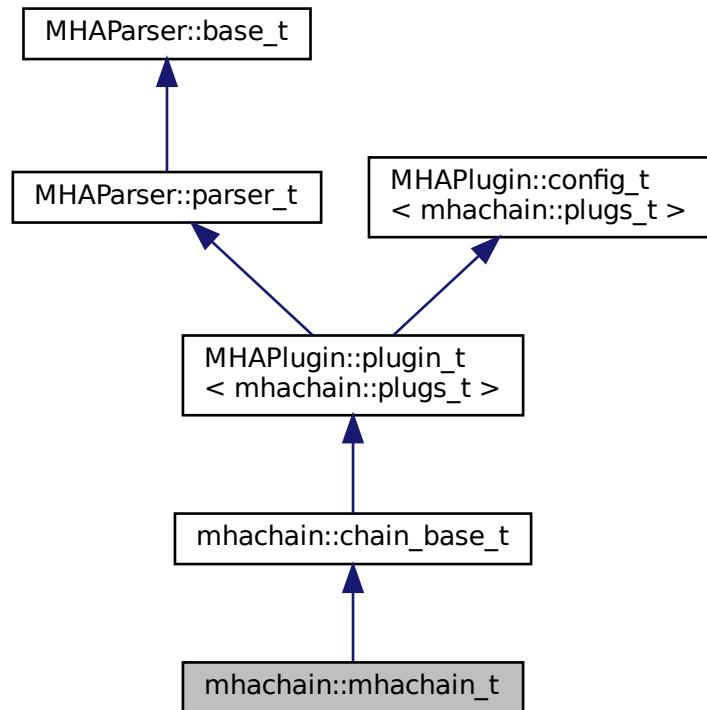
**5.228.3.7 b\_prepared** bool mhachain::chain\_base\_t::b\_prepared [private]

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

- **mha\_generic\_chain.h**
- **mha\_generic\_chain.cpp**

## 5.229 mhachain::mhachain\_t Class Reference

## Inheritance diagram for mhachain::mhachain\_t:



## Public Member Functions

- `mhachain_t ( algo_comm_t iac, const std::string &configured_name)`

## **Additional Inherited Members**

### 5.229.1 Constructor & Destructor Documentation

```
5.229.1.1 mhachain_t() mhachain::mhachain_t::mhachain_t (  
    algo_comm_t iac,  
    const std::string & configured_name )
```

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

- **mhachain.cpp**

## 5.230 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, 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**
- **MHAParser::parser\_t** **profiling**
- **MHAParser::vstring\_mon\_t** **prof\_algos**
- **MHAParser::vfloat\_mon\_t** **prof\_init**
- **MHAParser::vfloat\_mon\_t** **prof\_prepare**
- **MHAParser::vfloat\_mon\_t** **prof\_release**
- **MHAParser::vfloat\_mon\_t** **prof\_process**
- **MHAParser::float\_mon\_t** **prof\_process\_tt**
- **MHAParser::vfloat\_mon\_t** **prof\_process\_load**
- unsigned int **proc\_cnt**
- **mhaconfig\_t** **prof\_cfg**
- **MHAEvents::connector\_t**< **mhachain::plugs\_t** > **prof\_load\_con**
- **MHAEvents::connector\_t**< **mhachain::plugs\_t** > **prof\_tt\_con**
- bool **b\_use\_profiling**
- **mha\_platform\_tictoc\_t** **tictoc**

#### 5.230.1 Constructor & Destructor Documentation

**5.230.1.1 `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,`  
    `bool use_profiling )`

**5.230.1.2 `~plugs_t()`** `mhachain::plugs_t::~plugs_t ( )`

## 5.230.2 Member Function Documentation

**5.230.2.1 `prepare()`** `void mhachain::plugs_t::prepare (`  
    `mhaconfig_t & tf )`

**5.230.2.2 `release()`** `void mhachain::plugs_t::release ( )`

**5.230.2.3 `process()`** `void mhachain::plugs_t::process (`  
    `mha_wave_t * win,`  
    `mha_spec_t * sin,`  
    `mha_wave_t ** wout,`  
    `mha_spec_t ** sout )`

**5.230.2.4 `prepared()`** `bool mhachain::plugs_t::prepared ( ) const [inline]`

**5.230.2.5 alloc\_plugs()** void mhachain::plugs\_t::alloc\_plugs ( std::vector< std::string > algos ) [private]

**5.230.2.6 cleanup\_plugs()** void mhachain::plugs\_t::cleanup\_plugs ( ) [private]

**5.230.2.7 update\_proc\_load()** void mhachain::plugs\_t::update\_proc\_load ( ) [private]

### 5.230.3 Member Data Documentation

**5.230.3.1 b\_prepared** bool mhachain::plugs\_t::b\_prepared [private]

**5.230.3.2 algos** std::vector< **PluginLoader::mhapluginloader\_t\*** > mhachain::plugs\_t::algos [private]

**5.230.3.3 parser** **MHAParser::parser\_t**& mhachain::plugs\_t::parser [private]

**5.230.3.4 ac** **algo\_comm\_t** mhachain::plugs\_t::ac [private]

**5.230.3.5 profiling** **MHAParser::parser\_t** mhachain::plugs\_t::profiling [private]

**5.230.3.6 prof\_algos** `MHAParser::vstring_mon_t mhachain::plugs_t::prof_algos` [private]

**5.230.3.7 prof\_init** `MHAParser::vfloat_mon_t mhachain::plugs_t::prof_init` [private]

**5.230.3.8 prof\_prepare** `MHAParser::vfloat_mon_t mhachain::plugs_t::prof_prepare` [private]

**5.230.3.9 prof\_release** `MHAParser::vfloat_mon_t mhachain::plugs_t::prof_release` [private]

**5.230.3.10 prof\_process** `MHAParser::vfloat_mon_t mhachain::plugs_t::prof_process` [private]

**5.230.3.11 prof\_process\_tt** `MHAParser::float_mon_t mhachain::plugs_t::prof_process←_tt` [private]

**5.230.3.12 prof\_process\_load** `MHAParser::vfloat_mon_t mhachain::plugs_t::prof←_process_load` [private]

**5.230.3.13 proc\_cnt** `unsigned int mhachain::plugs_t::proc_cnt` [private]

**5.230.3.14 prof\_cfg** `mhaconfig_t mhachain::plugs_t::prof_cfg` [private]

**5.230.3.15 prof\_load\_con** `MHAEVENTS::connector_t< mhachain::plugs_t> mhachain<::plugs_t::prof_load_con` [private]

**5.230.3.16 prof\_tt\_con** `MHAEVENTS::connector_t< mhachain::plugs_t> mhachain::plugs_t::prof_tt_con` [private]

**5.230.3.17 b\_use\_profiling** `bool mhachain::plugs_t::b_use_profiling` [private]

**5.230.3.18 tictoc** `mha_platform_tictoc_t mhachain::plugs_t::tictoc` [private]

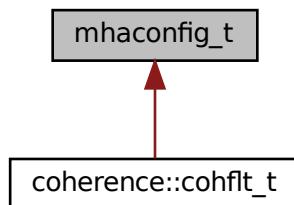
The documentation for this class was generated from the following files:

- [mha\\_generic\\_chain.h](#)
- [mha\\_generic\\_chain.cpp](#)

## 5.231 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.*

### 5.231.1 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).

### 5.231.2 Member Data Documentation

#### 5.231.2.1 **channels** `unsigned int mhaconfig_t::channels`

Number of audio channels.

#### 5.231.2.2 **domain** `unsigned int mhaconfig_t::domain`

Signal domain (MHA\_WAVEFORM or MHA\_SPECTRUM)

**5.231.2.3 fragsize** `unsigned int mhaconfig_t::fragsize`

Fragment size of waveform data.

**5.231.2.4 wndlen** `unsigned int mhaconfig_t::wndlen`

Window length of spectral data.

**5.231.2.5 fftlen** `unsigned int mhaconfig_t::fftlens`

FFT length of spectral data.

**5.231.2.6 srate** `mha_real_t mhaconfig_t::srate`

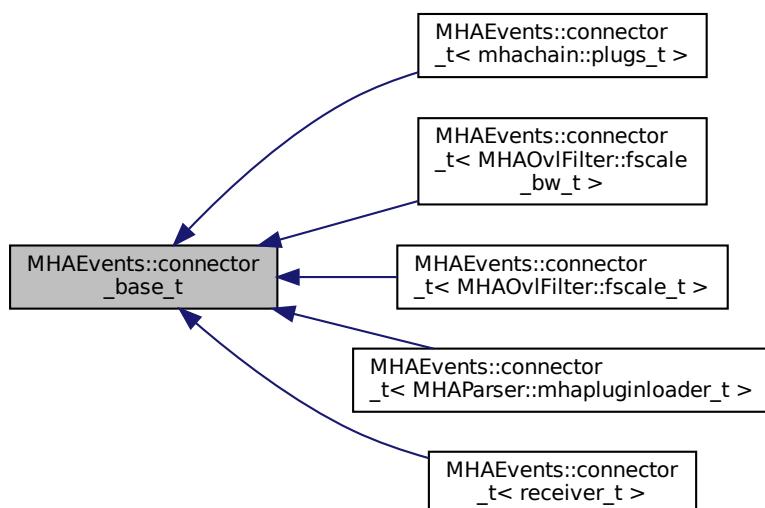
Sampling rate in Hz.

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

- `mha.hh`

**5.232 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`

### 5.232.1 Constructor & Destructor Documentation

#### 5.232.1.1 `connector_base_t()` `MHAEEvents::connector_base_t::connector_base_t()`

#### 5.232.1.2 `~connector_base_t()` `MHAEEvents::connector_base_t::~connector_base_t()` [virtual]

### 5.232.2 Member Function Documentation

#### 5.232.2.1 `emit_event()` [1/3] `void MHAEEvents::connector_base_t::emit_event()` [virtual]

Reimplemented in `MHAEEvents::connector_t< receiver_t >` (p. 853), `MHAEEvents::connector_t< MHAOvlFilter::fscale_bw_t >` (p. 853), `MHAEEvents::connector_t< MHPARSER::mhapluginloader_t >` (p. 853), `MHAEEvents::connector_t< mhachain::plugs_t >` (p. 853), and `MHAEEvents::connector_t< MHAOvlFilter::fscale_t >` (p. 853).

**5.232.2.2 emit\_event() [2/3]** void MHAEvents::connector\_base\_t::emit\_event ( const std::string & ) [virtual]

Reimplemented in **MHAEvents::connector\_t< receiver\_t >** (p. 854), **MHAEvents::connector\_t< MHAOvlFilter::fscale\_bw\_t >** (p. 854), **MHAEvents::connector\_t< MHAParser::mhapluginloader\_t >** (p. 854), **MHAEvents::connector\_t< mhachain::plugs\_t >** (p. 854), and **MHAEvents::connector\_t< MHAOvlFilter::fscale\_t >** (p. 854).

**5.232.2.3 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. 854), **MHAEvents::connector\_t< MHAOvlFilter::fscale\_bw\_t >** (p. 854), **MHAEvents::connector\_t< MHAParser::mhapluginloader\_t >** (p. 854), **MHAEvents::connector\_t< mhachain::plugs\_t >** (p. 854), and **MHAEvents::connector\_t< MHAOvlFilter::fscale\_t >** (p. 854).

**5.232.2.4 emitter\_die()** void MHAEvents::connector\_base\_t::emitter\_die ( )

### 5.232.3 Member Data Documentation

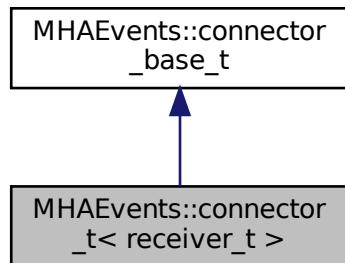
**5.232.3.1 emitter\_is\_alive** bool MHAEvents::connector\_base\_t::emitter\_is\_alive [protected]

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

- **mha\_event\_emitter.h**
- **mha\_events.cpp**

## 5.233 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

### 5.233.1 Constructor & Destructor Documentation

#### 5.233.1.1 connector\_t() [1/3] template<class receiver\_t >

```
MHAEvents::connector_t< receiver_t >:: connector_t (
    emitter_t * e,
    receiver_t * r,
    void(receiver_t::*)(rfunc )
```

#### 5.233.1.2 connector\_t() [2/3] template<class receiver\_t >

```
MHAEvents::connector_t< receiver_t >:: connector_t (
    emitter_t * e,
    receiver_t * r,
    void(receiver_t::*)(const std::string &) rfunc )
```

#### 5.233.1.3 connector\_t() [3/3] template<class receiver\_t >

```
MHAEvents::connector_t< receiver_t >:: connector_t (
    emitter_t * e,
    receiver_t * r,
    void(receiver_t::*)(const std::string &, unsigned int, unsigned int)
rfun )
```

#### 5.233.1.4 ~connector\_t() template<class receiver\_t >

```
MHAEvents::connector_t< receiver_t >::~ connector_t
```

### 5.233.2 Member Function Documentation

**5.233.2.1 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. 850).

**5.233.2.2 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. 850).

**5.233.2.3 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. 851).

### 5.233.3 Member Data Documentation

**5.233.3.1 emitter** template<class receiver\_t >  
**emitter\_t\*** **MHAEVENTS::CONNECTOR\_T**< receiver\_t >::emitter [private]

**5.233.3.2 receiver** template<class receiver\_t >  
**receiver\_t\*** **MHAEVENTS::CONNECTOR\_T**< receiver\_t >::receiver [private]

**5.233.3.3 eventhandler** template<class receiver\_t >  
**void(receiver\_t::\*) MHAEVENTS::CONNECTOR\_T< receiver\_t >::eventhandler)** () [private]

```
5.233.3.4 eventhandler_s template<class receiver_t >
void(receiver_t::* MHAEvents::connector_t< receiver_t >::eventhandler_s) (const
std::string &) [private]
```

```
5.233.3.5 eventhandler_suu template<class receiver_t >
void(receiver_t::* MHAEvents::connector_t< receiver_t >::eventhandler_suu) (const
std::string &, unsigned int, unsigned int) [private]
```

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

- **mha\_events.h**

## 5.234 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**

### 5.234.1 Detailed Description

Class for emitting openMHA events.

Use the template claas **MHAEvents::patchbay\_t** (p. 857) for connecting to an emitter.

## 5.234.2 Constructor & Destructor Documentation

**5.234.2.1 ~emitter\_t()** `MHAEEvents::emitter_t::~emitter_t ()`

## 5.234.3 Member Function Documentation

**5.234.3.1 operator()() [1/3]** `void MHAEEvents::emitter_t::operator() ()`

Emit an event without parameter.

**5.234.3.2 operator()() [2/3]** `void MHAEEvents::emitter_t::operator() ( const std::string & arg )`

Emit an event with string parameter.

**5.234.3.3 operator()() [3/3]** `void MHAEEvents::emitter_t::operator() ( const std::string & arg, unsigned int arg2, unsigned int arg3 )`

Emit an event with string parameter and two unsigned int parameters.

**5.234.3.4 connect()** `void MHAEEvents::emitter_t::connect ( connector_base_t * c )`

**5.234.3.5 disconnect()** `void MHAEEvents::emitter_t::disconnect ( connector_base_t * c )`

## 5.234.4 Member Data Documentation

**5.234.4.1 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**

## 5.235 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**

### 5.235.1 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. 858) 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.

## 5.235.2 Constructor & Destructor Documentation

**5.235.2.1 ~patchbay\_t()** template<class receiver\_t >  
**MHAEVENTS::patchbay\_t< receiver\_t >::~patchbay\_t**

## 5.235.3 Member Function Documentation

**5.235.3.1 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

**5.235.3.2 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>rfun</code>	Pointer to a member function of the receiver class

```
5.235.3.3 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)
rfun )
```

### 5.235.4 Member Data Documentation

```
5.235.4.1 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**

## 5.236 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

## 5.236.1 Constructor & Destructor Documentation

```
5.236.1.1 adapt_filter_param_t() MHAFilter::adapt_filter_param_t::adapt_filter_→
param_t (
    mha_real_t imu,
    bool ierr_in )
```

## 5.236.2 Member Data Documentation

5.236.2.1 **mu** mha\_real\_t MHAFilter::adapt\_filter\_param\_t::mu

5.236.2.2 **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**

## 5.237 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**

### 5.237.1 Constructor & Destructor Documentation

**5.237.1.1 adapt\_filter\_state\_t()** MHAFilter::adapt\_filter\_state\_t::adapt\_filter\_state\_t (

```
    int ntaps,
    int nchannels )
```

### 5.237.2 Member Function Documentation

**5.237.2.1 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 )
```

### 5.237.3 Member Data Documentation

**5.237.3.1 ntaps** int MHAFilter::adapt\_filter\_state\_t::ntaps [private]

**5.237.3.2 nchannels** int MHAFilter::adapt\_filter\_state\_t::nchannels [private]

**5.237.3.3 W** MHASignal::waveform\_t MHAFilter::adapt\_filter\_state\_t::W [private]

**5.237.3.4 X** `MHASignal::waveform_t` `MHAFilter::adapt_filter_state_t::X` [private]

**5.237.3.5 od** `MHASignal::waveform_t` `MHAFilter::adapt_filter_state_t::od` [private]

**5.237.3.6 oy** `MHASignal::waveform_t` `MHAFilter::adapt_filter_state_t::oy` [private]

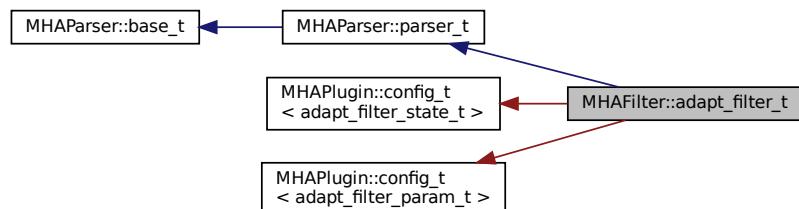
The documentation for this class was generated from the following files:

- `mha_filter.hh`
- `mha_filter.cpp`

## 5.238 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

### 5.238.1 Detailed Description

Adaptive filter.

### 5.238.2 Constructor & Destructor Documentation

**5.238.2.1 adapt\_filter\_t()** MHAFilter::adapt\_filter\_t::adapt\_filter\_t ( std::string help )

### 5.238.3 Member Function Documentation

**5.238.3.1 filter()** void MHAFilter::adapt\_filter\_t::filter ( mha\_wave\_t y, mha\_wave\_t e, mha\_wave\_t x, mha\_wave\_t d )

**5.238.3.2 set\_channelcnt()** void MHAFilter::adapt\_filter\_t::set\_channelcnt ( unsigned int nch )

**5.238.3.3 update\_mu()** void MHAFilter::adapt\_filter\_t::update\_mu ( ) [private]

**5.238.3.4 update\_ntaps()** void MHAFilter::adapt\_filter\_t::update\_ntaps ( ) [private]

#### 5.238.4 Member Data Documentation

**5.238.4.1 mu** MHAParser::float\_t MHAFilter::adapt\_filter\_t::mu [private]

**5.238.4.2 ntaps** MHAParser::int\_t MHAFilter::adapt\_filter\_t::ntaps [private]

**5.238.4.3 err\_in** MHAParser::bool\_t MHAFilter::adapt\_filter\_t::err\_in [private]

**5.238.4.4 connector** MHAEVENTS::patchbay\_t< adapt\_filter\_t> MHAFilter::adapt\_<filter\_t::connector [private]

**5.238.4.5 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**

#### 5.239 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**

### 5.239.1 Detailed Description

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

### 5.239.2 Constructor & Destructor Documentation

**5.239.2.1 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.

**5.239.2.2 ~blockprocessing\_polyphase\_resampling\_t()** virtual MHAFilter::blockprocessing←  
**\_polyphase\_resampling\_t::~blockprocessing\_polyphase\_resampling\_t( )** [inline],  
 [virtual]

## 5.239.3 Member Function Documentation

**5.239.3.1 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

<b>MHA_Error</b> (p. 760)	Raises exception if there is not enough room, if the number of channels does not match, or if the number of frames is not equal to the number specified in the constructor
---------------------------	--

**5.239.3.2 `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**

<b>MHA_Error</b> (p. <a href="#">760</a> )	Raises exception if there is not enough input signal, if the number of channels of frames does not match.
--	---

**5.239.3.3 `can_read()`** `bool MHAFilter::blockprocessing_polyphase_resampling_t::can_read ( ) const [inline]`

Checks if the resampling ring buffer can produce another output signal block.

**5.239.4 Member Data Documentation**

**5.239.4.1 `resampling`** `polyphase_resampling_t* MHAFilter::blockprocessing_polyphase_resampling_t::resampling [private]`

**5.239.4.2 `fragsize_in`** `unsigned MHAFilter::blockprocessing_polyphase_resampling_t::fragsize_in [private]`

**5.239.4.3 fragsize\_out** `unsigned MHAFilter::blockprocessing_polyphase_resampling_t::fragsize_out [private]`

**5.239.4.4 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**

## 5.240 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**

### 5.240.1 Detailed Description

Complex bandpass filter.

### 5.240.2 Constructor & Destructor Documentation

```
5.240.2.1 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

<b>A</b>	complex filter coefficients, one per band
<b>B</b>	complex weights

### 5.240.3 Member Function Documentation

**5.240.3.1 `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]`

**5.240.3.2 `creator_B()`** `std::vector< mha_complex_t > MHAFilter::complex_bandpass_`  
`← t::creator_B (`  
`std::vector< mha_complex_t > A,`  
`unsigned int order ) [static]`

**5.240.3.3 `set_state()` [1/3]** `void MHAFilter::complex_bandpass_t::set_state (`  
`mha_real_t val )`

**5.240.3.4 `set_state()` [2/3]** `void MHAFilter::complex_bandpass_t::set_state (`  
`std::vector< mha_real_t > val )`

**5.240.3.5 `set_state()` [3/3]** `void MHAFilter::complex_bandpass_t::set_state (`  
`mha_complex_t val )`

**5.240.3.6 `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.

**5.240.3.7 `get_weights()`** `std::vector< mha_complex_t > MHAFilter::complex_bandpass_`  
`← t::get_weights ( ) const [inline]`

**5.240.3.8 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.

**5.240.3.9 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.

**5.240.3.10 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.

**5.240.3.11 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.

**5.240.3.12 inspect()** std::string MHAFilter::complex\_bandpass\_t::inspect ( ) const [inline]

## 5.240.4 Member Data Documentation

**5.240.4.1 A\_** std::vector< **mha\_complex\_t** > MHAFilter::complex\_bandpass\_t::A\_ [private]

**5.240.4.2 B\_** std::vector< **mha\_complex\_t** > MHAFilter::complex\_bandpass\_t::B\_ [private]

**5.240.4.3 Yn** std::vector< **mha\_complex\_t** > MHAFilter::complex\_bandpass\_t::Yn [private]

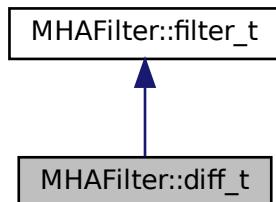
The documentation for this class was generated from the following files:

- **complex\_filter.h**
- **complex\_filter.cpp**

## 5.241 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

#### 5.241.1 Detailed Description

Differentiator class (non-normalized)

### 5.241.2 Constructor & Destructor Documentation

**5.241.2.1 diff\_t()** `MHAFilter::diff_t::diff_t ( unsigned int ch )`

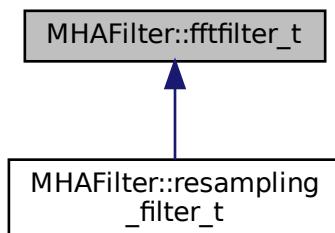
The documentation for this class was generated from the following files:

- `mha_filter.hh`
- `mha_filter.cpp`

## 5.242 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 fftlen`)  
*Constructor.*
- **~fftfilter\_t ()**
- **void update\_coeffs** (`const mha_wave_t *pwIRs`)  
*Update the set of coefficients.*
- **void filter** (`const mha_wave_t *pwIn, mha_wave_t **ppwOut, const mha_wave_t *pwIRs`)  
*Apply filter with changing coefficients to a waveform fragment.*
- **void filter** (`const mha_wave_t *pwIn, mha_wave_t **ppwOut`)  
*Apply filter to waveform fragment, without changing the coefficients.*
- **void filter** (`const mha_wave_t *pwIn, mha_wave_t **ppwOut, const mha_spec_t *psWeights`)  
*Apply filter with changing coefficients to a waveform fragment.*

## Private Attributes

- unsigned int **fragsize**
- unsigned int **channels**
- unsigned int **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**

### 5.242.1 Detailed Description

FFT based FIR filter implementation.

The maximal number of coefficients can be FFT length - fragsize + 1.

### 5.242.2 Constructor & Destructor Documentation

**5.242.2.1 fftfilter\_t()** `MHAFilter::fftfilter_t::fftfilter_t (`  
`unsigned int fragsize,`  
`unsigned int channels,`  
`unsigned int fftlen )`

Constructor.

#### Parameters

<b>fragsize</b>	Number of frames expected in input signal (each cycle).
<b>channels</b>	Number of channels expected in input signal.
<b>ffflen</b>	FFT length of filter.

**5.242.2.2 ~fftfilter\_t()** `MHAFilter::fftfilter_t::~fftfilter_t ( )`

### 5.242.3 Member Function Documentation

**5.242.3.1 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).

**5.242.3.2 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.
--------------	--

**5.242.3.3 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
---------------	---

**5.242.3.4 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.
------------------	--------------------------------------

## 5.242.4 Member Data Documentation

**5.242.4.1 fragsize** unsigned int MHAFilter::fftfilter\_t::fragsize [private]

**5.242.4.2 channels** unsigned int MHAFilter::fftfilter\_t::channels [private]

**5.242.4.3 fftlen** unsigned int MHAFilter::fftfilter\_t::ffflen [private]

**5.242.4.4 wInput\_fft** MHASignal::waveform\_t MHAFilter::fftfilter\_t::wInput\_fft [private]

**5.242.4.5 wInput** mha\_wave\_t MHAFilter::fftfilter\_t::wInput [private]

**5.242.4.6 wOutput\_fft** MHASignal::waveform\_t MHAFilter::fftfilter\_t::wOutput\_fft [private]

**5.242.4.7 wOutput** mha\_wave\_t MHAFilter::fftfilter\_t::wOutput [private]

**5.242.4.8 sInput** MHASignal::spectrum\_t MHAFilter::fftfilter\_t::sInput [private]

**5.242.4.9 sWeights** MHASignal::spectrum\_t MHAFilter::fftfilter\_t::sWeights [private]

**5.242.4.10 wIRS\_fft**    `MHASignal::waveform_t MHAFilter::fftfilter_t::wIRS_fft [private]`**5.242.4.11 fft**    `mha_fft_t MHAFilter::fftfilter_t::fft [private]`

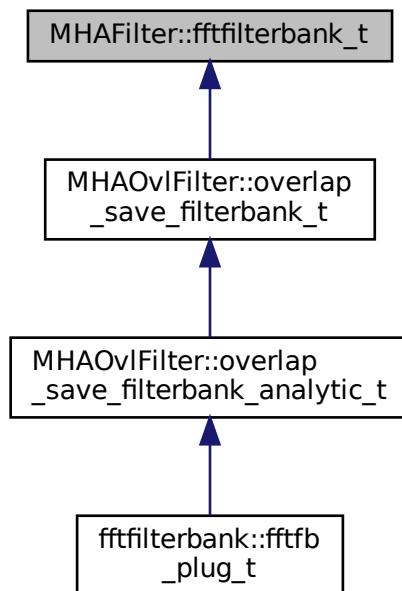
The documentation for this class was generated from the following files:

- `mha_filter.hh`
- `mha_filter.cpp`

**5.243 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*

### 5.243.1 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.

### 5.243.2 Constructor & Destructor Documentation

```
5.243.2.1 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*.

```
5.243.2.2 ~fftfilterbank_t() MHAFilter::fftfilterbank_t::~fftfilterbank_t ( )
```

### 5.243.3 Member Function Documentation

```
5.243.3.1 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.

**5.243.3.2 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<sub>in</sub></i>	Input signal pointer.
-----------------------	-----------------------

#### Return values

<i>s<sub>out</sub></i>	Pointer to output signal pointer, will be set to a valid signal
------------------------	---

#### Parameters

<i>h</i>	FIR coefficients
----------	------------------

**5.243.3.3 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<sub>in</sub></i>	Input signal pointer.
-----------------------	-----------------------

#### Return values

<i>s<sub>out</sub></i>	Pointer to output signal pointer, will be set to a valid signal
------------------------	---

**5.243.3.4 get\_irs()** const mha\_wave\_t\* MHAFilter::fftfilterbank\_t::get\_irs ( ) const  
[inline]

Return the current IRS.

## 5.243.4 Member Data Documentation

**5.243.4.1 `fragsize`** `unsigned int MHAFilter::fftfilterbank_t::fragsize [private]`

**5.243.4.2 `inputchannels`** `unsigned int MHAFilter::fftfilterbank_t::inputchannels [private]`

**5.243.4.3 `firchannels`** `unsigned int MHAFilter::fftfilterbank_t::firchannels [private]`

**5.243.4.4 `outputchannels`** `unsigned int MHAFilter::fftfilterbank_t::outputchannels [private]`

**5.243.4.5 `ffflen`** `unsigned int MHAFilter::fftfilterbank_t::ffflen [private]`

**5.243.4.6 `hw`** `MHASignal::waveform_t MHAFilter::fftfilterbank_t::hw [private]`

**5.243.4.7 `Hs`** `MHASignal::spectrum_t MHAFilter::fftfilterbank_t::Hs [private]`

**5.243.4.8 `xw`** `MHASignal::waveform_t MHAFilter::fftfilterbank_t::xw [private]`

**5.243.4.9 Xs** `MHASignal::spectrum_t` `MHAFilter::fftfilterbank_t::Xs` [private]

**5.243.4.10 yw** `MHASignal::waveform_t` `MHAFilter::fftfilterbank_t::yw` [private]

**5.243.4.11 Ys** `MHASignal::spectrum_t` `MHAFilter::fftfilterbank_t::Ys` [private]

**5.243.4.12 yw\_temp** `MHASignal::waveform_t` `MHAFilter::fftfilterbank_t::yw_temp` [private]

**5.243.4.13 tail** `MHASignal::waveform_t` `MHAFilter::fftfilterbank_t::tail` [private]

**5.243.4.14 fft** `mha_fft_t` `MHAFilter::fftfilterbank_t::fft` [private]

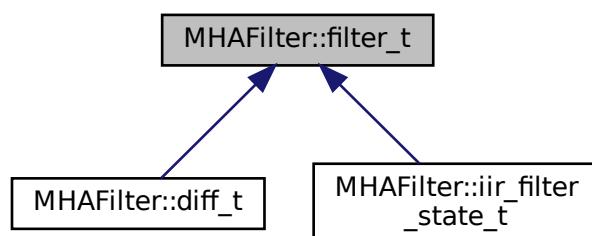
The documentation for this class was generated from the following files:

- `mha_filter.hh`
- `mha_filter.cpp`

## 5.244 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)  
*Copy constructor.*
- **filter\_t** & **operator=** (const **MHAFilter::filter\_t** &) = delete  
*Assignment operator is not implemented.*
- **~filter\_t** ()
- void **filter** ( **mha\_wave\_t** \*out, const **mha\_wave\_t** \*in)  
*Filter all channels in a waveform structure.*
- void **filter** ( **mha\_real\_t** \*dest, const **mha\_real\_t** \*src, unsigned int dframes, unsigned int frame\_dist, unsigned int channel\_dist, unsigned int channel\_begin, unsigned int channel\_end)  
*Filter parts of a waveform structure.*
- **mha\_real\_t filter** ( **mha\_real\_t** x, unsigned int ch)  
*Filter one sample.*
- unsigned int **get\_len\_A** () const  
*Return length of recursive coefficients.*
- unsigned int **get\_len\_B** () const  
*Return length of non-recursive coefficients.*

## Public Attributes

- double \* **A**  
*Pointer to recursive coefficients.*
- double \* **B**  
*Pointer to non-recursive coefficients.*

## Private Attributes

- unsigned int **len\_A**
- unsigned int **len\_B**
- unsigned int **len**
- unsigned int **channels**
- double \* **state**

### 5.244.1 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. 836) structs. The filter coefficients can be directly accessed.

### 5.244.2 Constructor & Destructor Documentation

#### 5.244.2.1 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

#### 5.244.2.2 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.

**5.244.2.3 filter\_t()** [3/3] `MHAFilter::filter_t::filter_t (`  
`const MHAFilter::filter_t & src )`

Copy constructor.

**5.244.2.4 ~filter\_t()** `MHAFilter::filter_t::~filter_t ( )`

### 5.244.3 Member Function Documentation

**5.244.3.1 operator=()** `filter_t& MHAFilter::filter_t::operator= (`  
`const MHAFilter::filter_t & ) [delete]`

Assignment operator is not implemented.

Use copy constructor instead.

**5.244.3.2 filter()** [1/3] `void MHAFilter::filter_t::filter (`  
`mha_wave_t * out,`  
`const mha_wave_t * in )`

Filter all channels in a waveform structure.

#### Parameters

<i>out</i>	Output signal
<i>in</i>	Input signal

**5.244.3.3 filter()** [2/3] `void MHAFilter::filter_t::filter (`  
`mha_real_t * dest,`  
`const mha_real_t * src,`  
`unsigned int dframes,`  
`unsigned int frame_dist,`  
`unsigned int channel_dist,`  
`unsigned int channel_begin,`  
`unsigned int channel_end )`

Filter parts of a waveform structure.

#### Parameters

<i>dest</i>	Output signal.
<i>src</i>	Input signal.
<i>dframes</i>	Number of frames to be filtered.
<i>frame_dist</i>	Index distance between frames of one channel
<i>channel_dist</i>	Index distance between audio channels
<i>channel_begin</i>	Number of first channel to be processed
<i>channel_end</i>	Number of last channel to be processed

**5.244.3.4 filter()** [3/3] `mha_real_t MHAFilter::filter_t::filter (`  
`mha_real_t x,`  
`unsigned int ch )`

Filter one sample.

#### Parameters

<i>x</i>	Input value
<i>ch</i>	Channel number to use in filter state

**5.244.3.5 get\_len\_A()** `unsigned int MHAFilter::filter_t::get_len_A ( ) const [inline]`

Return length of recursive coefficients.

**5.244.3.6 get\_len\_B()** `unsigned int MHAFilter::filter_t::get_len_B ( ) const [inline]`

Return length of non-recursive coefficients.

## 5.244.4 Member Data Documentation

**5.244.4.1 A** double\* MHAFilter::filter\_t::A

Pointer to recursive coefficients.

**5.244.4.2 B** double\* MHAFilter::filter\_t::B

Pointer to non-recursive coefficients.

**5.244.4.3 len\_A** unsigned int MHAFilter::filter\_t::len\_A [private]**5.244.4.4 len\_B** unsigned int MHAFilter::filter\_t::len\_B [private]**5.244.4.5 len** unsigned int MHAFilter::filter\_t::len [private]**5.244.4.6 channels** unsigned int MHAFilter::filter\_t::channels [private]**5.244.4.7 state** double\* MHAFilter::filter\_t::state [private]

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

- **mha\_filter.hh**
- **mha\_filter.cpp**

**5.245 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\_**

### 5.245.1 Detailed Description

Class for gammatone filter.

### 5.245.2 Constructor & Destructor Documentation

```
5.245.2.1 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 <i>cf</i> ).
<i>srate</i>	Sampling frequency in Hz.
<i>order</i>	Filter order.

**5.245.2.2 ~gamma\_flt\_t()** MHAFilter::gamma\_flt\_t::~gamma\_flt\_t ( )

## 5.245.3 Member Function Documentation

**5.245.3.1 operator() [1/3]** void MHAFilter::gamma\_flt\_t::operator() (   
*mha\_wave\_t* & *X*,  
*mha\_spec\_t* & *Y* ) [inline]

Filter method.

**5.245.3.2 operator() [2/3]** void MHAFilter::gamma\_flt\_t::operator() (   
*mha\_wave\_t* & *X*,  
*mha\_wave\_t* & *Yre*,  
*mha\_wave\_t* & *Yim* ) [inline]

Filter method.

**5.245.3.3 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.

**5.245.3.4 phase\_correction()** void MHAFilter::gamma\_flt\_t::phase\_correction ( unsigned int *desired\_delay*,  
unsigned int *inchannels* )

**5.245.3.5 set\_weights() [1/2]** void MHAFilter::gamma\_flt\_t::set\_weights ( std::vector< mha\_complex\_t > *new\_B* )

**5.245.3.6 set\_weights() [2/2]** void MHAFilter::gamma\_flt\_t::set\_weights ( unsigned int *stage*,  
std::vector< mha\_complex\_t > *new\_B* )

**5.245.3.7 get\_weights() [1/2]** std::vector< mha\_complex\_t > MHAFilter::gamma\_flt\_t::get\_weights ( ) const [inline]

**5.245.3.8 get\_weights() [2/2]** std::vector< mha\_complex\_t > MHAFilter::gamma\_flt\_t::get\_weights ( unsigned int *stage* ) const [inline]

**5.245.3.9 get\_resynthesis\_gain()** std::vector< mha\_real\_t > MHAFilter::gamma\_flt\_t::get\_resynthesis\_gain ( ) const [inline]

**5.245.3.10 reset\_state()** void MHAFilter::gamma\_flt\_t::reset\_state ( )

**5.245.3.11 get\_A()** const std::vector< mha\_complex\_t >& MHAFilter::gamma\_flt\_t::get\_A ( ) [inline]

**5.245.3.12 inspect()** std::string MHAFilter::gamma\_flt\_t::inspect () const [inline]

## 5.245.4 Member Data Documentation

**5.245.4.1 A** std::vector< mha\_complex\_t> MHAFilter::gamma\_flt\_t::A [private]

**5.245.4.2 GF** std::vector< complex\_bandpass\_t> MHAFilter::gamma\_flt\_t::GF [private]

**5.245.4.3 delay** MHASignal::delay\_t\* MHAFilter::gamma\_flt\_t::delay [private]

**5.245.4.4 envelope\_delay** std::vector<int> MHAFilter::gamma\_flt\_t::envelope\_delay [private]

**5.245.4.5 resynthesis\_gain** std::vector< mha\_real\_t> MHAFilter::gamma\_flt\_t::resynthesis←\_gain [private]

**5.245.4.6 cf\_** std::vector< mha\_real\_t> MHAFilter::gamma\_flt\_t::cf\_ [private]

**5.245.4.7 bw\_** std::vector< mha\_real\_t> MHAFilter::gamma\_flt\_t::bw\_ [private]

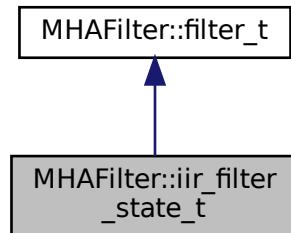
**5.245.4.8 srate\_ mha\_real\_t MHAFilter::gamma\_flt\_t::srate\_ [private]**

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

- **complex\_filter.h**
- **complex\_filter.cpp**

**5.246 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****5.246.1 Constructor & Destructor Documentation****5.246.1.1 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* )

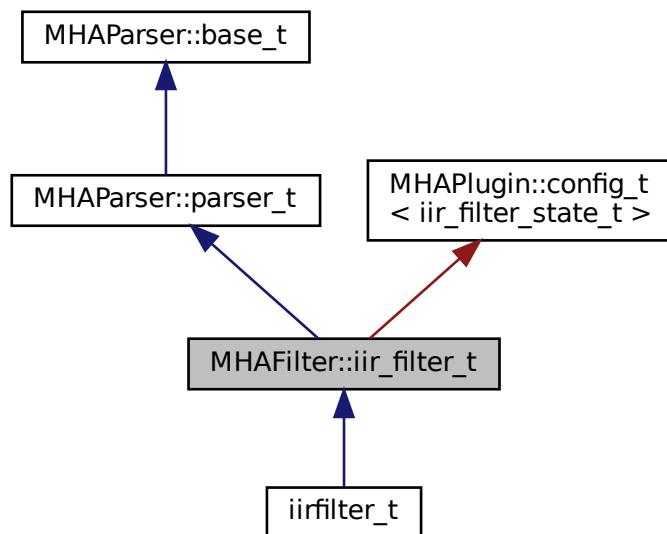
The documentation for this class was generated from the following files:

- **mha\_filter.hh**
- **mha\_filter.cpp**

## 5.247 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

### 5.247.1 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. 1098), it can easily be integrated in the openMHA configuration tree. It provides the configuration language variables "A" (vector of recursive filter coefficients) and "B" (vector of non-recursive filter coefficients).

The filter instance reacts to changes in filter coefficients through the openMHA configuration language, and uses the updated coefficients in the next invocation of the filter method.

Update of the coefficients is thread-safe and non-blocking. Simply add this subparser to your parser items and use the "filter" member function. Filter states are reset to all 0 on update.

### 5.247.2 Constructor & Destructor Documentation

```
5.247.2.1 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

### 5.247.3 Member Function Documentation

**5.247.3.1 filter() [1/2]** void MHAFilter::iir\_filter\_t::filter (

<code>mha_wave_t * y,</code>
<code>const mha_wave_t * x )</code>

The filter processes the audio signal.

All channels in the audio signal are processed using the same filter coefficients. Indipendent state is stored between calls for each audio channel.

#### Parameters

<code>y</code>	Pointer to output signal holder. The output signal is stored here. Has to have the same signal dimensions as the input signal <code>x</code> . In-place processing ( <code>y</code> and <code>x</code> pointing to the same signal holder) is possible.
<code>x</code>	Pointer to input signal holder. Number of channels has to be the same as given to the constructor, or to the <b>resize</b> (p. <a href="#">896</a> ) method.

**5.247.3.2 filter() [2/2]** `mha_real_t` MHAFilter::iir\_filter\_t::filter (

<code>mha_real_t x,</code>
<code>unsigned int ch )</code>

Filter a single audio sample.

#### Parameters

<code>x</code>	The single audio sample
<code>ch</code>	Zero-based channel index. Use and change the state of channel <code>ch</code> . <code>ch</code> has to be less than the number of channels given to the constructor or the <b>resize</b> (p. <a href="#">896</a> ) method.

#### Returns

the filtered result sample.

**5.247.3.3 `resize()`** `void MHAFilter::iir_filter_t::resize (`  
`unsigned int channels )`

Change the number of channels after object creation.

**Parameters**

<code>channels</code>	The new number of channels. Old filter states are lost.
-----------------------	---

**5.247.3.4 update\_filter()** `void MHAFilter::iir_filter_t::update_filter ( )` [private]

#### 5.247.4 Member Data Documentation

**5.247.4.1 A** `MHAParser::vfloat_t MHAFilter::iir_filter_t::A` [private]

**5.247.4.2 B** `MHAParser::vfloat_t MHAFilter::iir_filter_t::B` [private]

**5.247.4.3 connector** `MHAEvents::patchbay_t< iir_filter_t> MHAFilter::iir_filter_t::connector` [private]

**5.247.4.4 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`

#### 5.248 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`

### 5.248.1 Detailed Description

First order recursive filter.

### 5.248.2 Constructor & Destructor Documentation

#### 5.248.2.1 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)

**5.248.2.2 `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)

### 5.248.3 Member Function Documentation

**5.248.3.1 `set_state()` [1/3]** void MHAFilter::iir\_ord1\_real\_t::set\_state ( mha\_real\_t val )

**5.248.3.2 `set_state()` [2/3]** void MHAFilter::iir\_ord1\_real\_t::set\_state ( std::vector< mha\_real\_t > val )

**5.248.3.3 `set_state()` [3/3]** void MHAFilter::iir\_ord1\_real\_t::set\_state ( mha\_complex\_t val )

**5.248.3.4 `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.

**5.248.3.5 `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.

**5.248.3.6 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.

**5.248.3.7 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.

**5.248.3.8 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.

## 5.248.4 Member Data Documentation

**5.248.4.1 A\_** std::vector< mha\_real\_t> MHAFilter::iir\_ord1\_real\_t::A\_ [private]

**5.248.4.2 B\_** std::vector< mha\_real\_t> MHAFilter::iir\_ord1\_real\_t::B\_ [private]

**5.248.4.3 Yn** std::vector< mha\_complex\_t> MHAFilter::iir\_ord1\_real\_t::Yn [private]

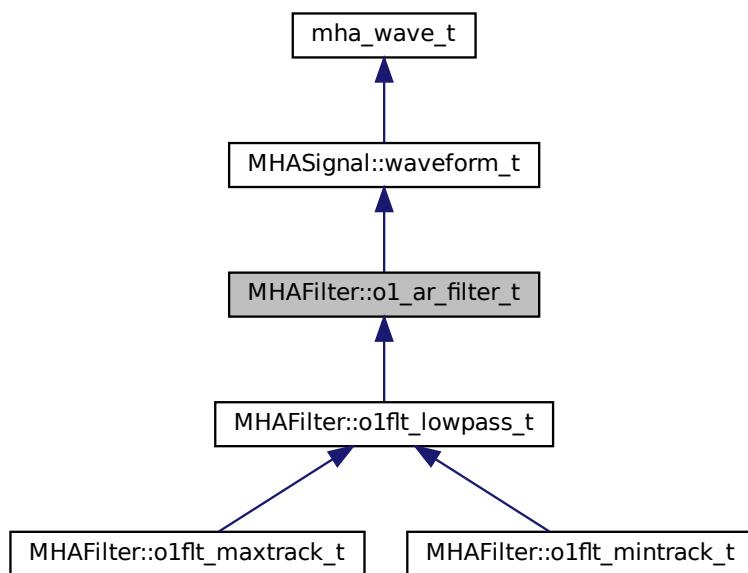
The documentation for this class was generated from the following files:

- **mha\_filter.hh**
- **mha\_filter.cpp**

## 5.249 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. 836) data.*

## Protected Attributes

- `MHASignal::waveform_t c1_a`
- `MHASignal::waveform_t c2_a`
- `MHASignal::waveform_t c1_r`
- `MHASignal::waveform_t c2_r`
- `mha_real_t fs`

## Additional Inherited Members

### 5.249.1 Detailed Description

First order attack-release lowpass filter.

This filter is the base of first order lowpass filter, maximum tracker and minimum tracker.

### 5.249.2 Constructor & Destructor Documentation

```
5.249.2.1 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

<code>channels</code>	Number of independent filters
<code>fs</code>	Sampling rate (optional, default = 1)
<code>tau_a</code>	Attack time constants (optional, default = 0)
<code>tau_r</code>	Release time constants (optional, default = 0)

### 5.249.3 Member Function Documentation

**5.249.3.1 set\_tau\_attack()** void MHAFilter::ol\_ar\_filter\_t::set\_tau\_attack ( unsigned int *ch*, mha\_real\_t *tau* )

Set the attack time constant.

#### Parameters

<i>ch</i>	Channel number
<i>tau</i>	Time constant

**5.249.3.2 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

**5.249.3.3 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

**5.249.3.4 operator()() [2/2]** void MHAFilter::o1\_ar\_filter\_t::operator() ( const mha\_wave\_t & in, mha\_wave\_t & out ) [inline]

Apply filter to a **mha\_wave\_t** (p. 836) data.

**Parameters**

<i>in</i>	Input signal
<i>out</i>	Output signal

The number of channels must match the number of filter bands.

**5.249.4 Member Data Documentation**

**5.249.4.1 c1\_a** MHASignal::waveform\_t MHAFilter::o1\_ar\_filter\_t::c1\_a [protected]

**5.249.4.2 c2\_a** MHASignal::waveform\_t MHAFilter::o1\_ar\_filter\_t::c2\_a [protected]

**5.249.4.3 c1\_r** MHASignal::waveform\_t MHAFilter::o1\_ar\_filter\_t::c1\_r [protected]

**5.249.4.4 c2\_r** MHASignal::waveform\_t MHAFilter::o1\_ar\_filter\_t::c2\_r [protected]

### 5.249.4.5 **fs** `mha_real_t` MHAFilter::o1\_ar\_filter\_t::fs [protected]

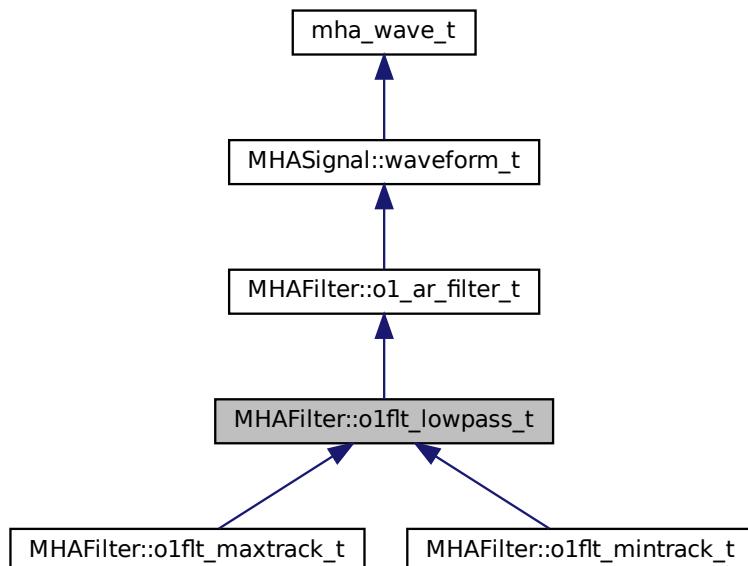
The documentation for this class was generated from the following files:

- `mha_filter.hh`
- `mha_filter.cpp`

## 5.250 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

### 5.250.1 Detailed Description

First order low pass filter.

### 5.250.2 Constructor & Destructor Documentation

**5.250.2.1 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

**5.250.2.2 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

### 5.250.3 Member Function Documentation

**5.250.3.1 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

**5.250.3.2 set\_tau() [2/2]** void MHAFilter::o1flt\_lowpass\_t::set\_tau ( mha\_real\_t tau )

set time constant in all channels to tau

**5.250.3.3 get\_c1()** mha\_real\_t MHAFilter::o1flt\_lowpass\_t::get\_c1 ( unsigned int ch ) const [inline]

**5.250.3.4 get\_last\_output()** mha\_real\_t MHAFilter::o1flt\_lowpass\_t::get\_last\_output ( unsigned int ch ) const [inline]

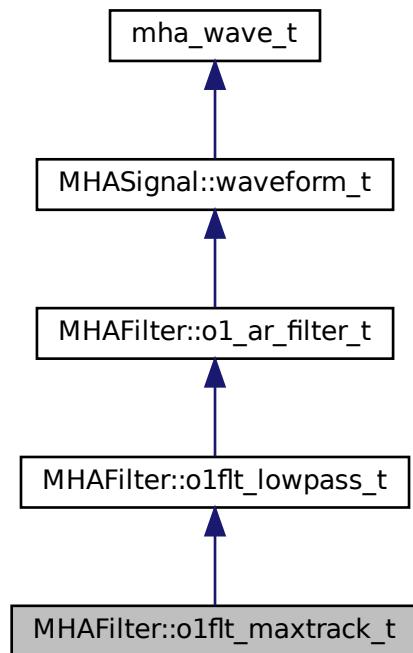
The documentation for this class was generated from the following files:

- [mha\\_filter.hh](#)
- [mha\\_filter.cpp](#)

## 5.251 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

#### 5.251.1 Detailed Description

First order maximum tracker.

## 5.251.2 Constructor & Destructor Documentation

**5.251.2.1 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

**5.251.2.2 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 )
```

## 5.251.3 Member Function Documentation

**5.251.3.1 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

**5.251.3.2 set\_tau() [2/2]** void MHAFilter::o1flt\_maxtrack\_t::set\_tau (

```
mha_real_t tau )
```

set time constant in all channels to tau

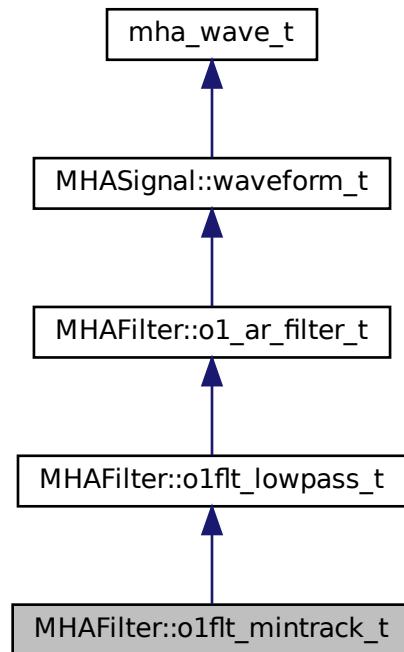
The documentation for this class was generated from the following files:

- **mha\_filter.hh**
- **mha\_filter.cpp**

## 5.252 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

#### 5.252.1 Detailed Description

First order minimum tracker.

## 5.252.2 Constructor & Destructor Documentation

**5.252.2.1 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 )
```

**5.252.2.2 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 )
```

## 5.252.3 Member Function Documentation

**5.252.3.1 set\_tau() [1/2]** void MHAFilter::o1flt\_mintrack\_t::set\_tau (

```
unsigned int ch,
mha_real_t tau )
```

change the time constant in one channel

**5.252.3.2 set\_tau() [2/2]** void MHAFilter::o1flt\_mintrack\_t::set\_tau (

```
mha_real_t tau )
```

set time constant in all channels to tau

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

- **mha\_filter.hh**
- **mha\_filter.cpp**

## 5.253 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.*

### 5.253.1 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.

### 5.253.2 Constructor & Destructor Documentation

```
5.253.2.1 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.

```
5.253.2.2 ~partitioned_convolution_t() MHAFilter::partitioned_convolution_t::~partitioned_convolution_t ( )
```

Free fftw resource allocated in constructor.

### 5.253.3 Member Function Documentation

**5.253.3.1 process()** `mha_wave_t * MHAFilter::partitioned_convolution_t::process ( const mha_wave_t * s_in )`

processing

### 5.253.4 Member Data Documentation

**5.253.4.1 fragsize** `unsigned int MHAFilter::partitioned_convolution_t::fragsize`

Audio fragment size, always equal to partition size.

**5.253.4.2 nchannels\_in** `unsigned int MHAFilter::partitioned_convolution_t::nchannels_in`

Number of audio input channels.

**5.253.4.3 nchannels\_out** `unsigned int MHAFilter::partitioned_convolution_t::nchannels_out`

Number of audio output channels.

**5.253.4.4 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.

**5.253.4.5 filter\_partitions** `unsigned int MHAFilter::partitioned_convolution_t::filter_partitions`

The total number of non-zero impulse response partitions.

**5.253.4.6 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

**5.253.4.7 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.

**5.253.4.8 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).

**5.253.4.9 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).

**5.253.4.10 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.

---

**5.253.4.11 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.

**5.253.4.12 current\_output\_partition\_index** unsigned int MHAFilter::partitioned\_convolution\_t::current\_output\_partition\_index

A counter modulo output\_partitions, indexing the "current" output partition.

**5.253.4.13 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

**5.253.4.14 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**

## 5.254 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.*

### 5.254.1 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.

### 5.254.2 Constructor & Destructor Documentation

**5.254.2.1 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

**5.254.2.2 index\_t() [2/2]** MHAFilter::partitioned\_convolution\_t::index\_t::index\_t ( )  
[inline]

Default constructor for STL compatibility.

### 5.254.3 Member Data Documentation

**5.254.3.1 source\_channel\_index** unsigned int MHAFilter::partitioned\_convolution\_t::index\_t::source\_channel\_index

The input channel index to apply the current partition to.

**5.254.3.2 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.

**5.254.3.3 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**

## 5.255 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\_ir, unsigned n\_ringbuffer, unsigned n\_channels, unsigned n\_prefill)  
*Construct a polyphase resampler instance.*
- **void write ( mha\_wave\_t &signal)**  
*Write signal to the ringbuffer.*
- **void read ( mha\_wave\_t &signal)**  
*Read resampled signal.*
- **unsigned readable\_frames () const**  
*Number of frames at target sampling rate that can be produced.*

## Private Attributes

- unsigned **upsampling\_factor**  
*Integer upsampling factor.*
- unsigned **downsampling\_factor**  
*Integer downsampling factor.*
- unsigned **now\_index**  
*Index of "now" in the interpolated sampling rate.*
- bool **underflow**  
*Set to true when an underflow has occurred.*
- **MHAWindow::hanning\_t impulse\_response**  
*Contains the impulse response of the lowpass filter needed for anti-aliasing.*
- **MHASignal::ringbuffer\_t ringbuffer**  
*Storage of input signal.*

### 5.255.1 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  $n^{\text{th}}$  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  $n^{\text{th}}$  sample needs to be computed.

When resampling from one sampling rate to another where neither is a multiple of the other, the signal first needs to be upsampled to a sampling rate that is a multiple of both (source and target) sampling rates, and then downsampled again to the target sampling rate. Instead of applying two separate lowpass filters directly after each other (one filter for upsampling and another for downsampling), it is sufficient to apply only one low-pass filter, when producing the output at the final target rate, with a cut-off frequency equal to the lower cut-off-frequency of the replaced two low-pass filters. Not filtering to produce a filtered signal already at the common multiple sampling rate has the side effect that this intermediate signal at the common multiple sampling rate keeps its filler zero samples unaltered. These zero samples can be taken advantage of when filtering to produce the output at the target rate: The zeros do not need to be multiplied with their corresponding filter coefficients, because the result is known to be zero again, and this zero product has no effect on the summation operation to compute a target sample at the target rate. To summarize, the following optimization techniques are available:

- The signal does not need to be stored in memory at the interpolation rate. It is sufficient to have the signal available at the source rate and to know where the zeros would be.
- The signal needs to be low-pass-filtered only once.

- The FIR low-pass filtering can take advantage of
  - computing only filter outputs for the required samples at the target rate,
  - skipping over zero-samples at the interpolation rate.

The procedure that takes advantage of these optimization possibilites is known as polyphase resampling.

This class implements polyphase resampling in this way for a source sampling rate and a target sampling rate that have common multiple, the interpolation sampling rate. Non-rational and drifting sample rates are outside the scope of this resampler.

## 5.255.2 Constructor & Destructor Documentation

```
5.255.2.1 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)

### 5.255.3 Member Function Documentation

**5.255.3.1 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

<code>signal</code>	input signal in original sampling rate
---------------------	--

#### Exceptions

<b>MHA_Error</b> (p. <a href="#">760</a> )	Raises exception if there is not enough room or if the number of channels does not match.
--	---

**5.255.3.2 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

<code>signal</code>	buffer to write the resampled signal to.
---------------------	--

#### Exceptions

<b>MHA_Error</b> (p. <a href="#">760</a> )	Raises exception if there is not enough input signal or if the number of channels is too high.
--	--

**5.255.3.3 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. 922) ensures that enough past samples are present to compute the next target sample.

#### 5.255.4 Member Data Documentation

##### 5.255.4.1 upsampling\_factor `unsigned MHAFilter::polyphase_resampling_t::upsampling_factor [private]`

Integer upsampling factor.

Interpolation rate divided by source rate.

##### 5.255.4.2 downsampling\_factor `unsigned MHAFilter::polyphase_resampling_t::downsampling_factor [private]`

Integer downsampling factor.

Interpolation rate divided by target rate.

##### 5.255.4.3 now\_index `unsigned MHAFilter::polyphase_resampling_t::now_index [private]`

Index of "now" in the interpolated sampling rate.

##### 5.255.4.4 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. 922) are present before calling **read** (p. 922)

**5.255.4.5 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.

**5.255.4.6 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.

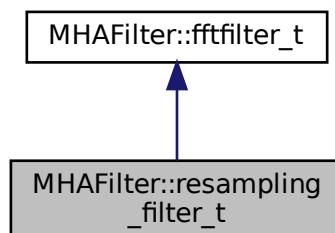
The documentation for this class was generated from the following files:

- `mha_filter.hh`
- `mha_filter.cpp`

## 5.256 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 **irslen**, 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 **irslen**)

## Private Attributes

- unsigned int **fragsize**

### 5.256.1 Detailed Description

Hann shaped low pass filter for resampling.

This class uses FFT filter at upsampled rate.

### 5.256.2 Constructor & Destructor Documentation

```
5.256.2.1 resampling_filter_t() MHAFilter::resampling_filter_t
(
    unsigned int ffflen,
    unsigned int irslen,
    unsigned int channels,
    unsigned int Nup,
    unsigned int Ndown,
    double fCutOff )
```

Constructor.

#### Parameters

<i>ffflen</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)

### 5.256.3 Member Function Documentation

**5.256.3.1 fragsize\_validator()** `unsigned int MHAFilter::resampling_filter_t::fragsize←_validator (`  
`unsigned int fftlen,`  
`unsigned int irslen ) [static]`

### 5.256.4 Member Data Documentation

**5.256.4.1 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`

## 5.257 MHAFilter::smoothspec\_t Class Reference

Smooth spectral gains, create a windowed impulse response.

### Public Member Functions

- **smoothspec\_t** (`unsigned int fftlen, unsigned int nchannels, const MHAWindow←::base_t & window, bool minphase, bool linphase_asym=false)`  
*Constructor.*
- **void smoothspec** (`const mha_spec_t &s_in, mha_spec_t &s_out)`  
*Create a smoothed spectrum.*
- **void smoothspec** (`mha_spec_t &spec)`  
*Create a smoothed spectrum (in place)*
- **void spec2fir** (`const mha_spec_t &spec, mha_wave_t &fir)`  
*Return FIR coefficients.*
- **~smoothspec\_t ()**

## Private Member Functions

- void **internal\_fir** (const **mha\_spec\_t** &)

## Private Attributes

- unsigned int **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**

### 5.257.1 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. 929) 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. 928) must correspond to the FFT length and number of channels provided in the constructor. The function **spec2fir()** (p. 929) can fill signal structures with more than window length frames.

### 5.257.2 Constructor & Destructor Documentation

```
5.257.2.1 smoothspec_t() MHAFilter::smoothspec_t::smoothspec_t (
    unsigned int ffflen,
    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

**5.257.2.2 ~smoothspec\_t()** `MHAFilter::smoothspec_t::~smoothspec_t ( )`

### 5.257.3 Member Function Documentation

**5.257.3.1 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
--------------	-----------------

**5.257.3.2 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.
-------------	--------------------------

**5.257.3.3 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
------------	---

**5.257.3.4 internal\_fir()** void MHAFilter::smoothspec\_t::internal\_fir ( const mha\_spec\_t & *s\_in* ) [private]

## 5.257.4 Member Data Documentation

**5.257.4.1 fftlen** unsigned int MHAFilter::smoothspec\_t::fftlen [private]

**5.257.4.2 nchannels** unsigned int MHAFilter::smoothspec\_t::nchannels [private]

**5.257.4.3 window** `MHAWindow::base_t MHAFilter::smoothspec_t::window` [private]

**5.257.4.4 tmp\_wave** `MHASignal::waveform_t MHAFilter::smoothspec_t::tmp_wave` [private]

**5.257.4.5 tmp\_spec** `MHASignal::spectrum_t MHAFilter::smoothspec_t::tmp_spec` [private]

**5.257.4.6 minphase** `MHASignal::minphase_t* MHAFilter::smoothspec_t::minphase` [private]

**5.257.4.7 \_linphase\_asym** `bool MHAFilter::smoothspec_t::_linphase_asym` [private]

**5.257.4.8 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`

## 5.258 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**

### 5.258.1 Constructor & Destructor Documentation

```
5.258.1.1 thirddoctave_analyzer_t() MHAFilter::thirddoctave_analyzer_t::thirddoctave_analyzer_t ( mhaconfig_t cfg )
```

### 5.258.2 Member Function Documentation

```
5.258.2.1 process() mha_wave_t * MHAFilter::thirddoctave_analyzer_t::process ( mha_wave_t * sIn )
```

```
5.258.2.2 nbands() unsigned int MHAFilter::thirddoctave_analyzer_t::nbands ( )
```

```
5.258.2.3 nchannels() unsigned int MHAFilter::thirddoctave_analyzer_t::nchannels ( )
```

**5.258.2.4 get\_cf\_hz()** std::vector< **mha\_real\_t** > MHAFilter::thirdoctave\_analyzer\_t::get\_cf\_hz ( )

**5.258.2.5 cf\_generator()** std::vector< **mha\_real\_t** > MHAFilter::thirdoctave\_analyzer\_t::cf\_generator ( **mhaconfig\_t** cfg ) [static]

**5.258.2.6 bw\_generator()** std::vector< **mha\_real\_t** > MHAFilter::thirdoctave\_analyzer\_t::bw\_generator ( **mhaconfig\_t** cfg ) [static]

**5.258.2.7 dup()** std::vector< **mha\_real\_t** > MHAFilter::thirdoctave\_analyzer\_t::dup ( std::vector< **mha\_real\_t** > vec, **mhaconfig\_t** cfg ) [static]

### 5.258.3 Member Data Documentation

**5.258.3.1 cfg\_** **mhaconfig\_t** MHAFilter::thirdoctave\_analyzer\_t::cfg\_ [private]

**5.258.3.2 cf** std::vector< **mha\_real\_t**> MHAFilter::thirdoctave\_analyzer\_t::cf [private]

**5.258.3.3 fb** **MHAFilter::gamma\_flt\_t** MHAFilter::thirdoctave\_analyzer\_t::fb [private]

**5.258.3.4 out\_chunk** `MHASignal::waveform_t MHAFilter::thirdoctave_analyzer_t::out_chunk [private]`

**5.258.3.5 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`

## 5.259 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.*

### 5.259.1 Detailed Description

a structure containing a source channel number, a target channel number, and an impulse response.

### 5.259.2 Constructor & Destructor Documentation

**5.259.2.1 transfer\_function\_t() [1/2]** MHAFilter::transfer\_function\_t::transfer\_↔  
function\_t ( ) const [inline]

Default constructor for STL conformity.

Not used.

**5.259.2.2 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

### 5.259.3 Member Function Documentation

**5.259.3.1 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

**5.259.3.2 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.

**5.259.3.3 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.

## 5.259.4 Member Data Documentation

**5.259.4.1 source\_channel\_index** `unsigned int MHAFilter::transfer_function_t::source_channel_index`

Source audio channel index for this transfer function.

**5.259.4.2 target\_channel\_index** `unsigned int MHAFilter::transfer_function_t::target_channel_index`

Target audio channel index for this transfer function.

**5.259.4.3 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`

## 5.260 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. 937) 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. 937) method on every matrix member.*

### 5.260.1 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.

### 5.260.2 Member Function Documentation

```
5.260.2.1 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. 937) method on every matrix member.

```
5.260.2.2 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. 937) method on every matrix member.

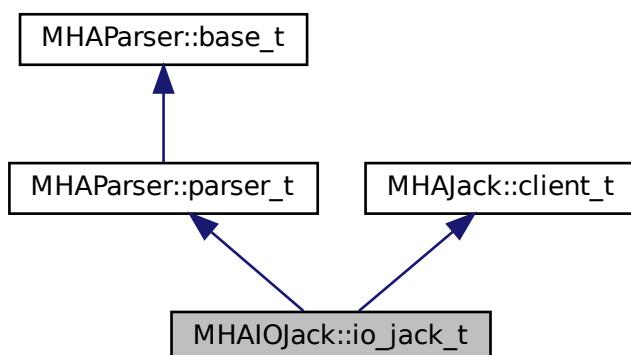
The documentation for this struct was generated from the following file:

- **mha\_filter.hh**

## 5.261 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

### 5.261.1 Detailed Description

Main class for JACK IO.

This class registers a JACK client. JACK and framework states are managed by this class.

### 5.261.2 Constructor & Destructor Documentation

```
5.261.2.1 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 )
```

### 5.261.3 Member Function Documentation

```
5.261.3.1 prepare() void io_jack_t::prepare (
    int nch_in,
    int nch_out )
```

Allocate buffers, activate JACK client and install internal ports.

```
5.261.3.2 release() void io_jack_t::release ( )
```

**5.261.3.3 reconnect\_inports()** void io\_jack\_t::reconnect\_inports ( ) [private]

Connect the input ports when connection variable is accessed.

**5.261.3.4 reconnect\_outports()** void io\_jack\_t::reconnect\_outports ( ) [private]

Connect the output ports when connection variable is accessed.

**5.261.3.5 get\_physical\_input\_ports()** void io\_jack\_t::get\_physical\_input\_ports ( ) [private]**5.261.3.6 get\_physical\_output\_ports()** void io\_jack\_t::get\_physical\_output\_ports ( ) [private]**5.261.3.7 get\_all\_input\_ports()** void io\_jack\_t::get\_all\_input\_ports ( ) [private]**5.261.3.8 get\_all\_output\_ports()** void io\_jack\_t::get\_all\_output\_ports ( ) [private]**5.261.3.9 get\_delays\_in()** void io\_jack\_t::get\_delays\_in ( ) [private]**5.261.3.10 get\_delays\_out()** void io\_jack\_t::get\_delays\_out ( ) [private]

**5.261.3.11 read\_get\_cpu\_load()** void io\_jack\_t::read\_get\_cpu\_load ( ) [private]

**5.261.3.12 read\_get\_xruns()** void io\_jack\_t::read\_get\_xruns ( ) [private]

**5.261.3.13 read\_get\_scheduler()** void io\_jack\_t::read\_get\_scheduler ( ) [private]

#### 5.261.4 Member Data Documentation

**5.261.4.1 fw\_fragsize** unsigned int MHAIOJack::io\_jack\_t::fw\_fragsize [private]

**5.261.4.2 fw\_samplerate** float MHAIOJack::io\_jack\_t::fw\_samplerate [private]

**5.261.4.3 servername** **MHAParser::string\_t** MHAIOJack::io\_jack\_t::servername [private]

**5.261.4.4 clientname** **MHAParser::string\_t** MHAIOJack::io\_jack\_t::clientname [private]

**5.261.4.5 connections\_in** **MHAParser::vstring\_t** MHAIOJack::io\_jack\_t::connections\_in [private]

**5.261.4.6 delays\_in** `MHAParser::vint_mon_t MHAIOJack::io_jack_t::delays_in` [private]

**5.261.4.7 connections\_out** `MHAParser::vstring_t MHAIOJack::io_jack_t::connections_out` [private]

**5.261.4.8 delays\_out** `MHAParser::vint_mon_t MHAIOJack::io_jack_t::delays_out` [private]

**5.261.4.9 portnames\_in** `MHAParser::vstring_t MHAIOJack::io_jack_t::portnames_in` [private]

**5.261.4.10 portnames\_out** `MHAParser::vstring_t MHAIOJack::io_jack_t::portnames_out` [private]

**5.261.4.11 ports\_in\_physical** `MHAParser::vstring_mon_t MHAIOJack::io_jack_t::ports_in_physical` [private]

**5.261.4.12 ports\_out\_physical** `MHAParser::vstring_mon_t MHAIOJack::io_jack_t::ports_out_physical` [private]

**5.261.4.13 ports\_in\_all** `MHAParser::vstring_mon_t MHAIOJack::io_jack_t::ports_in_all` [private]

**5.261.4.14 ports\_out\_all** `MHAParser::vstring_mon_t` MHAIOJack::io\_jack\_t::ports\_out\_all [private]

**5.261.4.15 ports\_parser** `MHAParser::parser_t` MHAIOJack::io\_jack\_t::ports\_parser [private]

**5.261.4.16 state\_cupload** `MHAParser::float_mon_t` MHAIOJack::io\_jack\_t::state\_cupload [private]

**5.261.4.17 state\_xruns** `MHAParser::int_mon_t` MHAIOJack::io\_jack\_t::state\_xruns [private]

**5.261.4.18 state\_priority** `MHAParser::int_mon_t` MHAIOJack::io\_jack\_t::state\_priority [private]

**5.261.4.19 state\_scheduler** `MHAParser::string_mon_t` MHAIOJack::io\_jack\_t::state\_scheduler [private]

**5.261.4.20 state\_parser** `MHAParser::parser_t` MHAIOJack::io\_jack\_t::state\_parser [private]

**5.261.4.21 patchbay** `MHAEVENTS::patchbay_t< io_jack_t>` MHAIOJack::io\_jack\_t::patchbay [private]

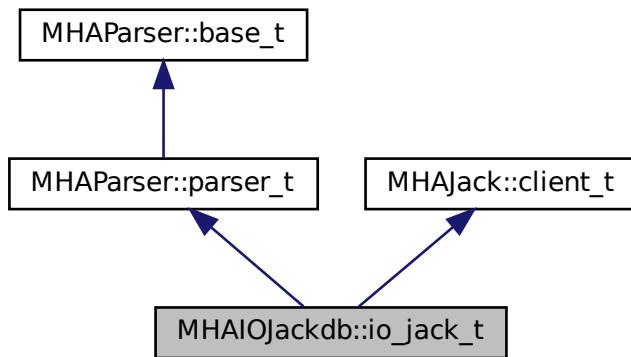
The documentation for this class was generated from the following file:

- **MHAIOJack.cpp**

## 5.262 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

### 5.262.1 Detailed Description

Main class for JACK IO.

This class registers a JACK client. JACK and framework states are managed by this class.

## 5.262.2 Constructor & Destructor Documentation

```
5.262.2.1 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 )
```

## 5.262.3 Member Function Documentation

```
5.262.3.1 prepare() void io_jack_t::prepare (
    int nch_in,
    int nch_out )
```

Allocate buffers, activate JACK client and install internal ports.

```
5.262.3.2 release() void io_jack_t::release ( )
```

```
5.262.3.3 fail_on_async_jackerror() bool MHAIOJackdb::io_jack_t::fail_on_async_←
jackerror ( ) const [inline]
```

```
5.262.3.4 IOProcessEvent_inner() [1/2] int io_jack_t::IOProcessEvent_inner (
```

```
    void * handle,
    mha_wave_t * sIn,
    mha_wave_t ** sOut ) [static], [private]
```

**5.262.3.5 IOProcessEvent\_inner() [2/2]**

```
int io_jack_t::IOProcessEvent_inner (
    mha_wave_t * sIn,
    mha_wave_t ** sOut ) [private]
```

**5.262.3.6 reconnect\_inports()**

```
void io_jack_t::reconnect_inports () [private]
```

Connect the input ports when connection variable is accessed.

**5.262.3.7 reconnect\_outports()**

```
void io_jack_t::reconnect_outports () [private]
```

Connect the output ports when connection variable is accessed.

**5.262.3.8 get\_physical\_input\_ports()**

```
void io_jack_t::get_physical_input_ports () [private]
```

**5.262.3.9 get\_physical\_output\_ports()**

```
void io_jack_t::get_physical_output_ports () [private]
```

**5.262.3.10 get\_all\_input\_ports()**

```
void io_jack_t::get_all_input_ports () [private]
```

**5.262.3.11 get\_all\_output\_ports()**

```
void io_jack_t::get_all_output_ports () [private]
```

**5.262.3.12 read\_get\_cpu\_load()**

```
void io_jack_t::read_get_cpu_load () [private]
```

**5.262.3.13 `read_get_xruns()`** void io\_jack\_t::read\_get\_xruns ( ) [private]

**5.262.3.14 `read_get_scheduler()`** void io\_jack\_t::read\_get\_scheduler ( ) [private]

**5.262.3.15 `set_use_jack_transport()`** void io\_jack\_t::set\_use\_jack\_transport ( ) [private]

**5.262.3.16 `set_locate()`** void io\_jack\_t::set\_locate ( ) [private]

## 5.262.4 Member Data Documentation

**5.262.4.1 `proc_event`** IOProcessEvent\_t MHAIOJackdb::io\_jack\_t::proc\_event [private]

**5.262.4.2 `proc_handle`** void\* MHAIOJackdb::io\_jack\_t::proc\_handle [private]

**5.262.4.3 `mha_fragsize`** unsigned int MHAIOJackdb::io\_jack\_t::mha\_fragsize [private]

**5.262.4.4 `mha_samplerate`** float MHAIOJackdb::io\_jack\_t::mha\_samplerate [private]

**5.262.4.5 fragsize\_ratio** `unsigned int MHAIOJackdb::io_jack_t::fragsize_ratio [private]`

**5.262.4.6 servername** `MHAParser::string_t MHAIOJackdb::io_jack_t::servername [private]`

**5.262.4.7 clientname** `MHAParser::string_t MHAIOJackdb::io_jack_t::clientname [private]`

**5.262.4.8 connections\_in** `MHAParser::vstring_t MHAIOJackdb::io_jack_t::connections_in [private]`

**5.262.4.9 connections\_out** `MHAParser::vstring_t MHAIOJackdb::io_jack_t::connections_out [private]`

**5.262.4.10 portnames\_in** `MHAParser::vstring_t MHAIOJackdb::io_jack_t::portnames_in [private]`

**5.262.4.11 portnames\_out** `MHAParser::vstring_t MHAIOJackdb::io_jack_t::portnames_out [private]`

**5.262.4.12 fail\_on\_async\_jackerr** `MHAParser::bool_t MHAIOJackdb::io_jack_t::fail_on_async_jackerr [private]`

**5.262.4.13 use\_jack\_transport** `MHAParser::bool_t MHAIOJackdb::io_jack_t::use_←  
jack_transport [private]`

**5.262.4.14 locate** `MHAParser::float_t MHAIOJackdb::io_jack_t::locate [private]`

**5.262.4.15 server\_srate** `MHAParser::float_mon_t MHAIOJackdb::io_jack_t::server_←  
srate [private]`

**5.262.4.16 server\_fragsize** `MHAParser::int_mon_t MHAIOJackdb::io_jack_t::server_←  
fragsize [private]`

**5.262.4.17 ports\_in\_physical** `MHAParser::vstring_mon_t MHAIOJackdb::io_jack_t←  
::ports_in_physical [private]`

**5.262.4.18 ports\_out\_physical** `MHAParser::vstring_mon_t MHAIOJackdb::io_jack_t←  
::ports_out_physical [private]`

**5.262.4.19 ports\_in\_all** `MHAParser::vstring_mon_t MHAIOJackdb::io_jack_t::ports_←  
in_all [private]`

**5.262.4.20 ports\_out\_all** `MHAParser::vstring_mon_t MHAIOJackdb::io_jack_t::ports_←  
out_all [private]`

**5.262.4.21 ports\_parser** `MHAParser::parser_t` MHAIOJackdb::io\_jack\_t::ports\_parser  
[private]

**5.262.4.22 state\_cupload** `MHAParser::float_mon_t` MHAIOJackdb::io\_jack\_t::state\_cupload  
[private]

**5.262.4.23 state\_xruns** `MHAParser::int_mon_t` MHAIOJackdb::io\_jack\_t::state\_xruns  
[private]

**5.262.4.24 state\_priority** `MHAParser::int_mon_t` MHAIOJackdb::io\_jack\_t::state\_priority  
[private]

**5.262.4.25 state\_scheduler** `MHAParser::string_mon_t` MHAIOJackdb::io\_jack\_t::state\_scheduler  
[private]

**5.262.4.26 state\_parser** `MHAParser::parser_t` MHAIOJackdb::io\_jack\_t::state\_parser  
[private]

**5.262.4.27 pwinner\_out** `MHASignal::waveform_t*` MHAIOJackdb::io\_jack\_t::pwinner\_out  
[private]

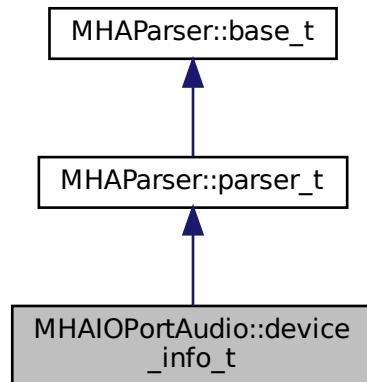
**5.262.4.28 patchbay** `MHAEVENTS::patchbay_t< io_jack_t>` MHAIOJackdb::io\_jack\_t::patchbay  
[private]

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

- **MHAIOJackdb.cpp**

## 5.263 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

#### 5.263.1 Constructor & Destructor Documentation

**5.263.1.1 device\_info\_t()** MHAIOPortAudio::device\_info\_t::device\_info\_t ( ) [inline]

## 5.263.2 Member Function Documentation

**5.263.2.1 fill\_info()** void MHAIOPortAudio::device\_info\_t::fill\_info ( ) [inline]

## 5.263.3 Member Data Documentation

**5.263.3.1 numDevices** MHAParser::int\_mon\_t MHAIOPortAudio::device\_info\_t::numDevices

**5.263.3.2 structVersion** MHAParser::vint\_mon\_t MHAIOPortAudio::device\_info\_t::structVersion

**5.263.3.3 name** MHAParser::vstring\_mon\_t MHAIOPortAudio::device\_info\_t::name

**5.263.3.4 hostApi** MHAParser::vint\_mon\_t MHAIOPortAudio::device\_info\_t::hostApi

**5.263.3.5 maxInputChannels** MHAParser::vint\_mon\_t MHAIOPortAudio::device\_info\_t::maxInputChannels

**5.263.3.6 maxOutputChannels** `MHAParser::vint_mon_t MHAIOPortAudio::device_info_t::maxOutputChannels`

**5.263.3.7 defaultLowInputLatency** `MHAParser::vfloat_mon_t MHAIOPortAudio::device_info_t::defaultLowInputLatency`

**5.263.3.8 defaultLowOutputLatency** `MHAParser::vfloat_mon_t MHAIOPortAudio::device_info_t::defaultLowOutputLatency`

**5.263.3.9 defaultHighInputLatency** `MHAParser::vfloat_mon_t MHAIOPortAudio::device_info_t::defaultHighInputLatency`

**5.263.3.10 defaultHighOutputLatency** `MHAParser::vfloat_mon_t MHAIOPortAudio::device_info_t::defaultHighOutputLatency`

**5.263.3.11 defaultSampleRate** `MHAParser::vfloat_mon_t MHAIOPortAudio::device_info_t::defaultSampleRate`

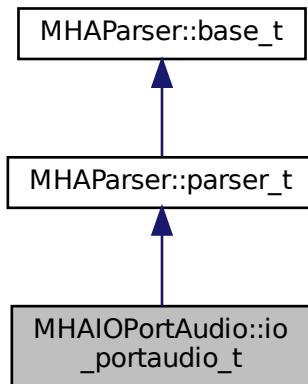
The documentation for this class was generated from the following file:

- **MHAIOPortAudio.cpp**

## 5.264 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`
- `stream_info_t stream_info`
- `MHASignal::waveform_t * s_in`
- `mha_wave_t * s_out`
- `float samplerate`
- `unsigned int nchannels_out`
- `unsigned int nchannels_in`
- `unsigned int fragsize`
- `IOProcessEvent_t proc_event`
- `void * proc_handle`
- `IOStartedEvent_t start_event`
- `void * start_handle`
- `IOStoppedEvent_t stop_event`
- `void * stop_handle`
- `PaStream * portaudio_stream`
- `MHAParser::string_t device_name_in`
- `MHAParser::int_t device_index_in`
- `MHAParser::string_t device_name_out`
- `MHAParser::int_t device_index_out`
- `MHAParser::float_t suggestedInputLatency`
- `MHAParser::float_t suggestedOutputLatency`
- `MHAEvents::patchbay_t< io_portaudio_t > patchbay`

## Additional Inherited Members

### 5.264.1 Detailed Description

Main class for Portaudio sound IO.

### 5.264.2 Constructor & Destructor Documentation

#### 5.264.2.1 `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]
```

**5.264.2.2 ~io\_portaudio\_t()** MHAIOPortAudio::io\_portaudio\_t::~io\_portaudio\_t ( )  
[inline]

### 5.264.3 Member Function Documentation

**5.264.3.1 device\_name\_in\_updated()** void MHAIOPortAudio::io\_portaudio\_t::device\_name\_in\_updated ( ) [inline]

**5.264.3.2 device\_name\_out\_updated()** void MHAIOPortAudio::io\_portaudio\_t::device\_name\_out\_updated ( ) [inline]

**5.264.3.3 device\_index\_in\_updated()** void MHAIOPortAudio::io\_portaudio\_t::device\_index\_in\_updated ( ) [inline]

**5.264.3.4 device\_index\_out\_updated()** void MHAIOPortAudio::io\_portaudio\_t::device\_index\_out\_updated ( ) [inline]

**5.264.3.5 cmd\_prepare()** void MHAIOPortAudio::io\_portaudio\_t::cmd\_prepare ( int *nchannels\_in*, int *nchannels\_out* )

**5.264.3.6 cmd\_start()** void MHAIOPortAudio::io\_portaudio\_t::cmd\_start ( )

**5.264.3.7 cmd\_stop()** void MHAIOPortAudio::io\_portaudio\_t::cmd\_stop ( )

**5.264.3.8 cmd\_release()** void MHAIOPortAudio::io\_portaudio\_t::cmd\_release ( )

**5.264.3.9 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* )

## 5.264.4 Member Data Documentation

**5.264.4.1 device\_info** **device\_info\_t** MHAIOPortAudio::io\_portaudio\_t::device\_info [private]

**5.264.4.2 stream\_info** **stream\_info\_t** MHAIOPortAudio::io\_portaudio\_t::stream\_info [private]

**5.264.4.3 s\_in** **MHASignal::waveform\_t\*** MHAIOPortAudio::io\_portaudio\_t::s\_in [private]

**5.264.4.4 s\_out** **mha\_wave\_t\*** MHAIOPortAudio::io\_portaudio\_t::s\_out [private]

**5.264.4.5 samplerate** float MHAIOPortAudio::io\_portaudio\_t::samplerate [private]

**5.264.4.6 nchannels\_out** unsigned int MHAIOPortAudio::io\_portaudio\_t::nchannels\_out [private]

**5.264.4.7 nchannels\_in** unsigned int MHAIOPortAudio::io\_portaudio\_t::nchannels\_in [private]

**5.264.4.8 fragsize** unsigned int MHAIOPortAudio::io\_portaudio\_t::fragsize [private]

**5.264.4.9 proc\_event** IOProcessEvent\_t MHAIOPortAudio::io\_portaudio\_t::proc\_event [private]

**5.264.4.10 proc\_handle** void\* MHAIOPortAudio::io\_portaudio\_t::proc\_handle [private]

**5.264.4.11 start\_event** IOStartedEvent\_t MHAIOPortAudio::io\_portaudio\_t::start\_event [private]

**5.264.4.12 start\_handle** void\* MHAIOPortAudio::io\_portaudio\_t::start\_handle [private]

**5.264.4.13 stop\_event** IOStoppedEvent\_t MHAIOPortAudio::io\_portaudio\_t::stop\_event [private]

**5.264.4.14 stop\_handle** void\* MHAIOPortAudio::io\_portaudio\_t::stop\_handle [private]

**5.264.4.15 portaudio\_stream** PaStream\* MHAIOPortAudio::io\_portaudio\_t::portaudio\_stream [private]

**5.264.4.16 device\_name\_in** MHAParser::string\_t MHAIOPortAudio::io\_portaudio\_t::device\_name\_in [private]

**5.264.4.17 device\_index\_in** MHAParser::int\_t MHAIOPortAudio::io\_portaudio\_t::device\_index\_in [private]

**5.264.4.18 device\_name\_out** MHAParser::string\_t MHAIOPortAudio::io\_portaudio\_t::device\_name\_out [private]

**5.264.4.19 device\_index\_out** MHAParser::int\_t MHAIOPortAudio::io\_portaudio\_t::device\_index\_out [private]

**5.264.4.20 suggestedInputLatency** MHAParser::float\_t MHAIOPortAudio::io\_portaudio\_t::suggestedInputLatency [private]

**5.264.4.21 suggestedOutputLatency** MHAParser::float\_t MHAIOPortAudio::io\_portaudio\_t::suggestedOutputLatency [private]

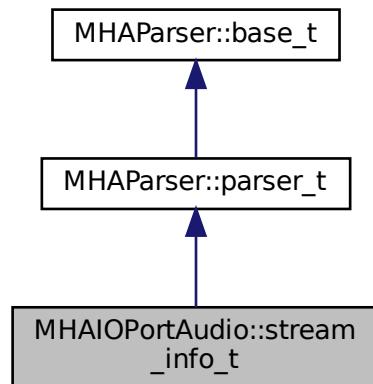
**5.264.4.22 patchbay** `MHAEVENTS::patchbay_t< io_portaudio_t> MHAIOPortAudio::io_<→`  
`portaudio_t::patchbay [private]`

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

- **MHAIOPortAudio.cpp**

## 5.265 MHAIOPortAudio::stream\_info\_t Class Reference

Inheritance diagram for MHAIOPortAudio::stream\_info\_t:



### Public Member Functions

- `stream_info_t()`
- `void fill_info (PaStream *stream_)`

### Public Attributes

- `MHAParser::float_mon_t palInputLatency`
- `MHAParser::float_mon_t paOutputLatency`
- `MHAParser::float_mon_t paSampleRate`

### Additional Inherited Members

#### 5.265.1 Constructor & Destructor Documentation

**5.265.1.1 `stream_info_t()`** `MHAIOPortAudio::stream_info_t::stream_info_t ( ) [inline]`

## 5.265.2 Member Function Documentation

**5.265.2.1 `fill_info()`** `void MHAIOPortAudio::stream_info_t::fill_info ( PaStream * stream_ ) [inline]`

## 5.265.3 Member Data Documentation

**5.265.3.1 `paInputLatency`** `MHAParser::float_mon_t MHAIOPortAudio::stream_info_t::paInputLatency`

**5.265.3.2 `paOutputLatency`** `MHAParser::float_mon_t MHAIOPortAudio::stream_info_t::paOutputLatency`

**5.265.3.3 `paSampleRate`** `MHAParser::float_mon_t MHAIOPortAudio::stream_info_t::paSampleRate`

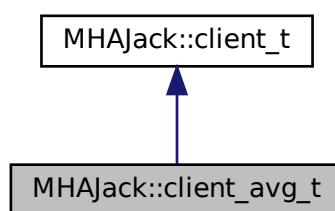
The documentation for this class was generated from the following file:

- **MHAIOPortAudio.cpp**

## 5.266 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

### 5.266.1 Detailed Description

Generic JACK client for averaging a system response across time.

### 5.266.2 Constructor & Destructor Documentation

**5.266.2.1 `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

### 5.266.3 Member Function Documentation

```
5.266.3.1 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

```
5.266.3.2 proc() [1/2] int MHAJack::client_avg_t::proc (
    void * handle,
    mha_wave_t * sIn,
    mha_wave_t ** sOut ) [static], [private]
```

```
5.266.3.3 IOStoppedEvent() [1/2] void MHAJack::client_avg_t::IOStoppedEvent (
    void * handle,
    int proc_err,
    int io_err ) [static], [private]
```

```
5.266.3.4 proc() [2/2] void MHAJack::client_avg_t::proc (
    mha_wave_t * sIn,
    mha_wave_t ** sOut ) [private]
```

**5.266.3.5 IOStoppedEvent() [2/2]** void MHAJack::client\_avg\_t::IOStoppedEvent ( )  
[private]

## 5.266.4 Member Data Documentation

**5.266.4.1 b\_stopped** bool MHAJack::client\_avg\_t::b\_stopped [private]

**5.266.4.2 pos** unsigned int MHAJack::client\_avg\_t::pos [private]

**5.266.4.3 sn\_in** mha\_wave\_t\* MHAJack::client\_avg\_t::sn\_in [private]

**5.266.4.4 sn\_out** mha\_wave\_t\* MHAJack::client\_avg\_t::sn\_out [private]

**5.266.4.5 name** std::string MHAJack::client\_avg\_t::name [private]

**5.266.4.6 frag\_out** MHASignal::waveform\_t\* MHAJack::client\_avg\_t::frag\_out [private]

**5.266.4.7 nrep** const unsigned int MHAJack::client\_avg\_t::nrep [private]

**5.266.4.8 n** unsigned int MHAJack::client\_avg\_t::n [private]

**5.266.4.9 b\_ready** bool MHAJack::client\_avg\_t::b\_ready [private]

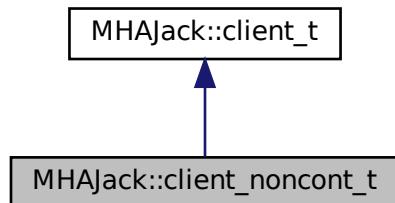
The documentation for this class was generated from the following files:

- **mhajack.h**
- **mhajack.cpp**

## 5.267 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

### 5.267.1 Detailed Description

Generic client for synchronous playback and recording of waveform fragments.

### 5.267.2 Constructor & Destructor Documentation

#### 5.267.2.1 **client\_noncont\_t()** MHAJack::client\_noncont\_t::client\_noncont\_t (

```
const std::string & name,
bool use_jack_transport = false )
```

### 5.267.3 Member Function Documentation

#### 5.267.3.1 **io()** void MHAJack::client\_noncont\_t::io (

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

**5.267.3.2 proc() [1/2]** int MHAJack::client\_noncont\_t::proc ( void \* *handle*,  
                  mha\_wave\_t \* *sIn*,  
                  mha\_wave\_t \*\* *sOut* ) [static], [private]

**5.267.3.3 IOStoppedEvent() [1/2]** void MHAJack::client\_noncont\_t::IOStoppedEvent ( void \* *handle*,  
                  int *proc\_err*,  
                  int *io\_err* ) [static], [private]

**5.267.3.4 proc() [2/2]** void MHAJack::client\_noncont\_t::proc ( mha\_wave\_t \* *sIn*,  
                  mha\_wave\_t \*\* *sOut* ) [private]

**5.267.3.5 IOStoppedEvent() [2/2]** void MHAJack::client\_noncont\_t::IOStoppedEvent ( ) [private]

## 5.267.4 Member Data Documentation

**5.267.4.1 b\_stopped** bool MHAJack::client\_noncont\_t::b\_stopped [private]

**5.267.4.2 pos** unsigned int MHAJack::client\_noncont\_t::pos [private]

**5.267.4.3 sn\_in** mha\_wave\_t\* MHAJack::client\_noncont\_t::sn\_in [private]

**5.267.4.4 sn\_out** `mha_wave_t*` `MHAJack::client_noncont_t::sn_out` [private]

**5.267.4.5 name** `std::string` `MHAJack::client_noncont_t::name` [private]

**5.267.4.6 frag\_out** `MHASignal::waveform_t*` `MHAJack::client_noncont_t::frag_out` [private]

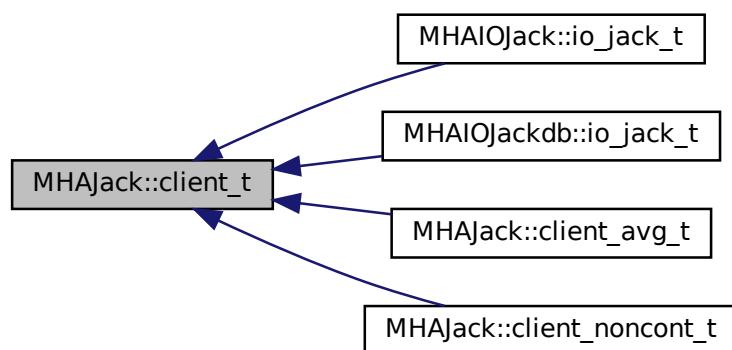
The documentation for this class was generated from the following files:

- `mhajack.h`
- `mhajack.cpp`

## 5.268 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**

### 5.268.1 Detailed Description

Generic asynchronous JACK client.

### 5.268.2 Constructor & Destructor Documentation

```
5.268.2.1 client_t() MHAJack::client_t::client_t (
    IOProcessEvent_t proc_event,
    void * proc_handle = NULL,
    IOStartedEvent_t start_event = NULL,
    void * start_handle = NULL,
    IOStoppedEvent_t stop_event = NULL,
    void * stop_handle = NULL,
    bool use_jack_transport = false )
```

### 5.268.3 Member Function Documentation

**5.268.3.1 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

**5.268.3.2 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

**5.268.3.3 `release()`** `void MHAJack::client_t::release ( )`

Remove JACK client and deallocate internal ports and buffers.

**5.268.3.4 `start()`** `void MHAJack::client_t::start (`  
`bool fail_on_async_jack_error = true )`**5.268.3.5 `stop()`** `void MHAJack::client_t::stop ( )`**5.268.3.6 `connect_input()`** `void MHAJack::client_t::connect_input (`  
`const std::vector< std::string > & con )`

Connect the input ports when connection variable is accessed.

**5.268.3.7 `connect_output()`** `void MHAJack::client_t::connect_output (`  
`const std::vector< std::string > & con )`

Connect the output ports when connection variable is accessed.

**5.268.3.8 `get_fragsize()`** `unsigned int MHAJack::client_t::get_fragsize ( ) const`  
[inline]**5.268.3.9 `get_srate()`** `float MHAJack::client_t::get_srate ( ) const [inline]`

**5.268.3.10 get\_xruns()** unsigned long MHAJack::client\_t::get\_xruns ( ) [inline]

**5.268.3.11 get\_xruns\_reset()** unsigned long MHAJack::client\_t::get\_xruns\_reset ( )

**5.268.3.12 str\_error()** std::string MHAJack::client\_t::str\_error ( int err )

**5.268.3.13 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.)

**5.268.3.14 get\_my\_input\_ports()** std::vector< std::string > MHAJack::client\_t::get\_my\_input\_ports ( )

**5.268.3.15 get\_my\_output\_ports()** std::vector< std::string > MHAJack::client\_t::get\_my\_output\_ports ( )

**5.268.3.16 set\_input\_portnames()** void MHAJack::client\_t::set\_input\_portnames ( const std::vector< std::string > & names )

---

**5.268.3.17 set\_output\_portnames()** void MHAJack::client\_t::set\_output\_portnames ( const std::vector< std::string > & names )

**5.268.3.18 get\_cpu\_load()** float MHAJack::client\_t::get\_cpu\_load ( )

**5.268.3.19 set\_use\_jack\_transport()** void MHAJack::client\_t::set\_use\_jack\_transport ( bool ut ) [inline]

**5.268.3.20 is\_prepared()** bool MHAJack::client\_t::is\_prepared ( ) const [inline]

**5.268.3.21 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

**5.268.3.22 internal\_start()** void MHAJack::client\_t::internal\_start ( ) [private]

**5.268.3.23 internal\_stop()** void MHAJack::client\_t::internal\_stop ( ) [private]

**5.268.3.24 stopped()** void MHAJack::client\_t::stopped ( int *proc\_err*, int *io\_err* ) [private]

**5.268.3.25 jack\_proc\_cb() [1/2]** int MHAJack::client\_t::jack\_proc\_cb ( jack\_nframes\_t *n*, void \* *h* ) [static], [private]

**5.268.3.26 jack\_proc\_cb() [2/2]** int MHAJack::client\_t::jack\_proc\_cb ( jack\_nframes\_t *n* ) [private]

This is the main processing callback.

Here happens double buffering and downsampling.

**5.268.3.27 jack\_xrun\_cb() [1/2]** int MHAJack::client\_t::jack\_xrun\_cb ( void \* *h* ) [static], [private]

**5.268.3.28 jack\_xrun\_cb() [2/2]** int MHAJack::client\_t::jack\_xrun\_cb ( ) [inline], [private]

## 5.268.4 Member Data Documentation

**5.268.4.1 num\_xruns** unsigned long MHAJack::client\_t::num\_xruns [private]

**5.268.4.2 fragsize** unsigned int MHAJack::client\_t::fragsize [private]

**5.268.4.3 samplerate** float MHAJack::client\_t::samplerate [private]

**5.268.4.4 nchannels\_in** unsigned int MHAJack::client\_t::nchannels\_in [private]

**5.268.4.5 nchannels\_out** unsigned int MHAJack::client\_t::nchannels\_out [private]

**5.268.4.6 proc\_event** IOProcessEvent\_t MHAJack::client\_t::proc\_event [private]

**5.268.4.7 proc\_handle** void\* MHAJack::client\_t::proc\_handle [private]

**5.268.4.8 start\_event** IOStartedEvent\_t MHAJack::client\_t::start\_event [private]

**5.268.4.9 start\_handle** void\* MHAJack::client\_t::start\_handle [private]

**5.268.4.10 stop\_event** IOStoppedEvent\_t MHAJack::client\_t::stop\_event [private]

**5.268.4.11 stop\_handle** void\* MHAJack::client\_t::stop\_handle [private]

**5.268.4.12 s\_in** MHASignal::waveform\_t\* MHAJack::client\_t::s\_in [private]

**5.268.4.13 s\_out** mha\_wave\_t\* MHAJack::client\_t::s\_out [private]

**5.268.4.14 inch** MHAJack::port\_t\*\* MHAJack::client\_t::inch [private]

**5.268.4.15 outch** MHAJack::port\_t\*\* MHAJack::client\_t::outch [private]

**5.268.4.16 jc** jack\_client\_t\* MHAJack::client\_t::jc [protected]

**5.268.4.17 flags** unsigned int MHAJack::client\_t::flags [private]

**5.268.4.18 b\_prepared** bool MHAJack::client\_t::b\_prepared [private]

**5.268.4.19 use\_jack\_transport** bool MHAJack::client\_t::use\_jack\_transport [private]

**5.268.4.20 jstate\_prev** jack\_transport\_state\_t MHAJack::client\_t::jstate\_prev [private]

**5.268.4.21 input\_portnames** std::vector<std::string> MHAJack::client\_t::input\_portnames [private]

**5.268.4.22 output\_portnames** std::vector<std::string> MHAJack::client\_t::output\_portnames [private]

**5.268.4.23 fail\_on\_async\_jackerror** bool MHAJack::client\_t::fail\_on\_async\_jackerror [private]

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

- **mhajack.h**
- **mhajack.cpp**

## 5.269 MHAJack::port\_t Class Reference

Class for one channel/port.

### Public Types

- enum **dir\_t** { **input**, **output** }

### Public Member Functions

- **port\_t** (jack\_client\_t \* **jc**, **dir\_t** **dir**, int **id**)
- **port\_t** (jack\_client\_t \* **jc**, **dir\_t** **dir**, const std::string &**id**)  
*Constructor to create port with specific name.*
- **~port\_t** ()
- void **read** ( **mha\_wave\_t** \***s**, unsigned int **ch** )
- void **write** ( **mha\_wave\_t** \***s**, unsigned int **ch** )
- void **mute** (unsigned int **n**)
- void **connect\_to** (const char \***pn**)
- const char \* **get\_short\_name** ()  
*Return the port name.*

## Private Attributes

- **dir\_t dir\_type**
- **jack\_port\_t \* port**
- **jack\_default\_audio\_sample\_t \* iob**
- **jack\_client\_t \* jc**

### 5.269.1 Detailed Description

Class for one channel/port.

This class represents one JACK port. Double buffering for asynchronous process callbacks is managed by this class.

### 5.269.2 Member Enumeration Documentation

#### 5.269.2.1 **dir\_t** enum MHAJack::port\_t::dir\_t

Enumerator

input	
output	

### 5.269.3 Constructor & Destructor Documentation

#### 5.269.3.1 **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).

**5.269.3.2 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.

**5.269.3.3 ~port\_t()** MHAJack::port\_t::~port\_t ( )

### 5.269.4 Member Function Documentation

**5.269.4.1 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.

**5.269.4.2 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.

**5.269.4.3 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).
----------	---

**5.269.4.4 connect\_to()** void MHAJack::port\_t::connect\_to ( const char \* *pn* )

**Parameters**

<i>pn</i>	Port name to connect to
-----------	-------------------------

**5.269.4.5 get\_short\_name()** const char \* MHAJack::port\_t::get\_short\_name ( )

Return the port name.

## 5.269.5 Member Data Documentation

**5.269.5.1 dir\_type** *dir\_t* MHAJack::port\_t::dir\_type [private]

**5.269.5.2 port** jack\_port\_t\* MHAJack::port\_t::port [private]

**5.269.5.3 iob** jack\_default\_audio\_sample\_t\* MHAJack::port\_t::iob [private]

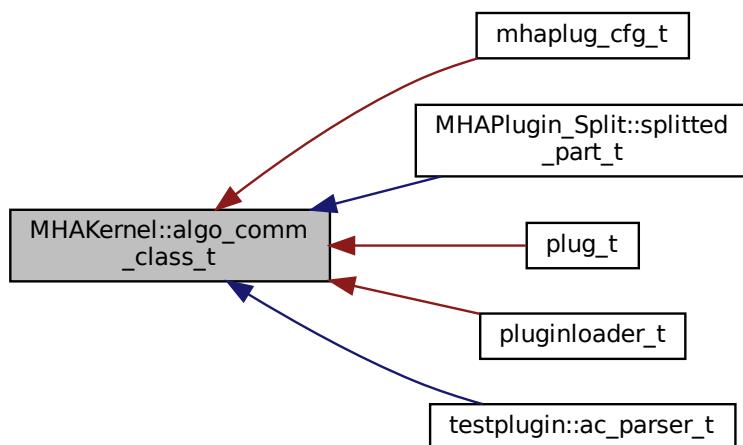
**5.269.5.4 jc** jack\_client\_t\* MHAJack::port\_t::jc [private]

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

- **mhajack.h**
- **mhajack.cpp**

## 5.270 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`

### 5.270.1 Constructor & Destructor Documentation

**5.270.1.1 algo\_comm\_class\_t()** MHAKernel::algo\_comm\_class\_t::algo\_comm\_class\_t ( )

**5.270.1.2 ~algo\_comm\_class\_t()** MHAKernel::algo\_comm\_class\_t::~algo\_comm\_class\_t ( ) [virtual]

## 5.270.2 Member Function Documentation

**5.270.2.1 get\_c\_handle()** algo\_comm\_t MHAKernel::algo\_comm\_class\_t::get\_c\_handle ( )

**5.270.2.2 insert\_var()** int MHAKernel::algo\_comm\_class\_t::insert\_var ( void \* handle,  
const char \* name,  
**comm\_var\_t** var ) [static]

**5.270.2.3 insert\_var\_int()** int MHAKernel::algo\_comm\_class\_t::insert\_var\_int ( void \* handle,  
const char \* name,  
int \* ivar ) [static]

**5.270.2.4 insert\_var\_vfloat()** int MHAKernel::algo\_comm\_class\_t::insert\_var\_vfloat ( void \* handle,  
const char \* name,  
std::vector< float > & ivar ) [static]

**5.270.2.5 insert\_var\_float()** int MHAKernel::algo\_comm\_class\_t::insert\_var\_float ( void \* *handle*, const char \* *name*, float \* *ivar* ) [static]

**5.270.2.6 insert\_var\_double()** int MHAKernel::algo\_comm\_class\_t::insert\_var\_double ( void \* *handle*, const char \* *name*, double \* *ivar* ) [static]

**5.270.2.7 remove\_var()** int MHAKernel::algo\_comm\_class\_t::remove\_var ( void \* *handle*, const char \* *name* ) [static]

**5.270.2.8 remove\_ref()** int MHAKernel::algo\_comm\_class\_t::remove\_ref ( void \* *handle*, void \* *ref* ) [static]

**5.270.2.9 is\_var()** int MHAKernel::algo\_comm\_class\_t::is\_var ( void \* *handle*, const char \* *name* ) [static]

**5.270.2.10 get\_var()** int MHAKernel::algo\_comm\_class\_t::get\_var ( void \* *handle*, const char \* *name*, comm\_var\_t \* *var* ) [static]

**5.270.2.11 `get_var_int()`** `int MHAKernel::algo_comm_class_t::get_var_int (`  
`void * handle,`  
`const char * name,`  
`int * ivar ) [static]`

**5.270.2.12 `get_var_float()`** `int MHAKernel::algo_comm_class_t::get_var_float (`  
`void * handle,`  
`const char * name,`  
`float * ivar ) [static]`

**5.270.2.13 `get_var_double()`** `int MHAKernel::algo_comm_class_t::get_var_double (`  
`void * handle,`  
`const char * name,`  
`double * ivar ) [static]`

**5.270.2.14 `get_entries()`** `int MHAKernel::algo_comm_class_t::get_entries (`  
`void * handle,`  
`char * ret,`  
`unsigned int len ) [static]`

**5.270.2.15 `get_error()`** `const char * MHAKernel::algo_comm_class_t::get_error (`  
`int e ) [static]`

**5.270.2.16 `local_insert_var()`** `void MHAKernel::algo_comm_class_t::local_insert_var (`  
`const char * name,`  
`comm_var_t var ) [virtual]`

**5.270.2.17 `local_remove_var()`** `void MHAKernel::algo_comm_class_t::local_remove_var (`  
`const char * name ) [virtual]`

**5.270.2.18 local\_remove\_ref()** void MHAKernel::algo\_comm\_class\_t::local\_remove\_ref ( void \* *addr* ) [virtual]

**5.270.2.19 local\_is\_var()** bool MHAKernel::algo\_comm\_class\_t::local\_is\_var ( const char \* *name* ) [virtual]

**5.270.2.20 local\_get\_var()** void MHAKernel::algo\_comm\_class\_t::local\_get\_var ( const char \* *name*, comm\_var\_t \* *var* ) [virtual]

**5.270.2.21 local\_get\_entries()** std::string MHAKernel::algo\_comm\_class\_t::local\_get\_entries ( ) [virtual]

**5.270.2.22 size()** MHAKernel::comm\_var\_map\_t::size\_type MHAKernel::algo\_comm\_class\_t::size ( ) const [virtual]

### 5.270.3 Member Data Documentation

**5.270.3.1 algo\_comm\_id\_string** char\* MHAKernel::algo\_comm\_class\_t::algo\_comm\_id\_string

**5.270.3.2 ac** algo\_comm\_t MHAKernel::algo\_comm\_class\_t::ac [private]

**5.270.3.3 algo\_comm\_id\_string\_len** int MHAKernel::algo\_comm\_class\_t::algo\_comm\_id\_string\_len [private]

**5.270.3.4 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**

## 5.271 MHAKernel::comm\_var\_map\_t Class Reference

Inherits map< std::string, comm\_var\_t >.

### Public Member Functions

- bool **has\_key** (const std::string &name)

#### 5.271.1 Member Function Documentation

**5.271.1.1 has\_key()** bool MHAKernel::comm\_var\_map\_t::has\_key (const std::string & name ) [inline]

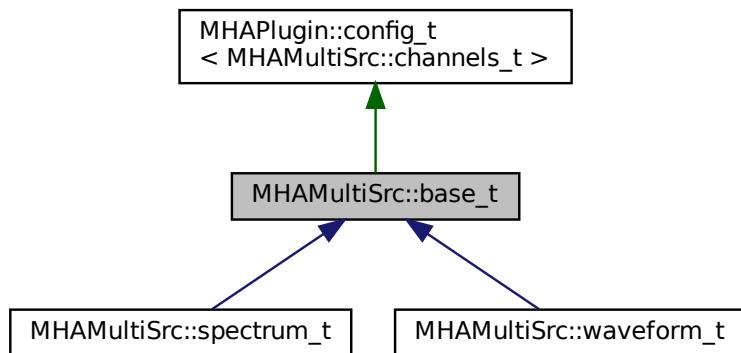
The documentation for this class was generated from the following file:

- **mha\_algo\_comm.hh**

## 5.272 MHAMultiSrc::base\_t Class Reference

Base class for source selection.

Inheritance diagram for MHAMultiSrc::base\_t:



### Public Member Functions

- **base\_t ( algo\_comm\_t iac)**
- void **select\_source** (const std::vector< std::string > &src, int in\_channels)  
*Change the selection of input sources.*

### Protected Attributes

- **algo\_comm\_t ac**

### Additional Inherited Members

#### 5.272.1 Detailed Description

Base class for source selection.

See also

- [MHAMultiSrc::channel\\_t \(p. 993\)](#)**
- [MHAMultiSrc::channels\\_t \(p. 993\)](#)**

## 5.272.2 Constructor & Destructor Documentation

**5.272.2.1 `base_t()`** `MHAMultiSrc::base_t::base_t ( algo_comm_t iac )`

## 5.272.3 Member Function Documentation

**5.272.3.1 `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)

## 5.272.4 Member Data Documentation

**5.272.4.1 `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`

## 5.273 MHAMultiSrc::channel\_t Class Reference

### Public Attributes

- std::string **name**
- int **channel**

#### 5.273.1 Member Data Documentation

##### 5.273.1.1 **name** std::string MHAMultiSrc::channel\_t::name

##### 5.273.1.2 **channel** int MHAMultiSrc::channel\_t::channel

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

- **mha\_multisrc.h**

## 5.274 MHAMultiSrc::channels\_t Class Reference

Inherits vector< MHAMultiSrc::channel\_t >.

### Public Member Functions

- **channels\_t** (const std::vector< std::string > &src, int in\_channels)  
*Separate a list of input sources into a parsable channel list.*

#### 5.274.1 Constructor & Destructor Documentation

##### 5.274.1.1 **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

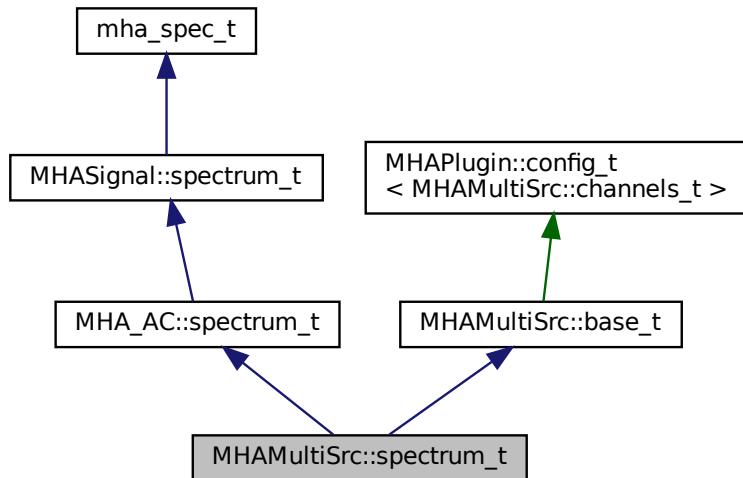
<i>route</i>	vector of source channel ids
<i>in_channels</i>	number of channels in the processed input signal

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

- **mha\_multisrc.h**
- **mha\_multisrc.cpp**

## 5.275 MHAMultiSrc::spectrum\_t Class Reference

Inheritance diagram for MHAMultiSrc::spectrum\_t:



## Public Member Functions

- **spectrum\_t ( algo\_comm\_t iac, std::string name, unsigned int frames, unsigned int channels)**
- **mha\_spec\_t \* update ( mha\_spec\_t \*s)**

*Update data of spectrum to hold actual input data.*

## Additional Inherited Members

### 5.275.1 Constructor & Destructor Documentation

```
5.275.1.1 spectrum_t() MHAMultiSrc::spectrum_t::spectrum_t (
    algo_comm_t iac,
    std::string name,
    unsigned int frames,
    unsigned int channels )
```

### 5.275.2 Member Function Documentation

```
5.275.2.1 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

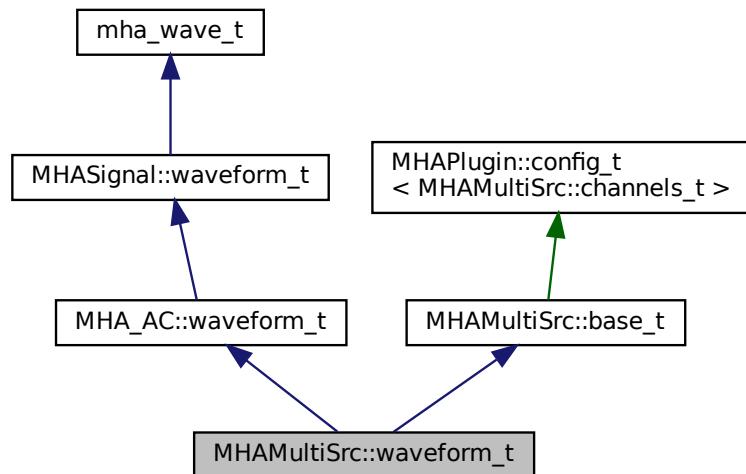
Return pointer to spectrum structure

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

- **mha\_multisrc.h**
- **mha\_multisrc.cpp**

## 5.276 MHAMultiSrc::waveform\_t Class Reference

Inheritance diagram for MHAMultiSrc::waveform\_t:



### Public Member Functions

- **waveform\_t ( algo\_comm\_t iac, std::string name, unsigned int frames, unsigned int channels)**
- **mha\_wave\_t \* update ( mha\_wave\_t \*s)**  
*Update data of waveform to hold actual input data.*

### Additional Inherited Members

#### 5.276.1 Constructor & Destructor Documentation

**5.276.1.1 waveform\_t()** MHAMultiSrc::waveform\_t::waveform\_t (

```

algo_comm_t iac,
std::string name,
unsigned int frames,
unsigned int channels )
  
```

## 5.276.2 Member Function Documentation

**5.276.2.1 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`

## 5.277 MHAOvlFilter::band\_descriptor\_t Class Reference

### Public Attributes

- `mha_real_t cf_l`
- `mha_real_t ef_l`
- `mha_real_t cf`
- `mha_real_t ef_h`
- `mha_real_t cf_h`
- `bool low_side_flat`
- `bool high_side_flat`

## 5.277.1 Member Data Documentation

**5.277.1.1 cf\_l** `mha_real_t` MHAOvlFilter::band\_descriptor\_t::cf\_l

**5.277.1.2 ef\_l** `mha_real_t` MHAOvlFilter::band\_descriptor\_t::ef\_l

**5.277.1.3 cf** `mha_real_t` MHAOvlFilter::band\_descriptor\_t::cf

**5.277.1.4 ef\_h** `mha_real_t` MHAOvlFilter::band\_descriptor\_t::ef\_h

**5.277.1.5 cf\_h** `mha_real_t` MHAOvlFilter::band\_descriptor\_t::cf\_h

**5.277.1.6 low\_side\_flat** `bool` MHAOvlFilter::band\_descriptor\_t::low\_side\_flat

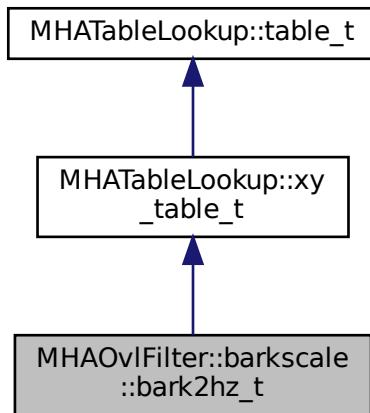
**5.277.1.7 high\_side\_flat** `bool` MHAOvlFilter::band\_descriptor\_t::high\_side\_flat

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

- `mha_fffb.hh`

## 5.278 MHAOvlFilter::barkscale::bark2hz\_t Class Reference

Inheritance diagram for MHAOvlFilter::barkscale::bark2hz\_t:



### Public Member Functions

- `bark2hz_t()`
- `~bark2hz_t()`

### Additional Inherited Members

#### 5.278.1 Constructor & Destructor Documentation

**5.278.1.1 bark2hz\_t()** MHAOvlFilter::barkscale::bark2hz\_t::bark2hz\_t ( )

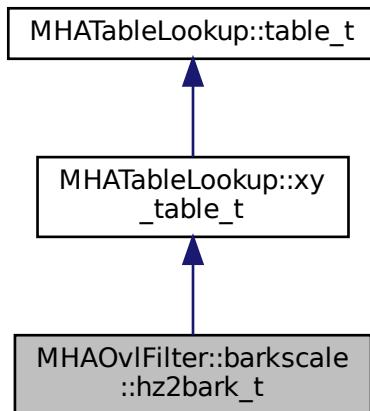
**5.278.1.2 ~bark2hz\_t()** MHAOvlFilter::barkscale::bark2hz\_t::~bark2hz\_t ( )

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

- `mha_fftfb.cpp`

## 5.279 MHAOvlFilter::barkscale::hz2bark\_t Class Reference

Inheritance diagram for MHAOvlFilter::barkscale::hz2bark\_t:



### Public Member Functions

- **hz2bark\_t ()**
- **~hz2bark\_t ()**

### Additional Inherited Members

#### 5.279.1 Constructor & Destructor Documentation

##### 5.279.1.1 **hz2bark\_t()** MHAOvlFilter::barkscale::hz2bark\_t::hz2bark\_t ( )

##### 5.279.1.2 **~hz2bark\_t()** MHAOvlFilter::barkscale::hz2bark\_t::~hz2bark\_t ( )

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

- **mha\_fffb.cpp**

## 5.280 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*

### 5.280.1 Constructor & Destructor Documentation

```
5.280.1.1 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 )
```

### 5.280.2 Member Function Documentation

```
5.280.2.1 insert() void MHAOvlFilter::fftfb_ac_info_t::insert ( )
```

### 5.280.3 Member Data Documentation

### 5.280.3.1 **cfv** `MHA_AC::waveform_t` `MHAOvlFilter::fftfb_ac_info_t::cfv` [private]

vector of nominal center frequencies / Hz

### 5.280.3.2 **efv** `MHA_AC::waveform_t` `MHAOvlFilter::fftfb_ac_info_t::efv` [private]

vector of edge frequencies / Hz

### 5.280.3.3 **bwv** `MHA_AC::waveform_t` `MHAOvlFilter::fftfb_ac_info_t::bwv` [private]

vector of band-weights (sum of squared fft-bin-weights)/num\_frames

### 5.280.3.4 **cLTASS** `MHA_AC::waveform_t` `MHAOvlFilter::fftfb_ac_info_t::cLTASS` [private]

vector of LTASS correction

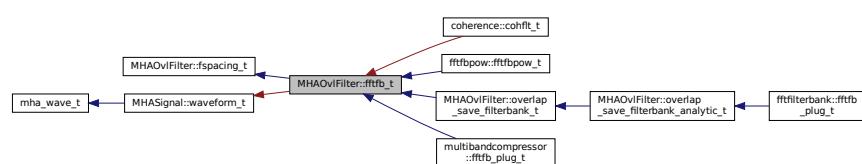
The documentation for this class was generated from the following files:

- `mha_fftfb.hh`
- `mha_fftfb.cpp`

## 5.281 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

### 5.281.1 Detailed Description

FFT based overlapping filter bank.

### 5.281.2 Constructor & Destructor Documentation

#### 5.281.2.1 **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

**5.281.2.2 ~fftfb\_t()** `MHAOvlFilter::fftfb_t::~fftfb_t ( )`

## 5.281.3 Member Function Documentation

**5.281.3.1 apply\_gains()** `void MHAOvlFilter::fftfb_t::apply_gains (`  
 `mha_spec_t * s_out,`  
 `const mha_spec_t * s_in,`  
 `const mha_wave_t * gains )`

**5.281.3.2 get\_fbpower()** `void MHAOvlFilter::fftfb_t::get_fbpower (`  
 `mha_wave_t * fbpow,`  
 `const mha_spec_t * s_in )`

**5.281.3.3 get\_fbpower\_db()** `void MHAOvlFilter::fftfb_t::get_fbpower_db (`  
 `mha_wave_t * fbpow,`  
 `const mha_spec_t * s_in )`

**5.281.3.4 get\_ltass\_gain\_db()** `std::vector< float > MHAOvlFilter::fftfb_t::get_ltass_gain_db ( ) const`

**5.281.3.5 bin1()** `unsigned int MHAOvlFilter::fftfb_t::bin1 ( unsigned int band ) const [inline]`

Return index of first non-zero filter shape window.

**5.281.3.6 bin2()** `unsigned int MHAOvlFilter::fftfb_t::bin2 ( unsigned int band ) const [inline]`

Return index of first zero filter shape window above center frequency.

**5.281.3.7 get\_ffflen()** `unsigned int MHAOvlFilter::fftfb_t::get_ffflen ( ) const [inline]`

Return fft length.

**5.281.3.8 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

## 5.281.4 Member Data Documentation

**5.281.4.1 vbin1** `unsigned int* MHAOvlFilter::fftfb_t::vbin1 [private]`

**5.281.4.2 vbin2** `unsigned int* MHAOvlFilter::fftfb_t::vbin2 [private]`

**5.281.4.3 shape** `mha_real_t (* MHAOvlFilter::fftfb_t::shape) (mha_real_t) [private]`

**5.281.4.4 fftlen** `unsigned int MHAOvlFilter::fftfb_t::ffflen [private]`

**5.281.4.5 samplingrate** `mha_real_t MHAOvlFilter::fftfb_t::samplingrate [private]`

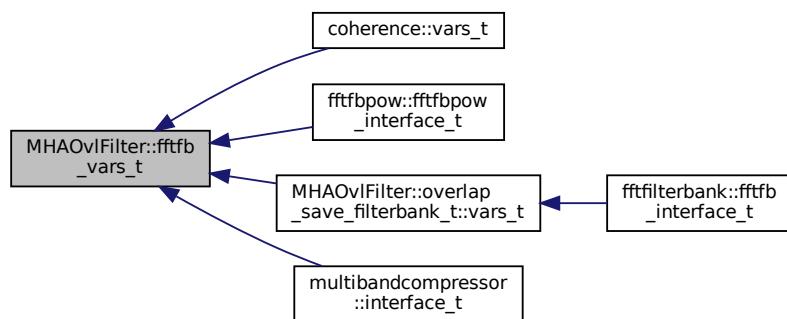
The documentation for this class was generated from the following files:

- `mha_fftfb.hh`
- `mha_fftfb.cpp`

## 5.282 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**

### 5.282.1 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.

## 5.282.2 Constructor & Destructor Documentation

**5.282.2.1 `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.
----------------	---

## 5.282.3 Member Data Documentation

**5.282.3.1 `fscale scale_var_t`** `MHAOvlFilter::fftfb_vars_t::fscale`

Frequency scale type (lin/bark/log/erb).

**5.282.3.2 `ovltype scale_var_t`** `MHAOvlFilter::fftfb_vars_t::ovltype`

Filter shape (rect/lin/hann).

**5.282.3.3 `plateau MHAParser::float_t`** `MHAOvlFilter::fftfb_vars_t::plateau`

relative plateau width.

**5.282.3.4 ftype** `MHAParser::kw_t MHAOvlFilter::fftfb_vars_t::ftype`

Flag to decide whether edge or center frequencies are used.

**5.282.3.5 f** `fscale_t MHAOvlFilter::fftfb_vars_t::f`

Frequency.

**5.282.3.6 normalize** `MHAParser::bool_t MHAOvlFilter::fftfb_vars_t::normalize`

Normalize sum of channels.

**5.282.3.7 fail\_on\_nonmonotonic** `MHAParser::bool_t MHAOvlFilter::fftfb_vars_t::fail_on_nonmonotonic`

Fail if frequency entries are non-monotonic (otherwise sort)

**5.282.3.8 fail\_on\_unique\_bins** `MHAParser::bool_t MHAOvlFilter::fftfb_vars_t::fail_on_unique_bins`

Fail if center frequencies share the same FFT bin.

**5.282.3.9 flag\_allow\_empty\_bands** `MHAParser::bool_t MHAOvlFilter::fftfb_vars_t::flag_allow_empty_bands`

Allow that frequency bands contain only zeros.

**5.282.3.10 cf** `MHAParser::vfloat_mon_t MHAOvlFilter::fftfb_vars_t::cf`

Final center frequencies in Hz.

**5.282.3.11 ef** `MHAParser::vfloat_mon_t MHAOvlFilter::fftfb_vars_t::ef`

Final edge frequencies in Hz.

**5.282.3.12 cLTASS** `MHAParser::vfloat_mon_t MHAOvlFilter::fftfb_vars_t::cLTASS`

Bandwidth correction for LTASS noise (level of 0 dB RMS LTASS noise)

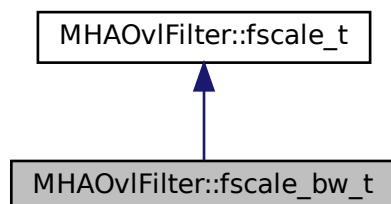
**5.282.3.13 shapes** `MHAParser::mfloat_mon_t MHAOvlFilter::fftfb_vars_t::shapes`

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

- `mha_fftfb.hh`
- `mha_fftfb.cpp`

**5.283 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

### 5.283.1 Constructor & Destructor Documentation

**5.283.1.1 `fscale_bw_t()`** `MHAOvlFilter::fscale_bw_t::fscale_bw_t ( MHParse::parser_t & parent )`

### 5.283.2 Member Function Documentation

**5.283.2.1 `get_bw_hz()`** `std::vector< mha_real_t > MHAOvlFilter::fscale_bw_t::get_bw_hz ( ) const`

**5.283.2.2 `update_hz()`** `void MHAOvlFilter::fscale_bw_t::update_hz ( ) [private]`

### 5.283.3 Member Data Documentation

**5.283.3.1 bw** `MHAParser::vfloat_t` `MHAOvlFilter::fscale_bw_t::bw` [protected]

**5.283.3.2 bw\_hz** `MHAParser::vfloat_mon_t` `MHAOvlFilter::fscale_bw_t::bw_hz` [protected]

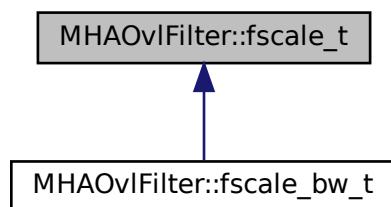
**5.283.3.3 updater** `MHAEvents::connector_t< fscale_bw_t>` `MHAOvlFilter::fscale_bw_t::updater` [private]

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

- `mha_fffb.hh`
- `mha_fffb.cpp`

### 5.284 MHAOvlFilter::fscale\_t Class Reference

Inheritance diagram for MHAOvlFilter::fscale\_t:



#### Public Member Functions

- `fscale_t ( MHAParser::parser_t &parent)`
- `std::vector< mha_real_t > get_f_hz () const`

## Public Attributes

- `scale_var_t unit`
- `MHAParser::vfloat_t f`
- `MHAParser::vfloat_mon_t f_hz`

## Private Member Functions

- `void update_hz()`

## Private Attributes

- `MHAEvents::connector_t< fscale_t > updater`

### 5.284.1 Constructor & Destructor Documentation

**5.284.1.1 `fscale_t()`** `MHAOvlFilter::fscale_t::fscale_t (`  
`MHAParser::parser_t & parent )`

### 5.284.2 Member Function Documentation

**5.284.2.1 `get_f_hz()`** `std::vector< mha_real_t > MHAOvlFilter::fscale_t::get_f_hz (`  
`) const`

**5.284.2.2 `update_hz()`** `void MHAOvlFilter::fscale_t::update_hz ( ) [private]`

### 5.284.3 Member Data Documentation

**5.284.3.1 unit scale\_var\_t MHAOvlFilter::fscale\_t::unit**

**5.284.3.2 f MHAParser::vfloat\_t MHAOvlFilter::fscale\_t::f**

**5.284.3.3 f\_hz MHAParser::vfloat\_mon\_t MHAOvlFilter::fscale\_t::f\_hz**

**5.284.3.4 updater MHAEvents::connector\_t< fscale\_t> MHAOvlFilter::fscale\_t:::updater [private]**

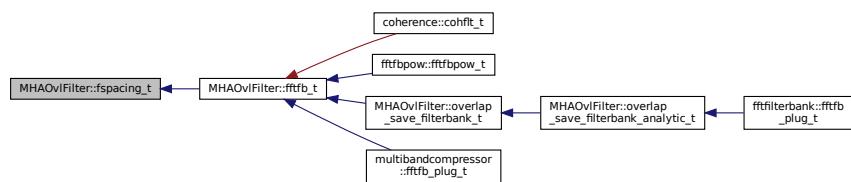
The documentation for this class was generated from the following files:

- **mha\_fftfb.hh**
- **mha\_fftfb.cpp**

## 5.285 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\_**

## 5.285.1 Detailed Description

Class for frequency spacing, used by filterbank shape generator class.

## 5.285.2 Constructor & Destructor Documentation

```
5.285.2.1 fspacing_t() MHAOvlFilter::fspacing_t::fspacing_t (
    const MHAOvlFilter::fftfb_vars_t & par,
    unsigned int nfft,
    mha_real_t fs )
```

## 5.285.3 Member Function Documentation

**5.285.3.1 `get_cf_fftbin()`** std::vector< unsigned int > MHAOvlFilter::fspacing\_t::get\_cf\_fftbin ( ) const

**5.285.3.2 `get_cf_hz()`** std::vector< mha\_real\_t > MHAOvlFilter::fspacing\_t::get\_cf\_hz ( ) const

**5.285.3.3 `get_ef_hz()`** std::vector< mha\_real\_t > MHAOvlFilter::fspacing\_t::get\_ef\_hz ( ) const

**5.285.3.4 `nbands()`** unsigned int MHAOvlFilter::fspacing\_t::nbands ( ) const [inline]

Return number of bands in filter bank.

**5.285.3.5 `fail_on_nonmonotonic_cf()`** void MHAOvlFilter::fspacing\_t::fail\_on\_nonmonotonic\_cf ( ) [protected]

**5.285.3.6 `fail_on_unique_fftbins()`** void MHAOvlFilter::fspacing\_t::fail\_on\_unique\_fftbins ( ) [protected]

**5.285.3.7 `ef2bands()`** void MHAOvlFilter::fspacing\_t::ef2bands ( std::vector< mha\_real\_t > vef ) [private]

**5.285.3.8 `cf2bands()`** void MHAOvlFilter::fspacing\_t::cf2bands ( std::vector< mha\_real\_t > vcf ) [private]

**5.285.3.9 equidist2bands()** void MHAOvlFilter::fspacing\_t::equidist2bands ( std::vector< mha\_real\_t > vcf ) [private]

## 5.285.4 Member Data Documentation

**5.285.4.1 bands** std::vector< MHAOvlFilter::band\_descriptor\_t > MHAOvlFilter::fspacing\_t::bands [protected]

**5.285.4.2 symmetry\_scale** mha\_real\_t (\* MHAOvlFilter::fspacing\_t::symmetry\_scale) ( mha\_real\_t ) [protected]

**5.285.4.3 nfft\_** unsigned int MHAOvlFilter::fspacing\_t::nfft\_ [private]

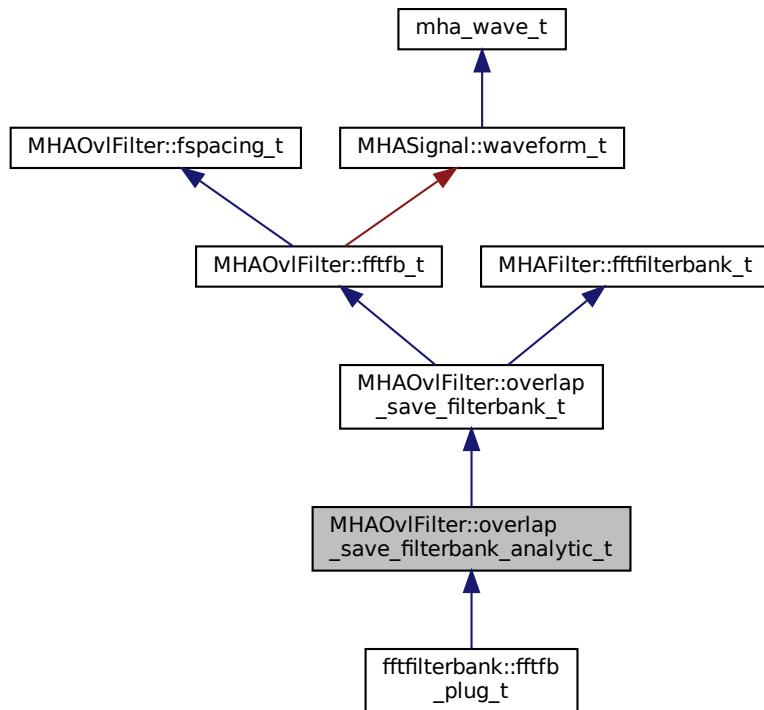
**5.285.4.4 fs\_** mha\_real\_t MHAOvlFilter::fspacing\_t::fs\_ [private]

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

- **mha\_fftfb.hh**
- **mha\_fftfb.cpp**

## 5.286 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

#### 5.286.1 Constructor & Destructor Documentation

## **5.287 MHAOvlFilter::overlap\_save\_filterbank\_t Class Reference**

```
5.286.1.1 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 )
```

### **5.286.2 Member Function Documentation**

```
5.286.2.1 filter_analytic() void MHAOvlFilter::overlap_save_filterbank_analytic_t::filter_analytic ( const mha_wave_t * sIn, mha_wave_t ** fltRe, mha_wave_t ** fltIm )
```

### **5.286.3 Member Data Documentation**

```
5.286.3.1 imagfb MHAFilter::fftfilterbank_t MHAOvlFilter::overlap_save_filterbank_analytic_t::imagfb [private]
```

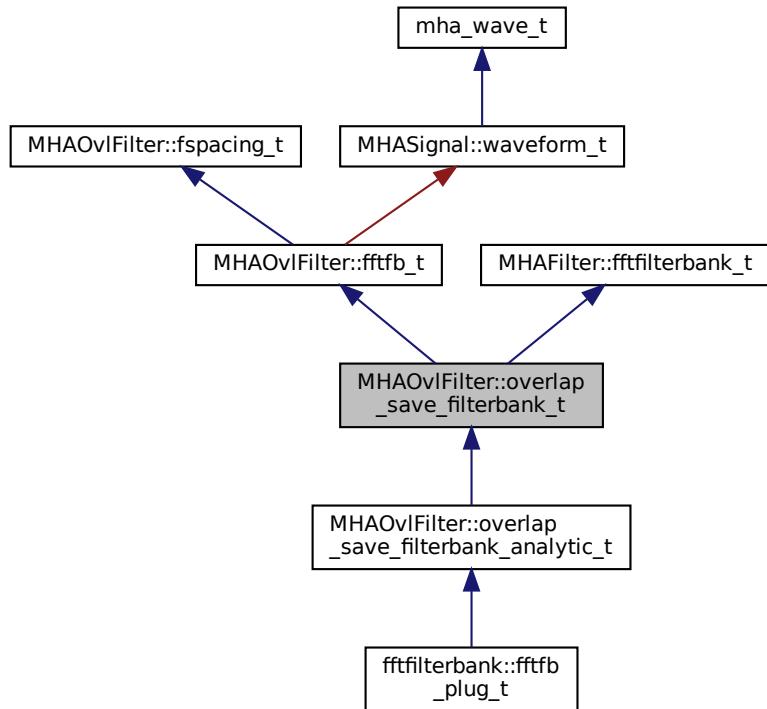
The documentation for this class was generated from the following files:

- **mha\_fftfb.hh**
- **mha\_fftfb.cpp**

## **5.287 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

### 5.287.1 Detailed Description

A time-domain minimal phase filter bank with frequency shapes from **MHAOvlFilter::fftfb\_t** (p. 1002).

## 5.287.2 Constructor & Destructor Documentation

**5.287.2.1 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 )`

## 5.287.3 Member Function Documentation

**5.287.3.1 get\_channelconfig()** `mhaconfig_t MHAOvlFilter::overlap_save_filterbank_t::get_channelconfig ( ) const [inline]`

## 5.287.4 Member Data Documentation

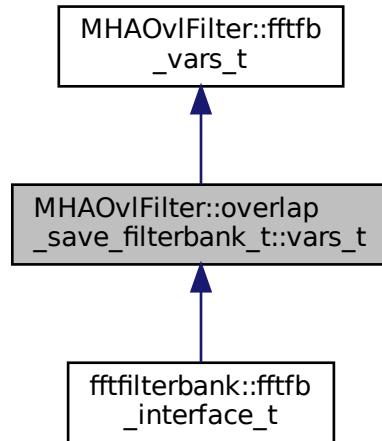
**5.287.4.1 channelconfig\_out\_** `mhaconfig_t MHAOvlFilter::overlap_save_filterbank_t::channelconfig_out_ [private]`

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

- `mha_fftffb.hh`
- `mha_fftffb.cpp`

## 5.288 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**

### 5.288.1 Constructor & Destructor Documentation

**5.288.1.1 vars\_t()** MHAOvlFilter::overlap\_save\_filterbank\_t::vars\_t::vars\_t ( MHParse::parser\_t & p )

## 5.288.2 Member Data Documentation

**5.288.2.1 fftlen** `MHAParser::int_t MHAOvlFilter::overlap_save_filterbank_t::vars_t::fftlen`

**5.288.2.2 phasemode1** `MHAParser::kw_t MHAOvlFilter::overlap_save_filterbank_t::vars_t::phasemode1`

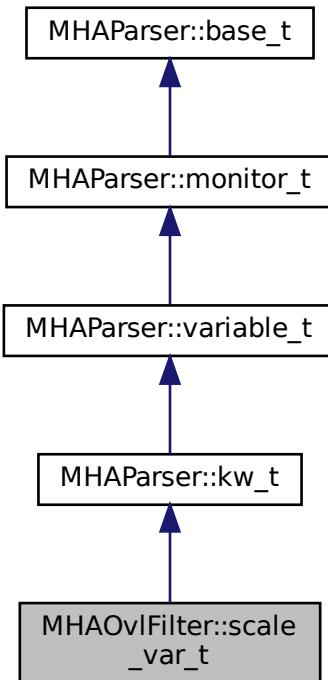
**5.288.2.3 irswnd** `MHAParser::window_t MHAOvlFilter::overlap_save_filterbank_t::vars_t::irswnd`

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

- `mha_fftfb.hh`
- `mha_fftfb.cpp`

## 5.289 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

### 5.289.1 Constructor & Destructor Documentation

**5.289.1.1 scale\_var\_t()** MHAOvlFilter::scale\_var\_t::scale\_var\_t (const std::string & *help*)

### 5.289.2 Member Function Documentation

**5.289.2.1 add\_fun()** void MHAOvlFilter::scale\_var\_t::add\_fun (const std::string & *name*,  
**scale\_fun\_t** \* *fun*)

**5.289.2.2 get\_name()** std::string MHAOvlFilter::scale\_var\_t::get\_name ( ) const [inline]

**5.289.2.3 get\_fun()** **scale\_fun\_t**\* MHAOvlFilter::scale\_var\_t::get\_fun ( ) const [inline]

**5.289.2.4 hz2unit()** **mha\_real\_t** MHAOvlFilter::scale\_var\_t::hz2unit (**mha\_real\_t** *x*) const

**5.289.2.5 unit2hz()** **mha\_real\_t** MHAOvlFilter::scale\_var\_t::unit2hz (**mha\_real\_t** *x*) const

### 5.289.3 Member Data Documentation

**5.289.3.1 names** std::vector<std::string> MHAOvlFilter::scale\_var\_t::names [private]

**5.289.3.2 funs** std::vector< scale\_fun\_t\*> MHAOvlFilter::scale\_var\_t::funs [private]

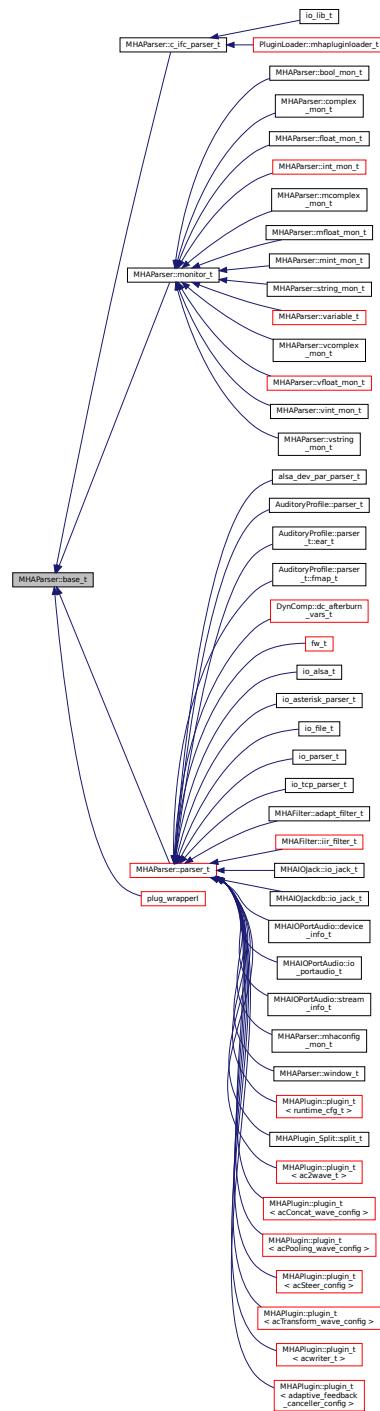
The documentation for this class was generated from the following files:

- **mha\_fftfb.hh**
- **mha\_fftfb.cpp**

## 5.290 MHAParser::base\_t Class Reference

Base class for all parser items.

## Inheritance diagram for MHAParser::base\_t:



## Classes

- class **replace** t

## Public Member Functions

- **base\_t** (const std::string &)
 

*Constructor for base class of all parser nodes.*
- **base\_t** (const **base\_t** &)
 

*Copy constructor for **base\_t** (p. 1026).*
- **base\_t & operator=** (const **base\_t** &) =default
- **base\_t ( base\_t &&)** =delete
- **base\_t & operator= ( base\_t &&)** =delete
- virtual ~**base\_t** ()
- virtual std::string **parse** (const std::string &)
 

*Causes this node to process a command in the openMHA configuration language.*
- virtual void **parse** (const char \*, char \*, unsigned int)
 

*This function parses a command and writes the parsing result into a C character array.*
- virtual void **parse** (const std::vector< std::string > &, std::vector< std::string > &)
- virtual std::string **op\_subparse** ( **expression\_t** &)
- virtual std::string **op\_setval** ( **expression\_t** &)
- virtual std::string **op\_query** ( **expression\_t** &)
- virtual std::string **query\_dump** (const std::string &)
- virtual std::string **query\_entries** (const std::string &)
- virtual std::string **query\_perm** (const std::string &)
- virtual std::string **query\_range** (const std::string &)
- virtual std::string **query\_type** (const std::string &)
- virtual std::string **query\_val** (const std::string &)
- virtual std::string **query\_readfile** (const std::string &)
- virtual std::string **query\_savefile** (const std::string &)
- virtual std::string **query\_savefile\_compact** (const std::string &)
- virtual std::string **query\_savemons** (const std::string &)
- virtual std::string **query\_listids** (const std::string &)
- std::string **query\_version** (const std::string &)
- std::string **query\_id** (const std::string &)
- std::string **query\_subst** (const std::string &)
- std::string **query\_addsubst** (const std::string &)
- std::string **query\_help** (const std::string &)
- std::string **query\_cmds** (const std::string &)
- void **set\_node\_id** (const std::string &)
 

*Set the identification string of this parser node.*
- void **set\_help** (const std::string &)
 

*Set the help comment of a variable or parser.*
- void **add\_parent\_on\_insert** ( **parser\_t** \*, std::string)
- void **rm\_parent\_on\_remove** ( **parser\_t** \*)
- const std::string & **fullname** () const
 

*Return the full dot-separated path name of this parser node in the openMHA configuration tree.*

## Public Attributes

- **MHAEvents::emitter\_t writeaccess**  
*Event emitted on write access.*
- **MHAEvents::emitter\_t valuechanged**  
*Event emitted if the value has changed.*
- **MHAEvents::emitter\_t readaccess**  
*Event emitted on read access.*
- **MHAEvents::emitter\_t prereadaccess**  
*Event emitted on read access, before the data field is accessed.*

## Protected Member Functions

- void **activate\_query** (const std::string &, **query\_t**)
- void **notify** ()

## Protected Attributes

- **query\_map\_t queries**
- bool **data\_is\_initialized**

## Private Types

- typedef std::vector< **replace\_t** > **repl\_list\_t**

## Private Member Functions

- void **add\_replace\_pair** (const std::string &, const std::string &)
- std::string **oplist** ()

## Private Attributes

- std::string **help**
- std::string **id\_str**
- **opact\_map\_t operators**
- **repl\_list\_t repl\_list**
- bool **nested\_lock**
- **parser\_t \* parent**
- std::string **thefullname**

### 5.290.1 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.

### 5.290.2 Member Typedef Documentation

**5.290.2.1 `repl_list_t`** `typedef std::vector< replace_t > MHParse::base_t::repl_list_t` [private]

### 5.290.3 Constructor & Destructor Documentation

**5.290.3.1 `base_t()` [1/3]** `MHParse::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.
----------------	--

**5.290.3.2 `base_t()` [2/3]** `MHParse::base_t::base_t ( const base_t & src )`

Copy constructor for `base_t` (p. 1026).

Copies help text and id string, but does not insert the new node into the parser tree structure.

**Parameters**

<code>src</code>	Source parser
------------------	---------------

**Deprecated** Copying parser nodes makes little sense, avoid wherever possible

**5.290.3.3 `base_t()` [3/3]** `MHAParser::base_t::base_t (`  
`base_t && ) [delete]`

**5.290.3.4 `~base_t()`** `MHAParser::base_t::~base_t ( ) [virtual]`

## 5.290.4 Member Function Documentation

**5.290.4.1 `operator=()` [1/2]** `base_t& MHAParser::base_t::operator= (`  
`const base_t & ) [default]`

**5.290.4.2 `operator=()` [2/2]** `base_t& MHAParser::base_t::operator= (`  
`base_t && ) [delete]`

**5.290.4.3 `parse()` [1/3]** `std::string MHAParser::base_t::parse (`  
`const std::string & cs ) [virtual]`

Causes this node to process a command in the openMHA configuration language.

## Parameters

<i>cs</i>	The command to parse
-----------	----------------------

## Returns

The response to the command, if successful

## Exceptions

<b>MHA_Error</b> (p. <a href="#">760</a> )	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. [1349](#)), **PluginLoader::mhapluginloader\_t** (p. [1367](#)), **plug\_wrapper** (p. [1347](#)), **io\_wrapper** (p. [640](#)), and **altplugs\_t** (p. [302](#)).

```
5.290.4.4 parse() [2/3] void MHAParser::base_t::parse (
    const char * cmd,
    char * retv,
    unsigned int len ) [virtual]
```

This function parses a command and writes the parsing result into a C character array.

This base class implementation delegates to **parse(const std::string &)** (p. [1031](#)).

## Parameters

<i>cmd</i>	Command to be parsed
<i>retv</i>	Buffer for the result
<i>len</i>	Length of buffer

Reimplemented in **altplugs\_t** (p. [302](#)).

```
5.290.4.5 parse() [3/3] void MHAParser::base_t::parse (
    const std::vector< std::string > & cs,
    std::vector< std::string > & retv ) [virtual]
```

**5.290.4.6 op\_subparse()** std::string MHAParser::base\_t::op\_subparse ( expression\_t & ) [virtual]

Reimplemented in **MHAParser::c\_ifc\_parser\_t** (p. 1047), and **MHAParser::parser\_t** (p. 1101).

**5.290.4.7 op\_setval()** std::string MHAParser::base\_t::op\_setval ( expression\_t & ) [virtual]

Reimplemented in **MHAParser::c\_ifc\_parser\_t** (p. 1047), **MHAParser::mcomplex\_t** (p. 1078), **MHAParser::mfloat\_t** (p. 1083), **MHAParser::mint\_t** (p. 1095), **MHAParser::vcomplex\_t** (p. 1119), **MHAParser::vfloat\_t** (p. 1125), **MHAParser::vint\_t** (p. 1129), **MHAParser::complex\_t** (p. 1054), **MHAParser::float\_t** (p. 1061), **MHAParser::int\_t** (p. 1066), **MHAParser::bool\_t** (p. 1045), **MHAParser::vstring\_t** (p. 1133), **MHAParser::string\_t** (p. 1113), **MHAParser::kw\_t** (p. 1073), **MHAParser::variable\_t** (p. 1115), and **MHAParser::parser\_t** (p. 1102).

**5.290.4.8 op\_query()** std::string MHAParser::base\_t::op\_query ( expression\_t & ) [virtual]

Reimplemented in **MHAParser::c\_ifc\_parser\_t** (p. 1047), **MHAParser::monitor\_t** (p. 1097), and **MHAParser::parser\_t** (p. 1102).

**5.290.4.9 query\_dump()** std::string MHAParser::base\_t::query\_dump ( const std::string & ) [virtual]

Reimplemented in **MHAParser::monitor\_t** (p. 1097), and **MHAParser::parser\_t** (p. 1102).

**5.290.4.10 query\_entries()** std::string MHAParser::base\_t::query\_entries ( const std::string & ) [virtual]

Reimplemented in **MHAParser::parser\_t** (p. 1102).

---

**5.290.4.11 query\_perm()** std::string MHAParser::base\_t::query\_perm (const std::string & ) [virtual]

Reimplemented in **MHAParser::variable\_t** (p. 1115), and **MHAParser::monitor\_t** (p. 1097).

**5.290.4.12 query\_range()** std::string MHAParser::base\_t::query\_range (const std::string & ) [virtual]

Reimplemented in **MHAParser::kw\_t** (p. 1074), and **MHAParser::range\_var\_t** (p. 1106).

**5.290.4.13 query\_type()** std::string MHAParser::base\_t::query\_type (const std::string & ) [virtual]

Reimplemented in **MHAParser::mcomplex\_mon\_t** (p. 1076), **MHAParser::vcomplex\_mon\_t** (p. 1117), **MHAParser::complex\_mon\_t** (p. 1052), **MHAParser::float\_mon\_t** (p. 1059), **MHAParser::mfloat\_mon\_t** (p. 1080), **MHAParser::vfloat\_mon\_t** (p. 1122), **MHAParser::mint\_mon\_t** (p. 1092), **MHAParser::vint\_mon\_t** (p. 1127), **MHAParser::vstring\_mon\_t** (p. 1132), **MHAParser::string\_mon\_t** (p. 1110), **MHAParser::bool\_mon\_t** (p. 1042), **MHAParser::int\_mon\_t** (p. 1064), **MHAParser::mcomplex\_t** (p. 1078), **MHAParser::mfloat\_t** (p. 1083), **MHAParser::mint\_t** (p. 1095), **MHAParser::vcomplex\_t** (p. 1120), **MHAParser::vfloat\_t** (p. 1125), **MHAParser::vint\_t** (p. 1129), **MHAParser::complex\_t** (p. 1054), **MHAParser::float\_t** (p. 1061), **MHAParser::int\_t** (p. 1067), **MHAParser::bool\_t** (p. 1045), **MHAParser::vstring\_t** (p. 1133), **MHAParser::string\_t** (p. 1113), **MHAParser::kw\_t** (p. 1074), and **MHAParser::parser\_t** (p. 1102).

**5.290.4.14 query\_val()** std::string MHAParser::base\_t::query\_val (const std::string & ) [virtual]

Reimplemented in **MHAParser::mcomplex\_mon\_t** (p. 1076), **MHAParser::vcomplex\_mon\_t** (p. 1117), **MHAParser::complex\_mon\_t** (p. 1052), **MHAParser::float\_mon\_t** (p. 1058), **MHAParser::mfloat\_mon\_t** (p. 1080), **MHAParser::vfloat\_mon\_t** (p. 1122), **MHAParser::mint\_mon\_t** (p. 1092), **MHAParser::vint\_mon\_t** (p. 1127), **MHAParser::vstring\_mon\_t** (p. 1131), **MHAParser::string\_mon\_t** (p. 1110), **MHAParser::bool\_mon\_t** (p. 1042), **MHAParser::int\_mon\_t** (p. 1064), **MHAParser::mcomplex\_t** (p. 1078), **MHAParser::mfloat\_t** (p. 1083), **MHAParser::mint\_t** (p. 1095), **MHAParser::vcomplex\_t** (p. 1120), **MHAParser::vfloat\_t** (p. 1125), **MHAParser::vint\_t** (p. 1129), **MHAParser::complex\_t** (p. 1054), **MHAParser::float\_t** (p. 1062), **MHAParser::int\_t** (p. 1067), **MHAParser::bool\_t** (p. 1045), **MHAParser::vstring\_t** (p. 1134), **MHAParser::string\_t** (p. 1113), **MHAParser::kw\_t** (p. 1074), and **MHAParser::parser\_t** (p. 1103).

**5.290.4.15 query\_readfile()** std::string MHAParser::base\_t::query\_readfile (const std::string & ) [virtual]

Reimplemented in **MHAParser::parser\_t** (p. 1102).

**5.290.4.16 query\_savefile()** std::string MHAParser::base\_t::query\_savefile (const std::string & ) [virtual]

Reimplemented in **MHAParser::parser\_t** (p. 1103).

**5.290.4.17 query\_savefile\_compact()** std::string MHAParser::base\_t::query\_savefile\_compact (const std::string & ) [virtual]

Reimplemented in **MHAParser::parser\_t** (p. 1103).

**5.290.4.18 query\_savemons()** std::string MHAParser::base\_t::query\_savemons (const std::string & ) [virtual]

Reimplemented in **MHAParser::parser\_t** (p. 1103).

**5.290.4.19 query\_listids()** std::string MHAParser::base\_t::query\_listids (const std::string & ) [virtual]

Reimplemented in **MHAParser::parser\_t** (p. 1103).

**5.290.4.20 query\_version()** std::string MHAParser::base\_t::query\_version (const std::string & )

**5.290.4.21 query\_id()** std::string MHAParser::base\_t::query\_id ( const std::string & )

**5.290.4.22 query\_subst()** std::string MHAParser::base\_t::query\_subst ( const std::string & )

**5.290.4.23 query\_addsubst()** std::string MHAParser::base\_t::query\_addsubst ( const std::string & s )

**5.290.4.24 query\_help()** std::string MHAParser::base\_t::query\_help ( const std::string & )

**5.290.4.25 query\_cmds()** std::string MHAParser::base\_t::query\_cmds ( const std::string & )

**5.290.4.26 set\_node\_id()** void MHAParser::base\_t::set\_node\_id ( const std::string & s )

Set the identification string of this parser node.

The id can be queried from the configuration language using the ?id query command. Nodes can be found by id using the ?listid query command on a containing parser node.

#### Parameters

<b>s</b>	The new identification string.
----------	--------------------------------

**5.290.4.27 set\_help()** void MHAParser::base\_t::set\_help (

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

Set the help comment of a variable or parser.

**Parameters**

s	New help comment.
---	-------------------

**5.290.4.28 add\_parent\_on\_insert()** void MHAParser::base\_t::add\_parent\_on\_insert (   
   **parser\_t** \* p,  
   std::string n )

**5.290.4.29 rm\_parent\_on\_remove()** void MHAParser::base\_t::rm\_parent\_on\_remove (   
   **parser\_t** \* )

**5.290.4.30 fullname()** const std::string & MHAParser::base\_t::fullname ( ) const

Return the full dot-separated path name of this parser node in the openMHA configuration tree.

**5.290.4.31 activate\_query()** void MHAParser::base\_t::activate\_query (   
   const std::string & n,  
   **query\_t** a ) [protected]

**5.290.4.32 notify()** void MHAParser::base\_t::notify ( ) [protected]

**5.290.4.33 add\_replace\_pair()** void MHAParser::base\_t::add\_replace\_pair (   
   const std::string & a,  
   const std::string & b ) [private]

**5.290.4.34 oplist()** std::string MHAParser::base\_t::oplist ( ) [private]**5.290.5 Member Data Documentation****5.290.5.1 writeaccess** MHAEvents::emitter\_t MHAParser::base\_t::writeaccess

Event emitted on write access.

To connect a callback that is invoked on write access to this parser variable, use MHAEvents::patchbay\_t<receiver\_t> method connect(&writeaccess,&receiver\_t::callback) where callback is a method that expects no parameters and returns void.

**5.290.5.2 valuechanged** MHAEvents::emitter\_t MHAParser::base\_t::valuechanged

Event emitted if the value has changed.

To connect a callback that is invoked when write access to this parser variable actually changes its value, use MHAEvents::patchbay\_t<receiver\_t> method connect(&valuechanged,&receiver\_t::callback) where callback is a method that expects no parameters and returns void.

**5.290.5.3 readaccess** MHAEvents::emitter\_t MHAParser::base\_t::readaccess

Event emitted on read access.

To connect a callback that is invoked after the value of this variable has been read through the configuration interface, use MHAEvents::patchbay\_t<receiver\_t> method connect(&readaccess,&receiver\_t::callback) where callback is a method that expects no parameters and returns void.

**5.290.5.4 prereadaccess** MHAEvents::emitter\_t MHAParser::base\_t::prereadaccess

Event emitted on read access, before the data field is accessed.

To connect a callback that is invoked when the value of this variable is about to be read through the configuration interface, so that the callback can influence the value that is reported, use MHAEvents::patchbay\_t<receiver\_t> method connect(&prereadaccess,&receiver\_t::callback) where callback is a method that expects no parameters and returns void.

**5.290.5.5 queries** `query_map_t` MHAParser::base\_t::queries [protected]

**5.290.5.6 data\_is\_initialized** `bool` MHAParser::base\_t::data\_is\_initialized [protected]

**5.290.5.7 help** `std::string` MHAParser::base\_t::help [private]

**5.290.5.8 id\_str** `std::string` MHAParser::base\_t::id\_str [private]

**5.290.5.9 operators** `opact_map_t` MHAParser::base\_t::operators [private]

**5.290.5.10 repl\_list** `repl_list_t` MHAParser::base\_t::repl\_list [private]

**5.290.5.11 nested\_lock** `bool` MHAParser::base\_t::nested\_lock [private]

**5.290.5.12 parent** `parser_t*` MHAParser::base\_t::parent [private]

**5.290.5.13 thefullname** `std::string` MHAParser::base\_t::thefullname [private]

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

- `mha_parser.hh`
- `mha_parser.cpp`

## 5.291 MHAParser::base\_t::replace\_t Class Reference

### Public Member Functions

- **replace\_t** (const std::string &, const std::string &)
- void **replace** (std::string &)
- const std::string & **get\_a** () const
- const std::string & **get\_b** () const

### Private Attributes

- std::string **a**
- std::string **b**

#### 5.291.1 Constructor & Destructor Documentation

**5.291.1.1 replace\_t()** MHAParser::base\_t::replace\_t::replace\_t (const std::string & ia, const std::string & ib )

#### 5.291.2 Member Function Documentation

**5.291.2.1 replace()** void MHAParser::base\_t::replace\_t::replace (std::string & s )

**5.291.2.2 get\_a()** const std::string& MHAParser::base\_t::replace\_t::get\_a ( ) const [inline]

**5.291.2.3 get\_b()** const std::string& MHAParser::base\_t::replace\_t::get\_b ( ) const [inline]

### 5.291.3 Member Data Documentation

**5.291.3.1 a** std::string MHAParser::base\_t::replace\_t::a [private]

**5.291.3.2 b** std::string MHAParser::base\_t::replace\_t::b [private]

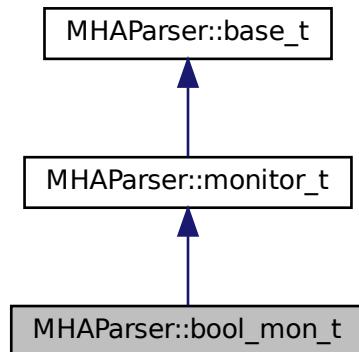
The documentation for this class was generated from the following files:

- [mha\\_parser.hh](#)
- [mha\\_parser.cpp](#)

## 5.292 MHAParser::bool\_mon\_t Class Reference

Monitor with string value.

Inheritance diagram for MHAParser::bool\_mon\_t:



### Public Member Functions

- **bool\_mon\_t** (const std::string &hlp)  
*Create a monitor variable for string values.*

## Public Attributes

- bool **data**

*Data field.*

## Protected Member Functions

- std::string **query\_val** (const std::string &)
- std::string **query\_type** (const std::string &)

## Additional Inherited Members

### 5.292.1 Detailed Description

Monitor with string value.

### 5.292.2 Constructor & Destructor Documentation

#### 5.292.2.1 **bool\_mon\_t()** MHAParser::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.
------------	---

### 5.292.3 Member Function Documentation

#### 5.292.3.1 **query\_val()** std::string MHAParser::bool\_mon\_t::query\_val ( const std::string & *s* ) [protected], [virtual]

Reimplemented from **MHAParser::base\_t** (p. 1034).

**5.292.3.2 query\_type()** std::string MHAParser::bool\_mon\_t::query\_type ( const std::string & ) [protected], [virtual]

Reimplemented from **MHAParser::base\_t** (p. 1034).

#### 5.292.4 Member Data Documentation

**5.292.4.1 data** bool MHAParser::bool\_mon\_t::data

Data field.

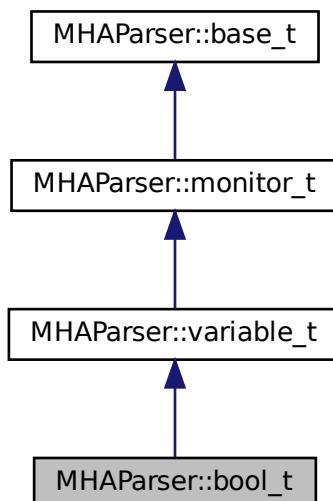
The documentation for this class was generated from the following files:

- **mha\_parser.hh**
- **mha\_parser.cpp**

## 5.293 MHAParser::bool\_t Class Reference

Variable with a boolean value ("yes"/"no")

Inheritance diagram for MHAParser::bool\_t:



## Public Member Functions

- **bool\_t** (const std::string &help\_text, const std::string &initial\_value)  
*Constructor for a configuration language variable for boolean values.*

## Public Attributes

- **bool data**  
*Data field.*

## Protected Member Functions

- std::string **op\_setval** ( expression\_t &)
- std::string **query\_type** (const std::string &)
- std::string **query\_val** (const std::string &)

## Additional Inherited Members

### 5.293.1 Detailed Description

Variable with a boolean value ("yes"/"no")

### 5.293.2 Constructor & Destructor Documentation

```
5.293.2.1 bool_t() MHAParser::bool_t::bool_t (
    const std::string & help_text,
    const std::string & initial_value )
```

Constructor for a configuration language variable for boolean values.

#### Parameters

<i>help_text</i>	A human-readable text describing the purpose of this configuration variable.
<i>initial_value</i>	The initial value for this variable as a string. The string representation of 'true' is either "yes" or "1". The string representation of 'false' is either "no" or "0".

### 5.293.3 Member Function Documentation

**5.293.3.1 `op_setval()`** std::string MHAParser::bool\_t::op\_setval ( expression\_t & x ) [protected], [virtual]

Reimplemented from **MHAParser::variable\_t** (p. 1115).

**5.293.3.2 `query_type()`** std::string MHAParser::bool\_t::query\_type ( const std::string & ) [protected], [virtual]

Reimplemented from **MHAParser::base\_t** (p. 1034).

**5.293.3.3 `query_val()`** std::string MHAParser::bool\_t::query\_val ( const std::string & s ) [protected], [virtual]

Reimplemented from **MHAParser::base\_t** (p. 1034).

### 5.293.4 Member Data Documentation

**5.293.4.1 `data`** bool MHAParser::bool\_t::data

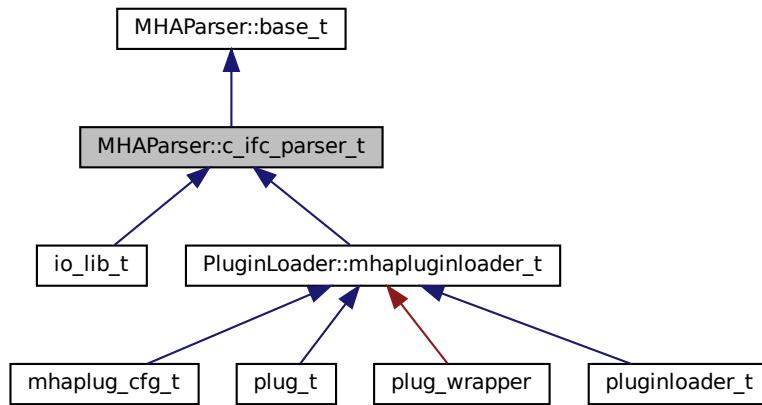
Data field.

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

- [mha\\_parser.hh](#)
- [mha\\_parser.cpp](#)

## 5.294 MHAParser::c\_ifc\_parser\_t Class Reference

Inheritance diagram for MHAParser::c\_ifc\_parser\_t:



### Public Member Functions

- `c_ifc_parser_t (const std::string &modulename_)`
- `~c_ifc_parser_t ()`
- `void set_parse_cb ( c_parse_cmd_t, c_parse_err_t, void * )`

### Protected Member Functions

- `std::string op_subparse ( MHAParser::expression_t &)`
- `std::string op_setval ( MHAParser::expression_t &)`
- `std::string op_query ( MHAParser::expression_t &)`

### Private Member Functions

- `void test_error ()`

### Private Attributes

- `std::string modulename`
- `c_parse_cmd_t c_parse_cmd`
- `c_parse_err_t c_parse_err`
- `int liberr`
- `void * libdata`
- `unsigned int ret_size`
- `char * retv`

## Additional Inherited Members

### 5.294.1 Constructor & Destructor Documentation

**5.294.1.1 c\_ifc\_parser\_t()** MHAParser::c\_ifc\_parser\_t::c\_ifc\_parser\_t (const std::string & modulename\_ )

**5.294.1.2 ~c\_ifc\_parser\_t()** MHAParser::c\_ifc\_parser\_t::~c\_ifc\_parser\_t ( )

### 5.294.2 Member Function Documentation

**5.294.2.1 set\_parse\_cb()** void MHAParser::c\_ifc\_parser\_t::set\_parse\_cb (c\_parse\_cmd\_t , c\_parse\_err\_t , void \* )

**5.294.2.2 op\_subparse()** std::string MHAParser::c\_ifc\_parser\_t::op\_subparse (MHAParser::expression\_t & x ) [protected], [virtual]

Reimplemented from **MHAParser::base\_t** (p. 1032).

**5.294.2.3 op\_setval()** std::string MHAParser::c\_ifc\_parser\_t::op\_setval (MHAParser::expression\_t & x ) [protected], [virtual]

Reimplemented from **MHAParser::base\_t** (p. 1033).

**5.294.2.4 `op_query()`** std::string MHAParser::c\_ifc\_parser\_t::op\_query (   
    **MHAParser::expression\_t** & *x* ) [protected], [virtual]

Reimplemented from **MHAParser::base\_t** (p. 1033).

**5.294.2.5 `test_error()`** void MHAParser::c\_ifc\_parser\_t::test\_error ( ) [private]

### 5.294.3 Member Data Documentation

**5.294.3.1 `modulename`** std::string MHAParser::c\_ifc\_parser\_t::modulename [private]

**5.294.3.2 `c_parse_cmd`** **c\_parse\_cmd\_t** MHAParser::c\_ifc\_parser\_t::c\_parse\_cmd [private]

**5.294.3.3 `c_parse_err`** **c\_parse\_err\_t** MHAParser::c\_ifc\_parser\_t::c\_parse\_err [private]

**5.294.3.4 `liberr`** int MHAParser::c\_ifc\_parser\_t::liberr [private]

**5.294.3.5 `libdata`** void\* MHAParser::c\_ifc\_parser\_t::libdata [private]

**5.294.3.6 `ret_size`** unsigned int MHAParser::c\_ifc\_parser\_t::ret\_size [private]

**5.294.3.7 `retv` char\* MHAParser::c\_ifc\_parser\_t::retv [private]**

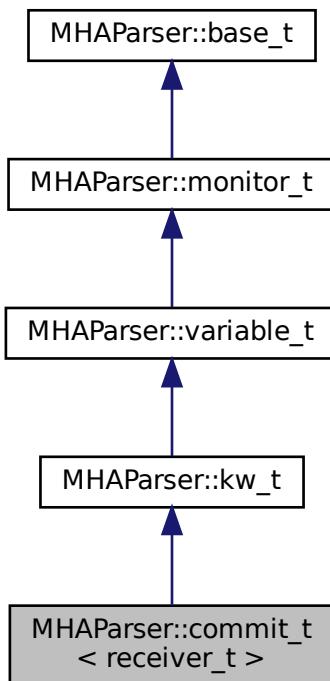
The documentation for this class was generated from the following files:

- `mha_parser.hh`
- `mha_parser.cpp`

**5.295 MHAParser::commit\_t< receiver\_t > Class Template Reference**

Parser variable with event-emission functionality.

Inheritance diagram for MHAParser::commit\_t< receiver\_t >:



**Public Member Functions**

- `commit_t (receiver_t *, void(receiver_t::*)(), const std::string & help="Variable changes action")`

## Private Attributes

- **MHAEVENTS::CONNECTOR\_T< receiver\_t > extern\_connector**

## Additional Inherited Members

### 5.295.1 Detailed Description

```
template<class receiver_t>
class MHAParser::commit_t< receiver_t >
```

Parser variable with event-emission functionality.

The **commit\_t** (p. 1049) variable can register an event receiver in its constructor, which is called whenever the variable is set to "commit".

### 5.295.2 Constructor & Destructor Documentation

```
5.295.2.1 commit_t() template<class receiver_t >
MHAParser::commit_t< receiver_t >:: commit_t (
    receiver_t * r,
    void(receiver_t::* )() rfun,
    const std::string & help = "Variable changes action" )
```

### 5.295.3 Member Data Documentation

```
5.295.3.1 extern_connector template<class receiver_t >
MHAEVENTS::CONNECTOR_T<receiver_t> MHAParser::commit_t< receiver_t >::extern_<-
connector [private]
```

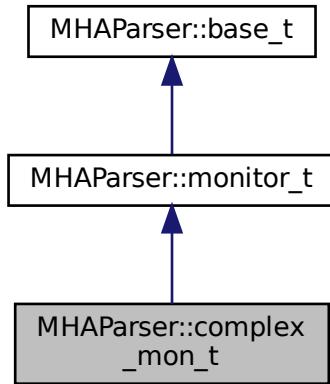
The documentation for this class was generated from the following file:

- **mha\_parser.hh**

## 5.296 MHAParser::complex\_mon\_t Class Reference

Monitor with complex value.

Inheritance diagram for MHAParser::complex\_mon\_t:



### Public Member Functions

- **complex\_mon\_t** (const std::string &hlp)  
*Create a complex monitor variable.*

### Public Attributes

- **mha\_complex\_t data**  
*Data field.*

### Protected Member Functions

- std::string **query\_val** (const std::string &)
- std::string **query\_type** (const std::string &)

### Additional Inherited Members

#### 5.296.1 Detailed Description

Monitor with complex value.

## 5.296.2 Constructor & Destructor Documentation

**5.296.2.1 `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.
------------------	---

## 5.296.3 Member Function Documentation

**5.296.3.1 `query_val()`** `std::string MHAParser::complex_mon_t::query_val ( const std::string & s ) [protected], [virtual]`

Reimplemented from **MHAParser::base\_t** (p. 1034).

**5.296.3.2 `query_type()`** `std::string MHAParser::complex_mon_t::query_type ( const std::string & ) [protected], [virtual]`

Reimplemented from **MHAParser::base\_t** (p. 1034).

## 5.296.4 Member Data Documentation

**5.296.4.1 `data mha_complex_t`** `MHAParser::complex_mon_t::data`

Data field.

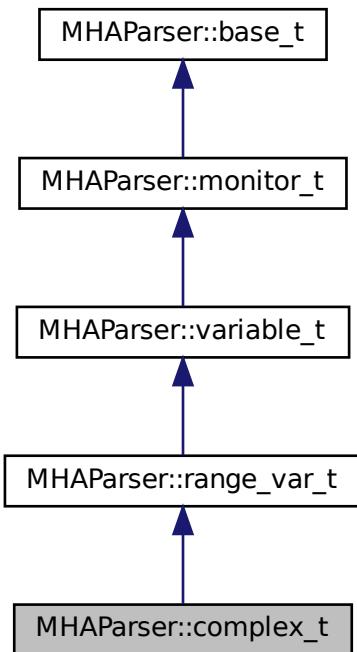
The documentation for this class was generated from the following files:

- **mha\_parser.hh**
- **mha\_parser.cpp**

## 5.297 MHAParser::complex\_t Class Reference

Variable with complex value.

Inheritance diagram for MHAParser::complex\_t:



### Public Member Functions

- **complex\_t** (const std::string &, const std::string &, const std::string &=""")

### Public Attributes

- **mha\_complex\_t data**  
*Data field.*

### Protected Member Functions

- std::string **op\_setval** ( expression\_t &)
- std::string **query\_type** (const std::string &)
- std::string **query\_val** (const std::string &)

## Additional Inherited Members

### 5.297.1 Detailed Description

Variable with complex value.

### 5.297.2 Constructor & Destructor Documentation

```
5.297.2.1 complex_t() MHAParser::complex_t::complex_t (
    const std::string & h,
    const std::string & v,
    const std::string & rg = "")
```

### 5.297.3 Member Function Documentation

```
5.297.3.1 op_setval() std::string MHAParser::complex_t::op_setval (
    expression_t & x) [protected], [virtual]
```

Reimplemented from **MHAParser::variable\_t** (p. 1115).

```
5.297.3.2 query_type() std::string MHAParser::complex_t::query_type (
    const std::string &) [protected], [virtual]
```

Reimplemented from **MHAParser::base\_t** (p. 1034).

```
5.297.3.3 query_val() std::string MHAParser::complex_t::query_val (
    const std::string & s) [protected], [virtual]
```

Reimplemented from **MHAParser::base\_t** (p. 1034).

## 5.297.4 Member Data Documentation

### 5.297.4.1 **data** mha\_complex\_t MHAParser::complex\_t::data

Data field.

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

- **mha\_parser.hh**
- **mha\_parser.cpp**

## 5.298 MHAParser::entry\_t Class Reference

### Public Member Functions

- **entry\_t** (const std::string &, **base\_t** \*)

### Public Attributes

- std::string **name**
- **base\_t** \* **entry**

## 5.298.1 Constructor & Destructor Documentation

### 5.298.1.1 **entry\_t()** MHAParser::entry\_t::entry\_t (

```
const std::string & n,
base_t * e )
```

## 5.298.2 Member Data Documentation

**5.298.2.1 name** std::string MHAParser::entry\_t::name

**5.298.2.2 entry** base\_t\* MHAParser::entry\_t::entry

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

- **mha\_parser.hh**
- **mha\_parser.cpp**

## 5.299 MHAParser::expression\_t Class Reference

### Public Member Functions

- **expression\_t** (const std::string &, const std::string &)  
*Constructor.*
- **expression\_t** ()

### Public Attributes

- std::string **lval**
- std::string **rval**
- std::string **op**

### 5.299.1 Constructor & Destructor Documentation

**5.299.1.1 expression\_t() [1/2]** expression\_t::expression\_t (

```
const std::string & s,
const std::string & o )
```

Constructor.

#### Parameters

<b>s</b>	String to be splitted
<b>o</b>	List of valid operators (single character only)

**5.299.1.2 expression\_t() [2/2]** expression\_t::expression\_t ( )**5.299.2 Member Data Documentation****5.299.2.1 lval** std::string MHAParser::expression\_t::lval**5.299.2.2 rval** std::string MHAParser::expression\_t::rval**5.299.2.3 op** std::string MHAParser::expression\_t::op

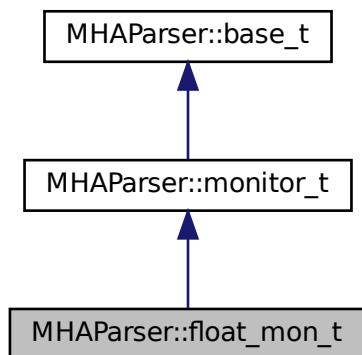
The documentation for this class was generated from the following files:

- **mha\_parser.hh**
- **mha\_parser.cpp**

**5.300 MHAParser::float\_mon\_t Class Reference**

Monitor with float value.

Inheritance diagram for MHAParser::float\_mon\_t:



## Public Member Functions

- **float\_mon\_t** (const std::string &hlp)  
*Initialize a floating point (32 bits) monitor variable.*

## Public Attributes

- float **data**  
*Data field.*

## Protected Member Functions

- std::string **query\_val** (const std::string &)
- std::string **query\_type** (const std::string &)

## Additional Inherited Members

### 5.300.1 Detailed Description

Monitor with float value.

### 5.300.2 Constructor & Destructor Documentation

#### 5.300.2.1 **float\_mon\_t()** MHAParser::float\_mon\_t::float\_mon\_t ( const std::string & hlp )

Initialize a floating point (32 bits) monitor variable.

#### Parameters

<i>hlp</i>	A help text describing this monitor variable.
------------	---

### 5.300.3 Member Function Documentation

**5.300.3.1 query\_val()** std::string MHAParser::float\_mon\_t::query\_val ( const std::string & s ) [protected], [virtual]

Reimplemented from **MHAParser::base\_t** (p. 1034).

**5.300.3.2 query\_type()** std::string MHAParser::float\_mon\_t::query\_type ( const std::string & ) [protected], [virtual]

Reimplemented from **MHAParser::base\_t** (p. 1034).

#### 5.300.4 Member Data Documentation

**5.300.4.1 data** float MHAParser::float\_mon\_t::data

Data field.

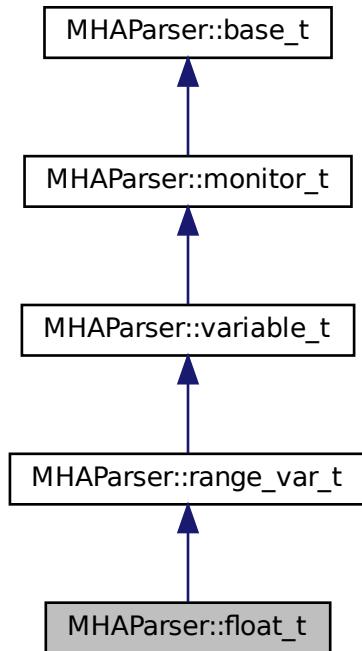
The documentation for this class was generated from the following files:

- **mha\_parser.hh**
- **mha\_parser.cpp**

#### 5.301 MHAParser::float\_t Class Reference

Variable with float value.

Inheritance diagram for MHAParser::float\_t:



## Public Member Functions

- **float\_t** (const std::string &help\_text, const std::string &initial\_value, const std::string &range="")

*Constructor for a configuration language variable for 32bit ieee floating-point values.*

## Public Attributes

- float **data**

*Data field.*

## Protected Member Functions

- std::string **op\_setval** ( expression\_t &)
- std::string **query\_type** (const std::string &)
- std::string **query\_val** (const std::string &)

## Additional Inherited Members

### 5.301.1 Detailed Description

Variable with float value.

### 5.301.2 Constructor & Destructor Documentation

```
5.301.2.1 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.

### 5.301.3 Member Function Documentation

```
5.301.3.1 op_setval() std::string MHAParser::float_t::op_setval (
    expression_t & x ) [protected], [virtual]
```

Reimplemented from **MHAParser::variable\_t** (p. 1115).

**5.301.3.2 `query_type()`** `std::string MHAParser::float_t::query_type (`  
`const std::string & ) [protected], [virtual]`

Reimplemented from **MHAParser::base\_t** (p. 1034).

**5.301.3.3 `query_val()`** `std::string MHAParser::float_t::query_val (`  
`const std::string & s ) [protected], [virtual]`

Reimplemented from **MHAParser::base\_t** (p. 1034).

#### 5.301.4 Member Data Documentation

**5.301.4.1 `data`** `float MHAParser::float_t::data`

Data field.

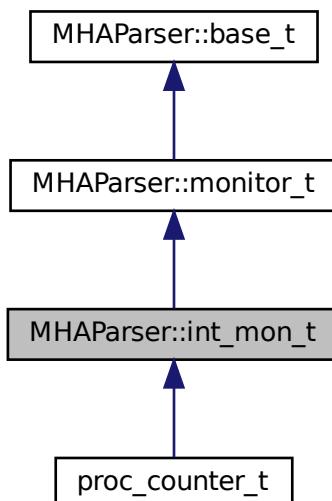
The documentation for this class was generated from the following files:

- `mha_parser.hh`
- `mha_parser.cpp`

### 5.302 MHAParser::int\_mon\_t Class Reference

Monitor variable with int value.

Inheritance diagram for MHAParser::int\_mon\_t:



## Public Member Functions

- **int\_mon\_t** (const std::string &hlp)  
*Create a monitor variable for integral values.*

## Public Attributes

- int **data**  
*Data field.*

## Protected Member Functions

- std::string **query\_val** (const std::string &)
- std::string **query\_type** (const std::string &)

## Additional Inherited Members

### 5.302.1 Detailed Description

Monitor variable with int value.

Monitor variables can be of many types. These variables can be queried through the parser. The public data element contains the monitored state. Write access is only possible from the C++ code by direct access to the data field.

### 5.302.2 Constructor & Destructor Documentation

#### 5.302.2.1 **int\_mon\_t()** MHParse::int\_mon\_t::int\_mon\_t ( const std::string & hlp )

Create a monitor variable for integral values.

##### Parameters

<i>hlp</i>	A help text describing this monitor variable.
------------	---

### 5.302.3 Member Function Documentation

**5.302.3.1 `query_val()`** std::string MHAParser::int\_mon\_t::query\_val (const std::string & s) [protected], [virtual]

Reimplemented from **MHAParser::base\_t** (p. [1034](#)).

**5.302.3.2 `query_type()`** std::string MHAParser::int\_mon\_t::query\_type (const std::string &) [protected], [virtual]

Reimplemented from **MHAParser::base\_t** (p. [1034](#)).

### 5.302.4 Member Data Documentation

**5.302.4.1 `data`** int MHAParser::int\_mon\_t::data

Data field.

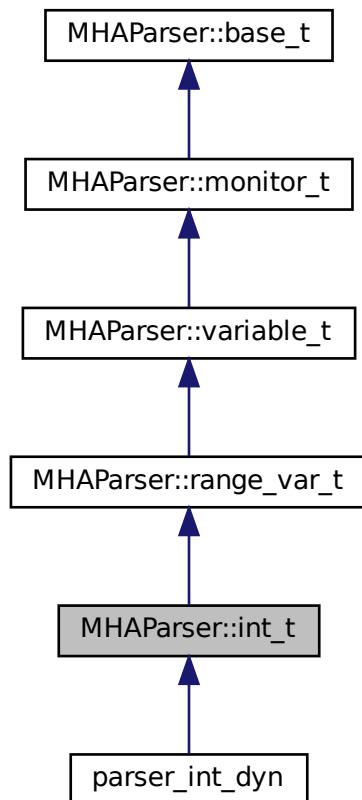
The documentation for this class was generated from the following files:

- [mha\\_parser.hh](#)
- [mha\\_parser.cpp](#)

## 5.303 MHParse::int\_t Class Reference

Variable with integer value.

Inheritance diagram for MHParse::int\_t:



### Public Member Functions

- **int\_t** (const std::string &help\_text, const std::string &initial\_value, const std::string &range="")  
*Constructor for a configuration language variable for integral values.*

### Public Attributes

- int **data**  
*Data field.*

## Protected Member Functions

- std::string **op\_setval** ( expression\_t &)
- std::string **query\_type** (const std::string &)
- std::string **query\_val** (const std::string &)

## Additional Inherited Members

### 5.303.1 Detailed Description

Variable with integer value.

### 5.303.2 Constructor & Destructor Documentation

**5.303.2.1 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]).

### 5.303.3 Member Function Documentation

**5.303.3.1 op\_setval()** std::string MHAParser::int\_t::op\_setval ( expression\_t & x ) [protected], [virtual]

Reimplemented from **MHAParser::variable\_t** (p. 1115).

**5.303.3.2 query\_type()** std::string MHAParser::int\_t::query\_type ( const std::string & ) [protected], [virtual]

Reimplemented from **MHAParser::base\_t** (p. 1034).

**5.303.3.3 query\_val()** std::string MHAParser::int\_t::query\_val ( const std::string & s ) [protected], [virtual]

Reimplemented from **MHAParser::base\_t** (p. 1034).

## 5.303.4 Member Data Documentation

**5.303.4.1 data** int MHAParser::int\_t::data

Data field.

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

- **mha\_parser.hh**
- **mha\_parser.cpp**

## 5.304 MHAParser::keyword\_list\_t Class Reference

Keyword list class.

### Public Types

- `typedef std::vector< std::string >::size_type size_t`

## Public Member Functions

- void **set\_value** (const std::string &)
 

*Select a value from keyword list.*
- void **set\_entries** (const std::string &)
 

*Set keyword list entries.*
- const std::string & **get\_value** () const
 

*Return selected value.*
- const std::vector< std::string > & **get\_entries** () const
 

*Return keyword list.*
- const **size\_t** & **get\_index** () const
 

*Return index of selected value.*
- void **set\_index** (unsigned int)
- void **validate** () const
 

*Check if index of selected value is valid.*
- void **add\_entry** (const std::string &en)
- **keyword\_list\_t** ()
 

*Constructor.*

## Private Attributes

- **size\_t index**

*Index into list.*
- std::vector< std::string > **entries**

*List of valid entries.*
- std::string **empty\_string**

### 5.304.1 Detailed Description

Keyword list class.

The structure **keyword\_list\_t** (p. 1067) defines a keyword list (vector of strings) with an index into the list. Used as **MHAParser::kw\_t** (p. 1071), it can be used to access a set of valid keywords through the parser (i.e. one of "pear apple banana").

### 5.304.2 Member Typedef Documentation

#### 5.304.2.1 **size\_t** `typedef std::vector<std::string>::size_type MHAParser::keyword_list_t::size_t`

### 5.304.3 Constructor & Destructor Documentation

#### 5.304.3.1 keyword\_list\_t() MHParse::keyword\_list\_t::keyword\_list\_t ( )

Constructor.

### 5.304.4 Member Function Documentation

#### 5.304.4.1 set\_value() void MHParse::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

<i>s</i>	Value to be selected.
----------	-----------------------

#### 5.304.4.2 set\_entries() void MHParse::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

<i>s</i>	Space separated entry list.
----------	-----------------------------

#### 5.304.4.3 get\_value() const std::string & MHParse::keyword\_list\_t::get\_value ( )

const

Return selected value.

**5.304.4.4 get\_entries()** const std::vector< std::string > & MHAParser::keyword\_list\_t::get\_entries ( ) const

Return keyword list.

**5.304.4.5 get\_index()** const MHAParser::keyword\_list\_t::size\_t & MHAParser::keyword\_list\_t::get\_index ( ) const

Return index of selected value.

**5.304.4.6 set\_index()** void MHAParser::keyword\_list\_t::set\_index ( unsigned int idx )

**5.304.4.7 validate()** void MHAParser::keyword\_list\_t::validate ( ) const

Check if index of selected value is valid.

**5.304.4.8 add\_entry()** void MHAParser::keyword\_list\_t::add\_entry ( const std::string & en ) [inline]

## 5.304.5 Member Data Documentation

**5.304.5.1 index size\_t** MHAParser::keyword\_list\_t::index [private]

Index into list.

**5.304.5.2 entries std::vector<std::string>** MHAParser::keyword\_list\_t::entries [private]

List of valid entries.

**5.304.5.3 empty\_string std::string** MHAParser::keyword\_list\_t::empty\_string [private]

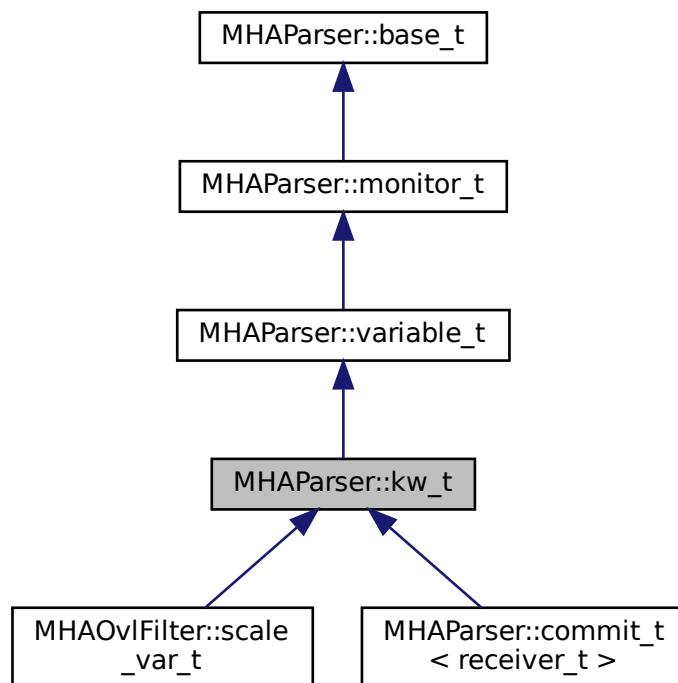
The documentation for this class was generated from the following files:

- **mha\_parser.hh**
- **mha\_parser.cpp**

**5.305 MHAParser::kw\_t Class Reference**

Variable with keyword list value.

Inheritance diagram for MHAParser::kw\_t:



## Public Member Functions

- **kw\_t** (const std::string &, const std::string &, const std::string &)  
*Constructor of a keyword list openMHA configuration variable.*
- **kw\_t** (const **kw\_t** &)  
*Copy constructor.*
- **void set\_range** (const std::string &)  
*Set/change the list of valid entries.*
- **bool isval** (const std::string &) const  
*Test if the given value is selected.*

## Public Attributes

- **keyword\_list\_t data**  
*Variable data in its native type.*

## Protected Member Functions

- **void validate** (const **keyword\_list\_t** &)
- **std::string op\_setval** ( **expression\_t** &)
- **std::string query\_range** (const std::string &)
- **std::string query\_val** (const std::string &)
- **std::string query\_type** (const std::string &)

## Additional Inherited Members

### 5.305.1 Detailed Description

Variable with keyword list value.

### 5.305.2 Constructor & Destructor Documentation

#### 5.305.2.1 **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 "[", "]".

**5.305.2.2 kw\_t() [2/2]** `MHAParser::kw_t::kw_t (`  
`const kw_t & src )`

Copy constructor.

### 5.305.3 Member Function Documentation

**5.305.3.1 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 "[", "]".
----------	--

**5.305.3.2 isval()** `bool MHAParser::kw_t::isval (`  
`const std::string & testval ) const`

Test if the given value is selected.

**5.305.3.3 validate()** `void MHAParser::kw_t::validate (`  
`const keyword_list_t & s ) [protected]`

**5.305.3.4 `op_setval()`** `std::string MHAParser::kw_t::op_setval ( expression_t & x )` [protected], [virtual]

Reimplemented from **MHAParser::variable\_t** (p. 1115).

**5.305.3.5 `query_range()`** `std::string MHAParser::kw_t::query_range ( const std::string & )` [protected], [virtual]

Reimplemented from **MHAParser::base\_t** (p. 1034).

**5.305.3.6 `query_val()`** `std::string MHAParser::kw_t::query_val ( const std::string & s )` [protected], [virtual]

Reimplemented from **MHAParser::base\_t** (p. 1034).

**5.305.3.7 `query_type()`** `std::string MHAParser::kw_t::query_type ( const std::string & )` [protected], [virtual]

Reimplemented from **MHAParser::base\_t** (p. 1034).

## 5.305.4 Member Data Documentation

**5.305.4.1 `data keyword_list_t`** `MHAParser::kw_t::data`

Variable data in its native type.

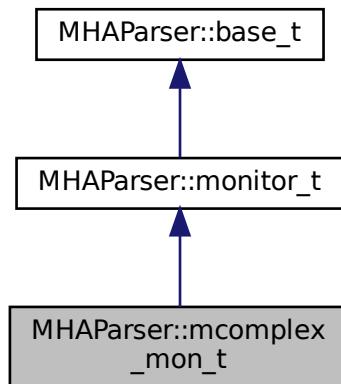
The documentation for this class was generated from the following files:

- **mha\_parser.hh**
- **mha\_parser.cpp**

## 5.306 MHAParser::mcomplex\_mon\_t Class Reference

Matrix of complex numbers monitor.

Inheritance diagram for MHAParser::mcomplex\_mon\_t:



### Public Member Functions

- **mcomplex\_mon\_t** (const std::string &hlp)  
*Create a matrix of complex floating point monitor values.*

### Public Attributes

- std::vector< std::vector< **mha\_complex\_t** > > **data**  
*Data field.*

### Protected Member Functions

- std::string **query\_val** (const std::string &)
- std::string **query\_type** (const std::string &)

### Additional Inherited Members

#### 5.306.1 Detailed Description

Matrix of complex numbers monitor.

## 5.306.2 Constructor & Destructor Documentation

**5.306.2.1 `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.
------------------	---

## 5.306.3 Member Function Documentation

**5.306.3.1 `query_val()`** `std::string MHAParser::mcomplex_mon_t::query_val ( const std::string & s ) [protected], [virtual]`

Reimplemented from **MHAParser::base\_t** (p. 1034).

**5.306.3.2 `query_type()`** `std::string MHAParser::mcomplex_mon_t::query_type ( const std::string & ) [protected], [virtual]`

Reimplemented from **MHAParser::base\_t** (p. 1034).

## 5.306.4 Member Data Documentation

**5.306.4.1 `data`** `std::vector< std::vector< mha_complex_t> > MHAParser::mcomplex_mon_t::data`

Data field.

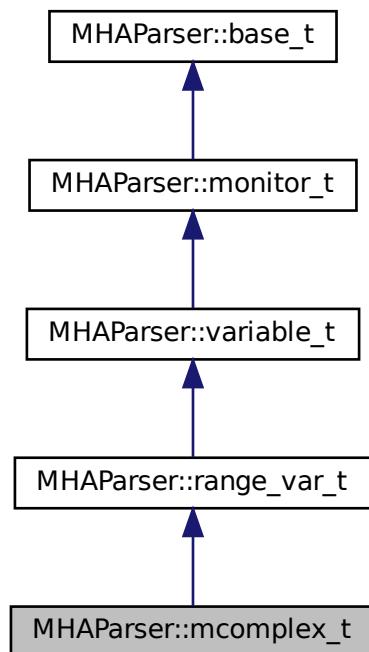
The documentation for this class was generated from the following files:

- **mha\_parser.hh**
- **mha\_parser.cpp**

## 5.307 MHAParser::mcomplex\_t Class Reference

Matrix variable with complex value.

Inheritance diagram for MHAParser::mcomplex\_t:



### Public Member Functions

- **mcomplex\_t** (const std::string &, const std::string &, const std::string &=""")

### Public Attributes

- std::vector< std::vector< **mha\_complex\_t** > > **data**  
*Data field.*

### Protected Member Functions

- std::string **op\_setval** ( **expression\_t** &)
- std::string **query\_type** (const std::string &)
- std::string **query\_val** (const std::string &)

## Additional Inherited Members

### 5.307.1 Detailed Description

Matrix variable with complex value.

### 5.307.2 Constructor & Destructor Documentation

**5.307.2.1 `mcomplex_t()`** `MHAParser::mcomplex_t::mcomplex_t (`  
    `const std::string & h,`  
    `const std::string & v,`  
    `const std::string & rg = "" )`

### 5.307.3 Member Function Documentation

**5.307.3.1 `op_setval()`** `std::string MHAParser::mcomplex_t::op_setval (`  
    `expression_t & x ) [protected], [virtual]`

Reimplemented from **MHAParser::variable\_t** (p. 1115).

**5.307.3.2 `query_type()`** `std::string MHAParser::mcomplex_t::query_type (`  
    `const std::string & ) [protected], [virtual]`

Reimplemented from **MHAParser::base\_t** (p. 1034).

**5.307.3.3 `query_val()`** `std::string MHAParser::mcomplex_t::query_val (`  
    `const std::string & s ) [protected], [virtual]`

Reimplemented from **MHAParser::base\_t** (p. 1034).

### 5.307.4 Member Data Documentation

**5.307.4.1 data** std::vector<std::vector<mha\_complex\_t>> MHAParser::mcomplex\_t<->::data

Data field.

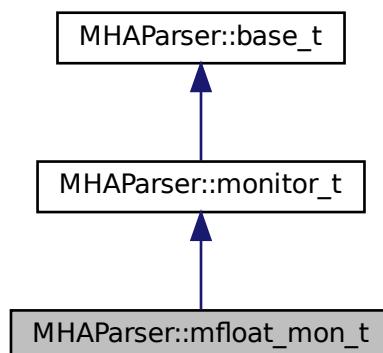
The documentation for this class was generated from the following files:

- [mha\\_parser.hh](#)
- [mha\\_parser.cpp](#)

## 5.308 MHAParser::mfloat\_mon\_t Class Reference

Matrix of floats monitor.

Inheritance diagram for MHAParser::mfloat\_mon\_t:



### Public Member Functions

- **mfloat\_mon\_t** (const std::string &hlp)  
*Create a matrix of floating point monitor values.*

## Public Attributes

- std::vector< std::vector< float > > **data**  
*Data field.*

## Protected Member Functions

- std::string **query\_val** (const std::string &)
- std::string **query\_type** (const std::string &)

## Additional Inherited Members

### 5.308.1 Detailed Description

Matrix of floats monitor.

### 5.308.2 Constructor & Destructor Documentation

#### 5.308.2.1 **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.
------------	---

### 5.308.3 Member Function Documentation

#### 5.308.3.1 **query\_val()** std::string MHAParser::mfloat\_mon\_t::query\_val ( const std::string & *s* ) [protected], [virtual]

Reimplemented from **MHAParser::base\_t** (p. 1034).

**5.308.3.2 query\_type()** std::string MHAParser::mfloat\_mon\_t::query\_type ( const std::string & ) [protected], [virtual]

Reimplemented from **MHAParser::base\_t** (p. 1034).

#### 5.308.4 Member Data Documentation

**5.308.4.1 data** std::vector< std::vector<float> > MHAParser::mfloat\_mon\_t::data

Data field.

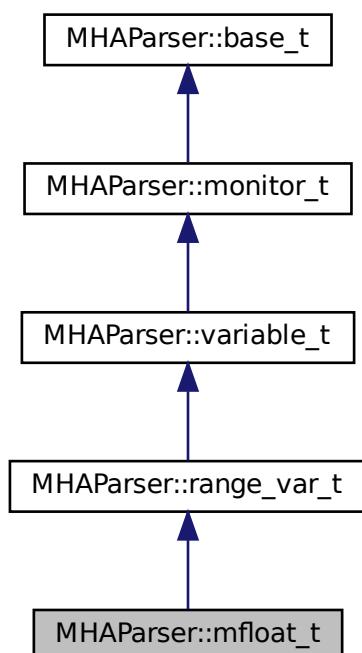
The documentation for this class was generated from the following files:

- **mha\_parser.hh**
- **mha\_parser.cpp**

## 5.309 MHAParser::mfloat\_t Class Reference

Matrix variable with float value.

Inheritance diagram for MHAParser::mfloat\_t:



## Public Member Functions

- **mfloat\_t** (const std::string &, const std::string &, const std::string &="")
   
*Create a float matrix parser variable.*

## Public Attributes

- std::vector< std::vector< float > > **data**
  
*Data field.*

## Protected Member Functions

- std::string **op\_setval** ( **expression\_t** &)
- std::string **query\_type** (const std::string &)
- std::string **query\_val** (const std::string &)

## Additional Inherited Members

### 5.309.1 Detailed Description

Matrix variable with float value.

### 5.309.2 Constructor & Destructor Documentation

```
5.309.2.1 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.

### 5.309.3 Member Function Documentation

**5.309.3.1 `op_setval()`** std::string MHAParser::mfloat\_t::op\_setval ( expression\_t & x ) [protected], [virtual]

Reimplemented from **MHAParser::variable\_t** (p. 1115).

**5.309.3.2 `query_type()`** std::string MHAParser::mfloat\_t::query\_type ( const std::string & ) [protected], [virtual]

Reimplemented from **MHAParser::base\_t** (p. 1034).

**5.309.3.3 `query_val()`** std::string MHAParser::mfloat\_t::query\_val ( const std::string & s ) [protected], [virtual]

Reimplemented from **MHAParser::base\_t** (p. 1034).

### 5.309.4 Member Data Documentation

**5.309.4.1 `data`** std::vector<std::vector<float>> MHAParser::mfloat\_t::data

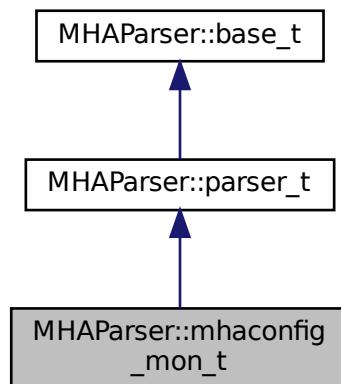
Data field.

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

- [mha\\_parser.hh](#)
- [mha\\_parser.cpp](#)

### 5.310 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

### 5.310.1 Constructor & Destructor Documentation

**5.310.1.1 mhaconfig\_mon\_t()** MHAParser::mhaconfig\_mon\_t::mhaconfig\_mon\_t (const std::string & help = "")

### 5.310.2 Member Function Documentation

**5.310.2.1 update()** void MHAParser::mhaconfig\_mon\_t::update (const mhaconfig\_t & cf)

### 5.310.3 Member Data Documentation

**5.310.3.1 channels** MHAParser::int\_mon\_t MHAParser::mhaconfig\_mon\_t::channels [private]

Number of audio channels.

**5.310.3.2 domain** MHAParser::string\_mon\_t MHAParser::mhaconfig\_mon\_t::domain [private]

Signal domain (MHA\_WAVEFORM or MHA\_SPECTRUM)

**5.310.3.3 fragsize** MHAParser::int\_mon\_t MHAParser::mhaconfig\_mon\_t::fragsize [private]

Fragment size of waveform data.

**5.310.3.4 wndlen** `MHAParser::int_mon_t` `MHAParser::mhaconfig_mon_t::wndlen` [private]

Window length of spectral data.

**5.310.3.5 fftlen** `MHAParser::int_mon_t` `MHAParser::mhaconfig_mon_t::fftlens` [private]

FFT length of spectral data.

**5.310.3.6 srate** `MHAParser::float_mon_t` `MHAParser::mhaconfig_mon_t::srate` [private]

Sampling rate in Hz.

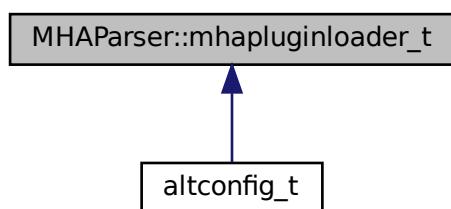
The documentation for this class was generated from the following files:

- `mha_parser.hh`
- `mha_parser.cpp`

**5.311 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 ( MHParse::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

- `MHParse::parser_t & parent_`
- `MHParse::string_t plugname`
- `std::string prefix_`
- `MHAEvents::connector_t< mhapluginloader_t > connector`
- `algo_comm_t ac_`
- `std::string last_name`
- `std::string plugname_name_`
- `mhaconfig_t cf_in_`
- `mhaconfig_t cf_out_`

## Static Private Attributes

- `static double bookkeeping`

### 5.311.1 Detailed Description

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

## 5.311.2 Constructor & Destructor Documentation

**5.311.2.1 `mhapluginloader_t()`** MHAParser::mhapluginloader\_t::mhapluginloader\_t (

```
MHAParser::parser_t & parent,  
const algo_comm_t & ac,  
const std::string & plugname_name = "plugin_name",  
const std::string & prefix = "")
```

**5.311.2.2 `~mhapluginloader_t()`** MHAParser::mhapluginloader\_t::~mhapluginloader\_t ( )

## 5.311.3 Member Function Documentation

**5.311.3.1 `prepare()`** void MHAParser::mhapluginloader\_t::prepare (

```
mhaconfig_t & cf )
```

**5.311.3.2 `release()`** void MHAParser::mhapluginloader\_t::release ( )

**5.311.3.3 `process() [1/4]`** void MHAParser::mhapluginloader\_t::process (

```
mha_wave_t * sIn,  
mha_wave_t ** sOut ) [inline]
```

**5.311.3.4 `process() [2/4]`** void MHAParser::mhapluginloader\_t::process (

```
mha_spec_t * sIn,  
mha_spec_t ** sOut ) [inline]
```

**5.311.3.5 process() [3/4]** void MHAParser::mhaplugloader\_t::process (

```
mha_wave_t * sIn,
mha_spec_t ** sOut ) [inline]
```

**5.311.3.6 process() [4/4]** void MHAParser::mhaplugloader\_t::process (

```
mha_spec_t * sIn,
mha_wave_t ** sOut ) [inline]
```

**5.311.3.7 get\_cfin()** mhaconfig\_t MHAParser::mhaplugloader\_t::get\_cfin ( ) const [inline]

**5.311.3.8 get\_cfout()** mhaconfig\_t MHAParser::mhaplugloader\_t::get\_cfout ( ) const [inline]

**5.311.3.9 get\_last\_name()** const std::string& MHAParser::mhaplugloader\_t::get\_last\_name ( ) const [inline]

**5.311.3.10 load\_plug()** void MHAParser::mhaplugloader\_t::load\_plug ( ) [private]

## 5.311.4 Member Data Documentation

**5.311.4.1 plug** PluginLoader::mhaplugloader\_t\* MHAParser::mhaplugloader\_t::plug [protected]

**5.311.4.2 parent\_** `MHAParser::parser_t&` `MHAParser::mhapluginloader_t::parent_` ↵  
[private]

**5.311.4.3 plugname** `MHAParser::string_t` `MHAParser::mhapluginloader_t::plugname`  
[private]

**5.311.4.4 prefix\_** `std::string` `MHAParser::mhapluginloader_t::prefix_` [private]

**5.311.4.5 connector** `MHAEVENTS::connector_t< mhapluginloader_t>` `MHAParser::mhapluginloader_` ↵  
`_t::connector` [private]

**5.311.4.6 ac\_** `algo_comm_t` `MHAParser::mhapluginloader_t::ac_` [private]

**5.311.4.7 last\_name** `std::string` `MHAParser::mhapluginloader_t::last_name` [private]

**5.311.4.8 plugname\_name\_** `std::string` `MHAParser::mhapluginloader_t::plugname_` ↵  
`name_` [private]

**5.311.4.9 cf\_in\_** `mhaconfig_t` `MHAParser::mhapluginloader_t::cf_in_` [private]

**5.311.4.10 cf\_out\_** `mhaconfig_t` `MHAParser::mhapluginloader_t::cf_out_` [private]

**5.311.4.11 bookkeeping** double MHAParser::mhapluginloader\_t::bookkeeping [static],  
[private]

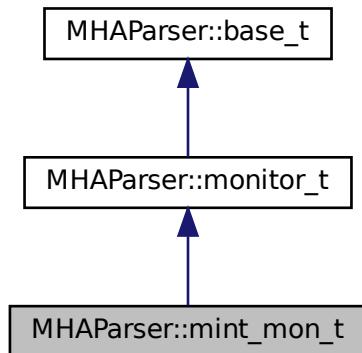
The documentation for this class was generated from the following files:

- **mhapluginloader.h**
- **mhapluginloader.cpp**

## 5.312 MHAParser::mint\_mon\_t Class Reference

Matrix of ints monitor.

Inheritance diagram for MHAParser::mint\_mon\_t:



### Public Member Functions

- **mint\_mon\_t (const std::string &hlp)**  
*Create a matrix of integer monitor values.*

### Public Attributes

- std::vector< std::vector< int > > **data**  
*Data field.*

## Protected Member Functions

- std::string **query\_val** (const std::string &)
- std::string **query\_type** (const std::string &)

## Additional Inherited Members

### 5.312.1 Detailed Description

Matrix of ints monitor.

### 5.312.2 Constructor & Destructor Documentation

**5.312.2.1 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.
------------	---

### 5.312.3 Member Function Documentation

**5.312.3.1 query\_val()** std::string MHAParser::mint\_mon\_t::query\_val (const std::string & *s*) [protected], [virtual]

Reimplemented from **MHAParser::base\_t** (p. 1034).

**5.312.3.2 query\_type()** std::string MHAParser::mint\_mon\_t::query\_type (const std::string & ) [protected], [virtual]

Reimplemented from **MHAParser::base\_t** (p. 1034).

### 5.312.4 Member Data Documentation

#### 5.312.4.1 **data** std::vector< std::vector<int> > MHAParser::mint\_mon\_t::data

Data field.

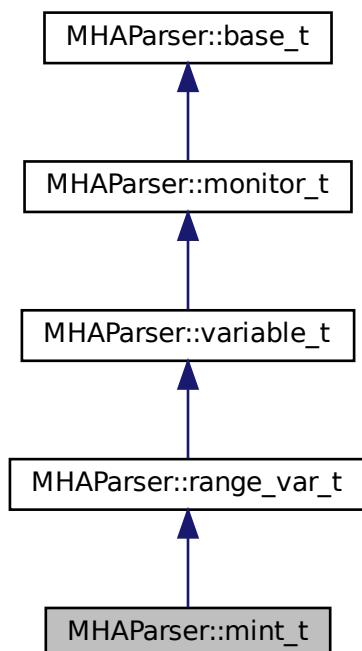
The documentation for this class was generated from the following files:

- [mha\\_parser.hh](#)
- [mha\\_parser.cpp](#)

### 5.313 MHAParser::mint\_t Class Reference

Matrix variable with int value.

Inheritance diagram for MHAParser::mint\_t:



## Public Member Functions

- **mint\_t** (const std::string &, const std::string &, const std::string &="")
   
*Create a int matrix parser variable.*

## Public Attributes

- std::vector< std::vector< int > > **data**
  
*Data field.*

## Protected Member Functions

- std::string **op\_setval** ( **expression\_t** &)
- std::string **query\_type** (const std::string &)
- std::string **query\_val** (const std::string &)

## Additional Inherited Members

### 5.313.1 Detailed Description

Matrix variable with int value.

### 5.313.2 Constructor & Destructor Documentation

```
5.313.2.1 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.

### 5.313.3 Member Function Documentation

**5.313.3.1 op\_setval()** std::string MHAParser::mint\_t::op\_setval ( expression\_t & x ) [protected], [virtual]

Reimplemented from **MHAParser::variable\_t** (p. 1115).

**5.313.3.2 query\_type()** std::string MHAParser::mint\_t::query\_type ( const std::string & ) [protected], [virtual]

Reimplemented from **MHAParser::base\_t** (p. 1034).

**5.313.3.3 query\_val()** std::string MHAParser::mint\_t::query\_val ( const std::string & s ) [protected], [virtual]

Reimplemented from **MHAParser::base\_t** (p. 1034).

### 5.313.4 Member Data Documentation

**5.313.4.1 data** std::vector<std::vector<int> > MHAParser::mint\_t::data

Data field.

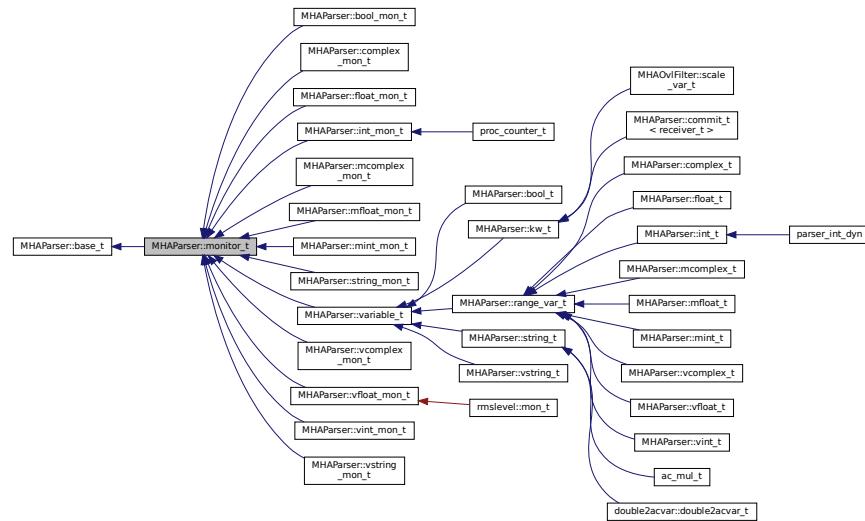
The documentation for this class was generated from the following files:

- **mha\_parser.hh**
- **mha\_parser.cpp**

## 5.314 MHParse::monitor\_t Class Reference

Base class for monitors and variable nodes.

Inheritance diagram for MHParse::monitor\_t:



### Public Member Functions

- `monitor_t (const std::string &)`
- `monitor_t (const monitor_t &)`
- `monitor_t & operator= (const monitor_t &) = default`
- `std::string op_query (expression_t &)`
- `std::string query_dump (const std::string &)`
- `std::string query_perm (const std::string &)`

### Additional Inherited Members

#### 5.314.1 Detailed Description

Base class for monitors and variable nodes.

#### 5.314.2 Constructor & Destructor Documentation

**5.314.2.1 monitor\_t() [1/2]** MHAParser::monitor\_t::monitor\_t (

```
const std::string & h )
```

**5.314.2.2 monitor\_t() [2/2]** MHAParser::monitor\_t::monitor\_t (

```
const monitor_t & src )
```

### 5.314.3 Member Function Documentation

**5.314.3.1 operator=()** monitor\_t& MHAParser::monitor\_t::operator= (

```
const monitor_t & ) [default]
```

**5.314.3.2 op\_query()** std::string MHAParser::monitor\_t::op\_query (

```
expression_t & x ) [virtual]
```

Reimplemented from **MHAParser::base\_t** (p. 1033).

**5.314.3.3 query\_dump()** std::string MHAParser::monitor\_t::query\_dump (

```
const std::string & ) [virtual]
```

Reimplemented from **MHAParser::base\_t** (p. 1033).

**5.314.3.4 query\_perm()** std::string MHAParser::monitor\_t::query\_perm (

```
const std::string & ) [virtual]
```

Reimplemented from **MHAParser::base\_t** (p. 1033).

Reimplemented in **MHAParser::variable\_t** (p. 1115).

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

- **mha\_parser.hh**
- **mha\_parser.cpp**

### 5.315 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`, `MHFilter::adapt_filter_t`, `MHFilter::iir_filter_t`, `MHAIOJack::io_jack_t`, `MHAIOJackdb::io_jack_t`, `MHAIOPortAudio::device_info_t`, `MHAIOPortAudio::io_portaudio_t`, `MHAIOPortAudio::stream_info_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< cohflit_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

### 5.315.1 Detailed Description

Parser node class.

A **parser\_t** (p. 1098) instance is a node in the configuration tree. A parser node can contain any number of other **parser\_t** (p. 1098) instances or configuration language variables. These items are inserted into a parser node using the **parser\_t::insert\_item** (p. 1100) method.

## 5.315.2 Constructor & Destructor Documentation

**5.315.2.1 `parser_t()`** `MHAParser::parser_t::parser_t (`  
`const std::string & help_text = "" )`

Construct detached node to be used in the configuration tree.

### Parameters

<code>help_text</code>	A text describing this node. E.g. if this node lives at the root of some openMHA plugin, then the help text should describe the functionality of the plugin.
------------------------	--

**5.315.2.2 `~parser_t()`** `MHAParser::parser_t::~parser_t ( )`

## 5.315.3 Member Function Documentation

**5.315.3.1 `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

<code>n</code>	Name of the item in the configuration tree
<code>e</code>	C++ pointer to the item instance. <code>e</code> can either point to a variable, to a monitor, or to another sub-parser.

**5.315.3.2 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.
----------	--

**5.315.3.3 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.
----------	--

**5.315.3.4 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.
-------------	---------------------------------------

**5.315.3.5 op\_subparse()** std::string MHAParser::parser\_t::op\_subparse ( **expression\_t** & *x* ) [protected], [virtual]

Reimplemented from **MHAParser::base\_t** (p. 1032).

**5.315.3.6 op\_setval()** std::string MHAParser::parser\_t::op\_setval (  
    **expression\_t** & x) [protected], [virtual]

Reimplemented from **MHAParser::base\_t** (p. 1033).

**5.315.3.7 op\_query()** std::string MHAParser::parser\_t::op\_query (  
    **expression\_t** & x) [protected], [virtual]

Reimplemented from **MHAParser::base\_t** (p. 1033).

**5.315.3.8 query\_type()** std::string MHAParser::parser\_t::query\_type (  
    const std::string &) [protected], [virtual]

Reimplemented from **MHAParser::base\_t** (p. 1034).

**5.315.3.9 query\_dump()** std::string MHAParser::parser\_t::query\_dump (  
    const std::string &) [protected], [virtual]

Reimplemented from **MHAParser::base\_t** (p. 1033).

**5.315.3.10 query\_entries()** std::string MHAParser::parser\_t::query\_entries (  
    const std::string &) [protected], [virtual]

Reimplemented from **MHAParser::base\_t** (p. 1033).

**5.315.3.11 query\_readfile()** std::string MHAParser::parser\_t::query\_readfile ( const std::string & fname ) [protected], [virtual]

Reimplemented from **MHAParser::base\_t** (p. 1034).

**5.315.3.12 query\_savefile()** std::string MHAParser::parser\_t::query\_savefile ( const std::string & fname ) [protected], [virtual]

Reimplemented from **MHAParser::base\_t** (p. 1035).

**5.315.3.13 query\_savefile\_compact()** std::string MHAParser::parser\_t::query\_↔ savefile\_compact ( const std::string & fname ) [protected], [virtual]

Reimplemented from **MHAParser::base\_t** (p. 1035).

**5.315.3.14 query\_savemons()** std::string MHAParser::parser\_t::query\_savemons ( const std::string & fname ) [protected], [virtual]

Reimplemented from **MHAParser::base\_t** (p. 1035).

**5.315.3.15 query\_val()** std::string MHAParser::parser\_t::query\_val ( const std::string & s ) [protected], [virtual]

Reimplemented from **MHAParser::base\_t** (p. 1034).

**5.315.3.16 query\_listids()** std::string MHAParser::parser\_t::query\_listids ( const std::string & s ) [protected], [virtual]

Reimplemented from **MHAParser::base\_t** (p. 1035).

**5.315.3.17 set\_id\_string()** void MHAParser::parser\_t::set\_id\_string ( const std::string & s ) [protected]

**5.315.3.18 has\_entry()** bool MHAParser::parser\_t::has\_entry ( const std::string & s ) [protected]

## 5.315.4 Member Data Documentation

**5.315.4.1 entries entry\_map\_t** MHAParser::parser\_t::entries [private]

**5.315.4.2 id\_string** std::string MHAParser::parser\_t::id\_string [private]

identification string

**5.315.4.3 last\_errormsg** std::string MHAParser::parser\_t::last\_errormsg [private]

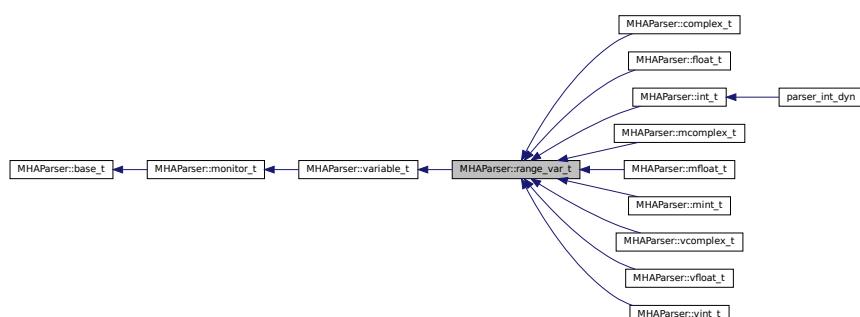
The documentation for this class was generated from the following files:

- **mha\_parser.hh**
- **mha\_parser.cpp**

## 5.316 MHAParser::range\_var\_t Class Reference

Base class for all variables with a numeric value range.

Inheritance diagram for MHAParser::range\_var\_t:



## Public Member Functions

- **range\_var\_t** (const std::string &, const std::string &="")
- **range\_var\_t** (const **range\_var\_t** &)
- std::string **query\_range** (const std::string &)
- void **set\_range** (const std::string &r)  
*Change the valid range of a variable.*
- void **validate** (const int &)
- void **validate** (const float &)
- void **validate** (const **mha\_complex\_t** &)
- void **validate** (const std::vector< int > &)
- void **validate** (const std::vector< float > &)
- void **validate** (const std::vector< **mha\_complex\_t** > &)
- void **validate** (const std::vector< std::vector< int > > &)
- void **validate** (const std::vector< std::vector< float > > &)
- void **validate** (const std::vector< std::vector< **mha\_complex\_t** > > &)

## Protected Attributes

- float **low\_limit**  
*Lower limit of range.*
- float **up\_limit**  
*Upper limit of range.*
- bool **low\_incl**  
*Lower limit is included (or excluded) in range.*
- bool **up\_incl**  
*Upper limit is included (or excluded) in range.*
- bool **check\_low**  
*Check lower limit.*
- bool **check\_up**  
*Check upper limit.*
- bool **check\_range**  
*Range checking is active.*

## Additional Inherited Members

### 5.316.1 Detailed Description

Base class for all variables with a numeric value range.

## 5.316.2 Constructor & Destructor Documentation

**5.316.2.1 `range_var_t()` [1/2]** MHAParser::range\_var\_t::range\_var\_t (

```
const std::string & h,  
const std::string & r = "")
```

**5.316.2.2 `range_var_t()` [2/2]** MHAParser::range\_var\_t::range\_var\_t (

```
const range_var_t & src)
```

## 5.316.3 Member Function Documentation

**5.316.3.1 `query_range()`** std::string MHAParser::range\_var\_t::query\_range (

```
const std::string & ) [virtual]
```

Reimplemented from **MHAParser::base\_t** (p. 1034).

**5.316.3.2 `set_range()`** void MHAParser::range\_var\_t::set\_range (

```
const std::string & r)
```

Change the valid range of a variable.

### Parameters

<i>r</i>	New range of the variable (string representation)
----------	---

**5.316.3.3 `validate()` [1/9]** void MHAParser::range\_var\_t::validate (

```
const int & v)
```

**5.316.3.4 validate() [2/9]** void MHParse::range\_var\_t::validate ( const float & v )

**5.316.3.5 validate() [3/9]** void MHParse::range\_var\_t::validate ( const mha\_complex\_t & v )

**5.316.3.6 validate() [4/9]** void MHParse::range\_var\_t::validate ( const std::vector< int > & v )

**5.316.3.7 validate() [5/9]** void MHParse::range\_var\_t::validate ( const std::vector< float > & v )

**5.316.3.8 validate() [6/9]** void MHParse::range\_var\_t::validate ( const std::vector< mha\_complex\_t > & v )

**5.316.3.9 validate() [7/9]** void MHParse::range\_var\_t::validate ( const std::vector< std::vector< int > > & v )

**5.316.3.10 validate() [8/9]** void MHParse::range\_var\_t::validate ( const std::vector< std::vector< float > > & v )

**5.316.3.11 validate() [9/9]** void MHParse::range\_var\_t::validate ( const std::vector< std::vector< mha\_complex\_t > > & v )

---

## 5.316.4 Member Data Documentation

**5.316.4.1 low\_limit** float MHAParser::range\_var\_t::low\_limit [protected]

Lower limit of range.

**5.316.4.2 up\_limit** float MHAParser::range\_var\_t::up\_limit [protected]

Upper limit of range.

**5.316.4.3 low\_incl** bool MHAParser::range\_var\_t::low\_incl [protected]

Lower limit is included (or excluded) in range.

**5.316.4.4 up\_incl** bool MHAParser::range\_var\_t::up\_incl [protected]

Upper limit is included (or excluded) in range.

**5.316.4.5 check\_low** bool MHAParser::range\_var\_t::check\_low [protected]

Check lower limit.

**5.316.4.6 check\_up** bool MHAParser::range\_var\_t::check\_up [protected]

Check upper limit.

### 5.316.4.7 check\_range bool MHAParser::range\_var\_t::check\_range [protected]

Range checking is active.

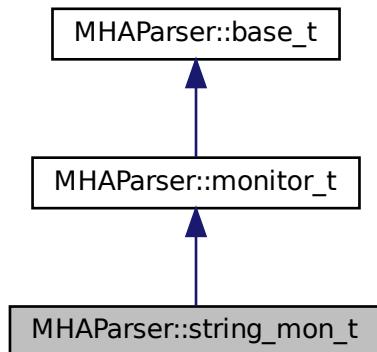
The documentation for this class was generated from the following files:

- [mha\\_parser.hh](#)
- [mha\\_parser.cpp](#)

## 5.317 MHAParser::string\_mon\_t Class Reference

Monitor with string value.

Inheritance diagram for MHAParser::string\_mon\_t:



### Public Member Functions

- **string\_mon\_t** (const std::string &hlp)  
*Create a monitor variable for string values.*

### Public Attributes

- std::string **data**  
*Data field.*

## Protected Member Functions

- std::string **query\_val** (const std::string &)
- std::string **query\_type** (const std::string &)

## Additional Inherited Members

### 5.317.1 Detailed Description

Monitor with string value.

### 5.317.2 Constructor & Destructor Documentation

#### 5.317.2.1 **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.
------------	---

### 5.317.3 Member Function Documentation

#### 5.317.3.1 **query\_val()** std::string MHAParser::string\_mon\_t::query\_val ( const std::string & *s* ) [protected], [virtual]

Reimplemented from **MHAParser::base\_t** (p. 1034).

#### 5.317.3.2 **query\_type()** std::string MHAParser::string\_mon\_t::query\_type ( const std::string & ) [protected], [virtual]

Reimplemented from **MHAParser::base\_t** (p. 1034).

### 5.317.4 Member Data Documentation

#### 5.317.4.1 **data** std::string MHAParser::string\_mon\_t::data

Data field.

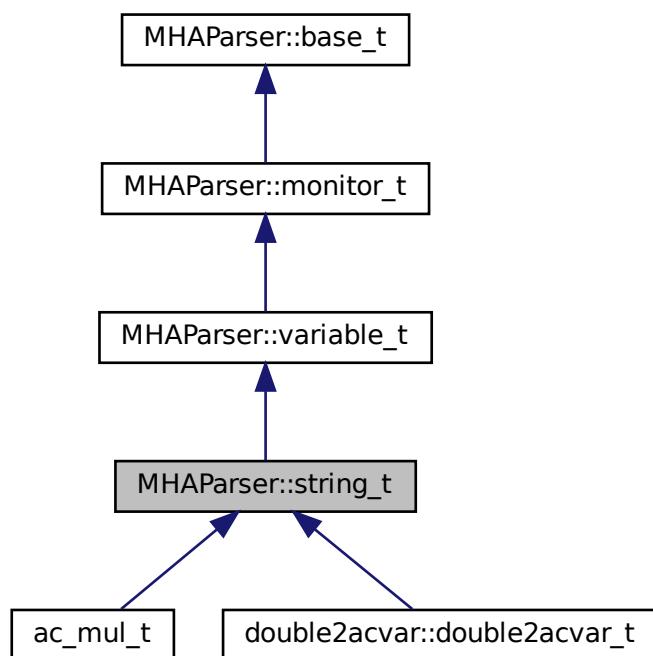
The documentation for this class was generated from the following files:

- **mha\_parser.hh**
- **mha\_parser.cpp**

## 5.318 MHAParser::string\_t Class Reference

Variable with a string value.

Inheritance diagram for MHAParser::string\_t:



## Public Member Functions

- **string\_t** (const std::string &, const std::string &)  
*Constructor of a openMHA configuration variable for string values.*

## Public Attributes

- std::string **data**  
*Data field.*

## Protected Member Functions

- std::string **op\_setval** ( expression\_t &)
- std::string **query\_type** (const std::string &)
- std::string **query\_val** (const std::string &)

## Additional Inherited Members

### 5.318.1 Detailed Description

Variable with a string value.

### 5.318.2 Constructor & Destructor Documentation

#### 5.318.2.1 **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

### 5.318.3 Member Function Documentation

**5.318.3.1 op\_setval()** std::string MHAParser::string\_t::op\_setval ( expression\_t & x ) [protected], [virtual]

Reimplemented from **MHAParser::variable\_t** (p. 1115).

**5.318.3.2 query\_type()** std::string MHAParser::string\_t::query\_type ( const std::string & ) [protected], [virtual]

Reimplemented from **MHAParser::base\_t** (p. 1034).

**5.318.3.3 query\_val()** std::string MHAParser::string\_t::query\_val ( const std::string & s ) [protected], [virtual]

Reimplemented from **MHAParser::base\_t** (p. 1034).

### 5.318.4 Member Data Documentation

**5.318.4.1 data** std::string MHAParser::string\_t::data

Data field.

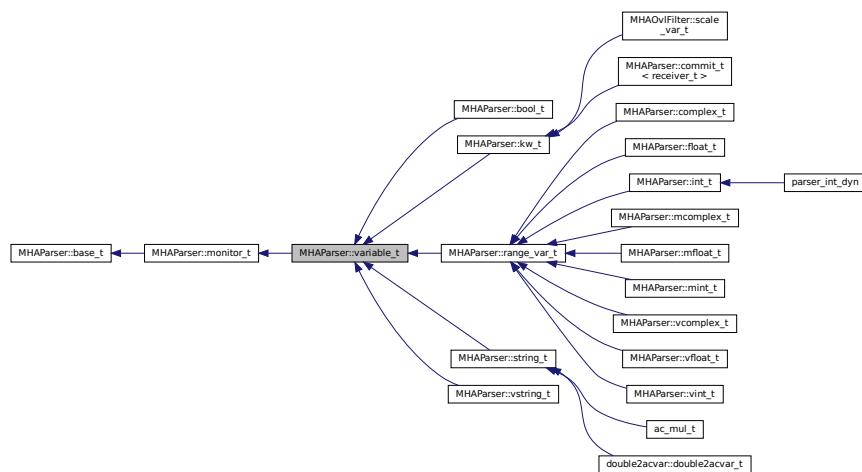
The documentation for this class was generated from the following files:

- [mha\\_parser.hh](#)
- [mha\\_parser.cpp](#)

## 5.319 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

#### 5.319.1 Detailed Description

Base class for variable nodes.

## 5.319.2 Constructor & Destructor Documentation

**5.319.2.1 variable\_t()** `MHAParser::variable_t::variable_t (const std::string & h )`

## 5.319.3 Member Function Documentation

**5.319.3.1 op\_setval()** `std::string MHAParser::variable_t::op_setval (expression_t & x ) [virtual]`

Reimplemented from **MHAParser::base\_t** (p. 1033).

Reimplemented in **MHAParser::mcomplex\_t** (p. 1078), **MHAParser::mfloat\_t** (p. 1083), **MHAParser::mint\_t** (p. 1095), **MHAParser::vcomplex\_t** (p. 1119), **MHAParser::vfloat\_t** (p. 1125), **MHAParser::vint\_t** (p. 1129), **MHAParser::complex\_t** (p. 1054), **MHAParser::float\_t** (p. 1061), **MHAParser::int\_t** (p. 1066), **MHAParser::bool\_t** (p. 1045), **MHAParser::vstring\_t** (p. 1133), **MHAParser::string\_t** (p. 1113), and **MHAParser::kw\_t** (p. 1073).

**5.319.3.2 query\_perm()** `std::string MHAParser::variable_t::query_perm (const std::string & ) [virtual]`

Reimplemented from **MHAParser::monitor\_t** (p. 1097).

**5.319.3.3 setlock()** `void MHAParser::variable_t::setlock (const bool & b )`

Lock a variable against write access.

### Parameters

<code>b</code>	Lock state
----------------	------------

## 5.319.4 Member Data Documentation

### 5.319.4.1 **locked** bool MHAParser::variable\_t::locked [private]

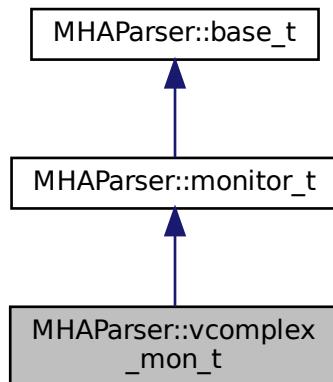
The documentation for this class was generated from the following files:

- **mha\_parser.hh**
- **mha\_parser.cpp**

## 5.320 MHAParser::vcomplex\_mon\_t Class Reference

Monitor with vector of complex values.

Inheritance diagram for MHAParser::vcomplex\_mon\_t:



### Public Member Functions

- **vcomplex\_mon\_t** (const std::string &hlp)  
*Create a vector of complex monitor values.*

### Public Attributes

- std::vector< **mha\_complex\_t** > **data**  
*Data field.*

## Protected Member Functions

- std::string **query\_val** (const std::string &)
- std::string **query\_type** (const std::string &)

## Additional Inherited Members

### 5.320.1 Detailed Description

Monitor with vector of complex values.

### 5.320.2 Constructor & Destructor Documentation

**5.320.2.1 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.
------------	---

### 5.320.3 Member Function Documentation

**5.320.3.1 query\_val()** std::string MHAParser::vcomplex\_mon\_t::query\_val ( const std::string & *s* ) [protected], [virtual]

Reimplemented from **MHAParser::base\_t** (p. 1034).

**5.320.3.2 query\_type()** std::string MHAParser::vcomplex\_mon\_t::query\_type ( const std::string & ) [protected], [virtual]

Reimplemented from **MHAParser::base\_t** (p. 1034).

## 5.320.4 Member Data Documentation

### 5.320.4.1 **data** std::vector< **mha\_complex\_t**> MHAParser::vcomplex\_mon\_t::data

Data field.

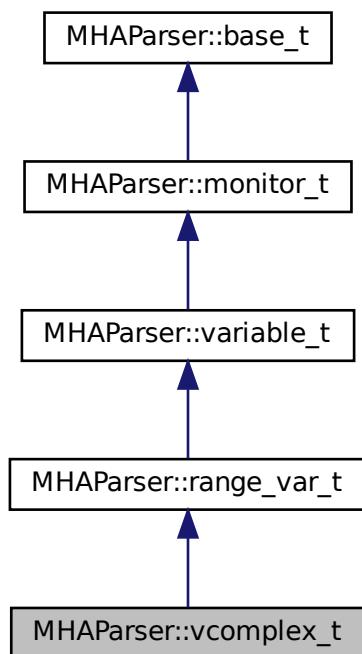
The documentation for this class was generated from the following files:

- **mha\_parser.hh**
- **mha\_parser.cpp**

## 5.321 MHAParser::vcomplex\_t Class Reference

Vector variable with complex value.

Inheritance diagram for MHAParser::vcomplex\_t:



## Public Member Functions

- **vcomplex\_t** (const std::string &, const std::string &, const std::string &= "")

## Public Attributes

- std::vector< **mha\_complex\_t** > **data**

*Data field.*

## Protected Member Functions

- std::string **op\_setval** ( **expression\_t** &)
- std::string **query\_type** (const std::string &)
- std::string **query\_val** (const std::string &)

## Additional Inherited Members

### 5.321.1 Detailed Description

Vector variable with complex value.

### 5.321.2 Constructor & Destructor Documentation

```
5.321.2.1 vcomplex_t() MHAParser::vcomplex_t::vcomplex_t (
    const std::string & h,
    const std::string & v,
    const std::string & rg = "" )
```

### 5.321.3 Member Function Documentation

**5.321.3.1 `op_setval()`** std::string MHAParser::vcomplex\_t::op\_setval ( expression\_t & x ) [protected], [virtual]

Reimplemented from **MHAParser::variable\_t** (p. 1115).

**5.321.3.2 `query_type()`** std::string MHAParser::vcomplex\_t::query\_type ( const std::string & ) [protected], [virtual]

Reimplemented from **MHAParser::base\_t** (p. 1034).

**5.321.3.3 `query_val()`** std::string MHAParser::vcomplex\_t::query\_val ( const std::string & s ) [protected], [virtual]

Reimplemented from **MHAParser::base\_t** (p. 1034).

## 5.321.4 Member Data Documentation

**5.321.4.1 `data`** std::vector< mha\_complex\_t > MHAParser::vcomplex\_t::data

Data field.

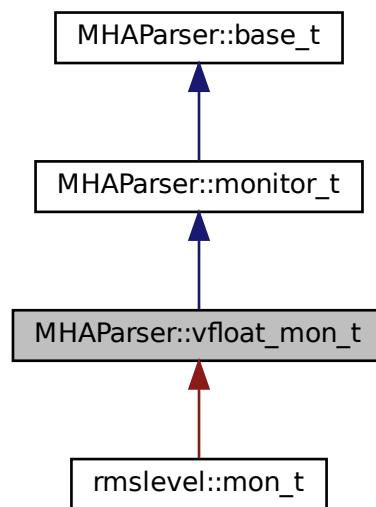
The documentation for this class was generated from the following files:

- **mha\_parser.hh**
- **mha\_parser.cpp**

## 5.322 MHAParser::vfloat\_mon\_t Class Reference

Vector of floats monitor.

Inheritance diagram for MHAParser::vfloat\_mon\_t:



### Public Member Functions

- **vfloat\_mon\_t** (const std::string &hlp)  
*Create a vector of floating point monitor values.*

### Public Attributes

- std::vector< float > **data**  
*Data field.*

### Protected Member Functions

- std::string **query\_val** (const std::string &)
- std::string **query\_type** (const std::string &)

## Additional Inherited Members

### 5.322.1 Detailed Description

Vector of floats monitor.

### 5.322.2 Constructor & Destructor Documentation

**5.322.2.1 `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.
------------------	---

### 5.322.3 Member Function Documentation

**5.322.3.1 `query_val()`** `std::string MHAParser::vfloat_mon_t::query_val ( const std::string & s ) [protected], [virtual]`

Reimplemented from **MHAParser::base\_t** (p. 1034).

**5.322.3.2 `query_type()`** `std::string MHAParser::vfloat_mon_t::query_type ( const std::string & ) [protected], [virtual]`

Reimplemented from **MHAParser::base\_t** (p. 1034).

### 5.322.4 Member Data Documentation

**5.322.4.1 data** std::vector<float> MHAParser::vfloat\_mon\_t::data

Data field.

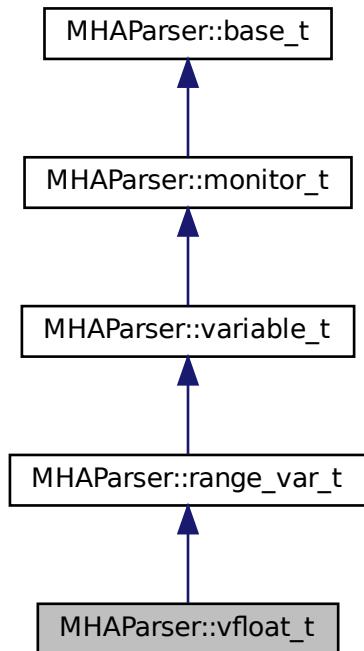
The documentation for this class was generated from the following files:

- [mha\\_parser.hh](#)
- [mha\\_parser.cpp](#)

## 5.323 MHAParser::vfloat\_t Class Reference

Vector variable with float value.

Inheritance diagram for MHAParser::vfloat\_t:



### Public Member Functions

- **vfloat\_t** (const std::string &, const std::string &, const std::string &="")  
*Create a float vector parser variable.*

## Public Attributes

- std::vector< float > **data**

*Data field.*

## Protected Member Functions

- std::string **op\_setval** ( expression\_t & )
- std::string **query\_type** (const std::string & )
- std::string **query\_val** (const std::string & )

## Additional Inherited Members

### 5.323.1 Detailed Description

Vector variable with float value.

### 5.323.2 Constructor & Destructor Documentation

```
5.323.2.1 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.

•

### 5.323.3 Member Function Documentation

**5.323.3.1 op\_setval()** std::string MHAParser::vfloat\_t::op\_setval ( expression\_t & x ) [protected], [virtual]

Reimplemented from **MHAParser::variable\_t** (p. 1115).

**5.323.3.2 query\_type()** std::string MHAParser::vfloat\_t::query\_type ( const std::string & ) [protected], [virtual]

Reimplemented from **MHAParser::base\_t** (p. 1034).

**5.323.3.3 query\_val()** std::string MHAParser::vfloat\_t::query\_val ( const std::string & s ) [protected], [virtual]

Reimplemented from **MHAParser::base\_t** (p. 1034).

### 5.323.4 Member Data Documentation

**5.323.4.1 data** std::vector<float> MHAParser::vfloat\_t::data

Data field.

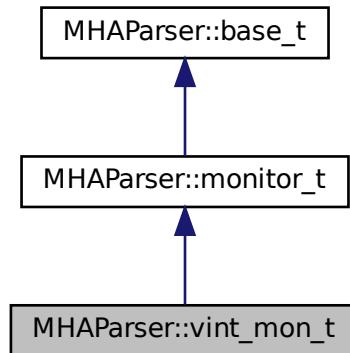
The documentation for this class was generated from the following files:

- **mha\_parser.hh**
- **mha\_parser.cpp**

## 5.324 MHAParser::vint\_mon\_t Class Reference

Vector of ints monitor.

Inheritance diagram for MHAParser::vint\_mon\_t:



### Public Member Functions

- **vint\_mon\_t** (const std::string &hlp)  
*Create a vector of integer monitor values.*

### Public Attributes

- std::vector< int > **data**  
*Data field.*

### Protected Member Functions

- std::string **query\_val** (const std::string &)
- std::string **query\_type** (const std::string &)

### Additional Inherited Members

#### 5.324.1 Detailed Description

Vector of ints monitor.

### 5.324.2 Constructor & Destructor Documentation

**5.324.2.1 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.
------------------	---

### 5.324.3 Member Function Documentation

**5.324.3.1 query\_val()** `std::string MHAParser::vint_mon_t::query_val (const std::string & s) [protected], [virtual]`

Reimplemented from **MHAParser::base\_t** (p. 1034).

**5.324.3.2 query\_type()** `std::string MHAParser::vint_mon_t::query_type (const std::string & ) [protected], [virtual]`

Reimplemented from **MHAParser::base\_t** (p. 1034).

### 5.324.4 Member Data Documentation

**5.324.4.1 data** `std::vector<int> MHAParser::vint_mon_t::data`

Data field.

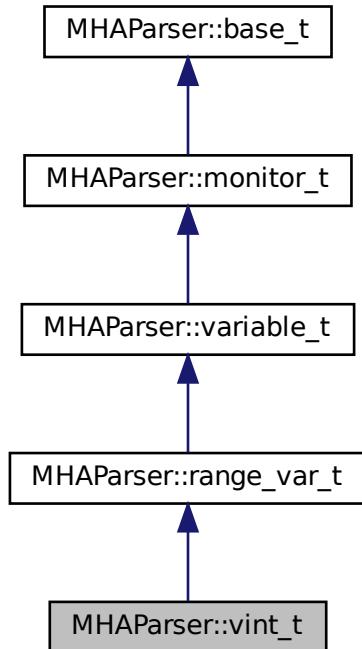
The documentation for this class was generated from the following files:

- **mha\_parser.hh**
- **mha\_parser.cpp**

### 5.325 MHAParser::vint\_t Class Reference

Variable with vector<int> value.

Inheritance diagram for MHAParser::vint\_t:



#### Public Member Functions

- **vint\_t** (const std::string &, const std::string &, const std::string &="")
   
*Constructor.*

#### Public Attributes

- std::vector< int > **data**
  
*Data field.*

#### Protected Member Functions

- std::string **op\_setval** ( expression\_t &)
- std::string **query\_type** (const std::string &)
- std::string **query\_val** (const std::string &)

## Additional Inherited Members

### 5.325.1 Detailed Description

Variable with vector<int> value.

### 5.325.2 Constructor & Destructor Documentation

#### 5.325.2.1 vint\_t() MHAParser::vint\_t::vint\_t (

```
    const std::string & h,
    const std::string & v,
    const std::string & rg = "" )
```

Constructor.

#### Parameters

<i>h</i>	help string
<i>v</i>	initial value
<i>rg</i>	optional: range constraint for all elements

### 5.325.3 Member Function Documentation

#### 5.325.3.1 op\_setval() std::string MHAParser::vint\_t::op\_setval (

```
    expression_t & x ) [protected], [virtual]
```

Reimplemented from **MHAParser::variable\_t** (p. [1115](#)).

#### 5.325.3.2 query\_type() std::string MHAParser::vint\_t::query\_type (

```
    const std::string & ) [protected], [virtual]
```

Reimplemented from **MHAParser::base\_t** (p. [1034](#)).

**5.325.3.3 query\_val()** std::string MHAParser::vint\_t::query\_val ( const std::string & s ) [protected], [virtual]

Reimplemented from **MHAParser::base\_t** (p. 1034).

#### 5.325.4 Member Data Documentation

**5.325.4.1 data** std::vector<int> MHAParser::vint\_t::data

Data field.

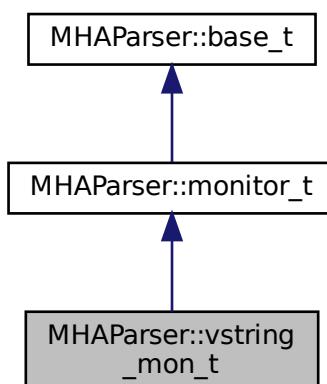
The documentation for this class was generated from the following files:

- **mha\_parser.hh**
- **mha\_parser.cpp**

### 5.326 MHAParser::vstring\_mon\_t Class Reference

Vector of monitors with string value.

Inheritance diagram for MHAParser::vstring\_mon\_t:



## Public Member Functions

- **vstring\_mon\_t** (const std::string &hlp)

*Create a vector of string monitor values.*

## Public Attributes

- std::vector< std::string > **data**

*Data field.*

## Protected Member Functions

- std::string **query\_val** (const std::string &)
- std::string **query\_type** (const std::string &)

## Additional Inherited Members

### 5.326.1 Detailed Description

Vector of monitors with string value.

### 5.326.2 Constructor & Destructor Documentation

#### 5.326.2.1 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.
------------	---

### 5.326.3 Member Function Documentation

**5.326.3.1 `query_val()`** `std::string MHAParser::vstring_mon_t::query_val ( const std::string & s ) [protected], [virtual]`

Reimplemented from **MHAParser::base\_t** (p. 1034).

**5.326.3.2 `query_type()`** `std::string MHAParser::vstring_mon_t::query_type ( const std::string & ) [protected], [virtual]`

Reimplemented from **MHAParser::base\_t** (p. 1034).

#### 5.326.4 Member Data Documentation

**5.326.4.1 `data`** `std::vector<std::string> MHAParser::vstring_mon_t::data`

Data field.

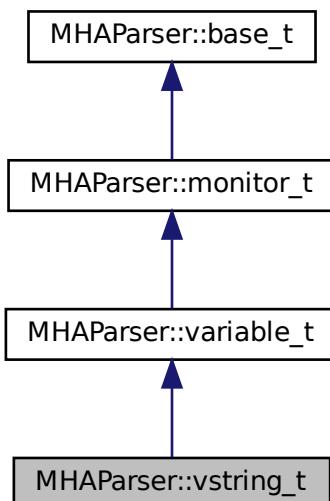
The documentation for this class was generated from the following files:

- `mha_parser.hh`
- `mha_parser.cpp`

#### 5.327 MHAParser::vstring\_t Class Reference

Vector variable with string values.

Inheritance diagram for MHAParser::vstring\_t:



## Public Member Functions

- **vstring\_t** (const std::string &, const std::string &)

## Public Attributes

- std::vector< std::string > **data**  
*Data field.*

## Protected Member Functions

- std::string **op\_setval** ( **expression\_t** & )
- std::string **query\_type** (const std::string & )
- std::string **query\_val** (const std::string & )

## Additional Inherited Members

### 5.327.1 Detailed Description

Vector variable with string values.

### 5.327.2 Constructor & Destructor Documentation

```
5.327.2.1 vstring_t() MHAParser::vstring_t::vstring_t (
    const std::string & h,
    const std::string & v )
```

### 5.327.3 Member Function Documentation

```
5.327.3.1 op_setval() std::string MHAParser::vstring_t::op_setval (
    expression_t & x ) [protected], [virtual]
```

Reimplemented from **MHAParser::variable\_t** (p. 1115).

**5.327.3.2 `query_type()`** `std::string MHAParser::vstring_t::query_type (`  
`const std::string & ) [protected], [virtual]`

Reimplemented from **MHAParser::base\_t** (p. 1034).

**5.327.3.3 `query_val()`** `std::string MHAParser::vstring_t::query_val (`  
`const std::string & s ) [protected], [virtual]`

Reimplemented from **MHAParser::base\_t** (p. 1034).

#### 5.327.4 Member Data Documentation

**5.327.4.1 `data`** `std::vector<std::string> MHAParser::vstring_t::data`

Data field.

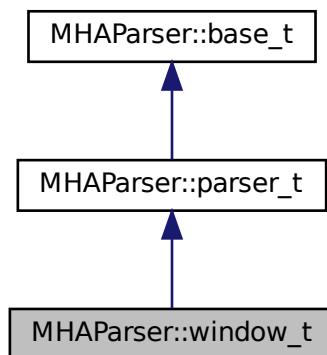
The documentation for this class was generated from the following files:

- [mha\\_parser.hh](#)
- [mha\\_parser.cpp](#)

#### 5.328 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.*
- **MHParse::window\_t::wtype\_t get\_type** () const  
*Return currently selected window type.*
- void **setlock** (bool b)

## Private Attributes

- **MHParse::kw\_t wtype**
- **MHParse::vfloat\_t user**

## Additional Inherited Members

### 5.328.1 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 **MHParse::parser\_t** (p. 1098) and can thus be inserted into the configuration tree using the **insert\_item()** (p. 1100) 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.

## 5.328.2 Member Enumeration Documentation

### 5.328.2.1 `wtype_t` enum `MHAParser::window_t::wtype_t`

Enumerator

wnd_rect	
wnd_hann	
wnd_hamming	
wnd_blackman	
wnd_bartlett	
wnd_user	

## 5.328.3 Constructor & Destructor Documentation

### 5.328.3.1 `window_t()` `MHAParser::window_t::window_t (` `const std::string & help = "Window type configuration." )`

Constructor to create parser class.

## 5.328.4 Member Function Documentation

### 5.328.4.1 `get_window() [1/5]` `MHAWindow::base_t MHAParser::window_t::get_window (` `unsigned int len ) const`

Create a window instance, use default parameters.

**5.328.4.2 get\_window() [2/5]** `MHAWindow::base_t MHAParser::window_t::get_window (`  
    `unsigned int len,`  
    `float xmin ) const`

Create a window instance.

**5.328.4.3 get\_window() [3/5]** `MHAWindow::base_t MHAParser::window_t::get_window (`  
    `unsigned int len,`  
    `float xmin,`  
    `float xmax ) const`

Create a window instance.

**5.328.4.4 get\_window() [4/5]** `MHAWindow::base_t MHAParser::window_t::get_window (`  
    `unsigned int len,`  
    `float xmin,`  
    `float xmax,`  
    `bool minincluded ) const`

Create a window instance.

**5.328.4.5 get\_window() [5/5]** `MHAWindow::base_t MHAParser::window_t::get_window (`  
    `unsigned int len,`  
    `float xmin,`  
    `float xmax,`  
    `bool minincluded,`  
    `bool maxincluded ) const`

Create a window instance.

**5.328.4.6 get\_type()** `MHAParser::window_t::wtype_t MHAParser::window_t::get_type (`  
    `) const`

Return currently selected window type.

**5.328.4.7 `setlock()`** `void MHAParser::window_t::setlock ( bool b ) [inline]`

## 5.328.5 Member Data Documentation

**5.328.5.1 `wtype`** `MHAParser::kw_t MHAParser::window_t::wtype [private]`

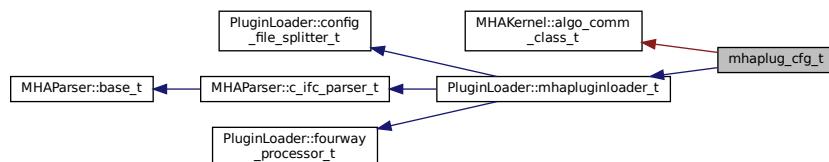
**5.328.5.2 `user`** `MHAParser::vfloat_t MHAParser::window_t::user [private]`

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

- `mha_windowparser.h`
- `mha_windowparser.cpp`

## 5.329 `mhaplug_cfg_t` Class Reference

Inheritance diagram for `mhaplug_cfg_t`:



### Public Member Functions

- `mhaplug_cfg_t ( algo_comm_t iac, const std::string & libname, bool use_own_ac )`
- `~mhaplug_cfg_t () throw ()`

### Additional Inherited Members

#### 5.329.1 Constructor & Destructor Documentation

```
5.329.1.1 mhaplug_cfg_t() mhaplug_cfg_t::mhaplug_cfg_t (
    algo_comm_t iac,
    const std::string & libname,
    bool use_own_ac )
```

**5.329.1.2 ~mhaplug\_cfg\_t()** mhaplug\_cfg\_t::~mhaplug\_cfg\_t () throw () [inline]

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

- **altplugs.cpp**

## 5.330 MHAPlugin::cfg\_node\_t< runtime\_cfg\_t > Class Template Reference

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

### Public Member Functions

- **cfg\_node\_t (runtime\_cfg\_t \*runtime\_cfg)**  
*Constructor for a singly linked list node.*
- **~cfg\_node\_t ()**  
*Destructor of the singly linked list node.*

### Public Attributes

- std::atomic< cfg\_node\_t< runtime\_cfg\_t > \* > **next**  
*A pointer to the next node in the singly linked list.*
- std::atomic< bool > **not\_in\_use**  
*Initially this data member is set to false by the constructor.*
- runtime\_cfg\_t \* **data**  
*A native pointer to the runtime configuration managed by this node.*

### 5.330.1 Detailed Description

```
template<class runtime_cfg_t>
class MHAPlugin::cfg_node_t< runtime_cfg_t >
```

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

The singly linked list is designed for a single producer thread and a single consumer thread, where the producer is also responsible for destroying objects when they are no longer needed because the consumer is the signal processing thread that cannot afford memory allocation or deallocation operations.

### 5.330.2 Constructor & Destructor Documentation

**5.330.2.1 `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

<code>runtime_cfg</code>	Pointer to a runtime configuration object that was just created on the heap. The newly constructed <code>cfg_node_t</code> (p. 1139) object takes over object ownership of the pointed-to runtime configuration and will call delete on it in its destructor.
--------------------------	---

**5.330.2.2 `~cfg_node_t()`** template<class runtime\_cfg\_t >  
**MHAPlugin::cfg\_node\_t< runtime\_cfg\_t >::~ cfg\_node\_t**

Destructor of the singly linked list node.

Will also delete the pointed-to data object (the runtime configuration)

### 5.330.3 Member Data Documentation

```
5.330.3.1 next template<class runtime_cfg_t >
std::atomic< cfg_node_t<runtime_cfg_t>*> MHAPlugin::cfg_node_t< runtime_cfg_t >::next
```

A pointer to the next node in the singly linked list.

On construction, this will be a NULL pointer. New objects can be appended to the singly linked list by writing the address to the next node into this data member. Since this pointer is std::atomic, writing to it is a release operation which means all threads seeing the new value of the next pointer will also see all other writes to memory that the thread doing this write has performed, which includes anything the constructor of the new runtime config has written and the assignment of the address of the new runtime config to the data pointer of the next node. This prevents making half-constructed runtime configuration objects visible to the consumer thread.

```
5.330.3.2 not_in_use template<class runtime_cfg_t >
std::atomic<bool> MHAPlugin::cfg_node_t< runtime_cfg_t >::not_in_use
```

Initially this data member is set to false by the constructor.

It is set to true by the consumer thread when it no longer needs the run time configuration pointed to by this node's data member. This bool is atomic because this node and the runtime object it points to can be deleted by the configuration thread as soon as value true is stored in this bool, which happens right after the processing thread acquires a pointer to the next node in the singly linked list. The atomic ensures all threads agree on this "right after" ordering.

```
5.330.3.3 data template<class runtime_cfg_t >
runtime_cfg_t* MHAPlugin::cfg_node_t< runtime_cfg_t >::data
```

A native pointer to the runtime configuration managed by this node.

The runtime configuration lives on the heap and is owned by this node. It is deleted in this node's destructor. The runtime configuration object must be created by client code with operator new before ownership is transferred to this node by passing a pointer to it as the constructor's parameter. This pointer does not need to be atomic, memory access ordering is ensured by the atomic next pointer and placing accesses to "data" in the correct places relative to accesses to "next".

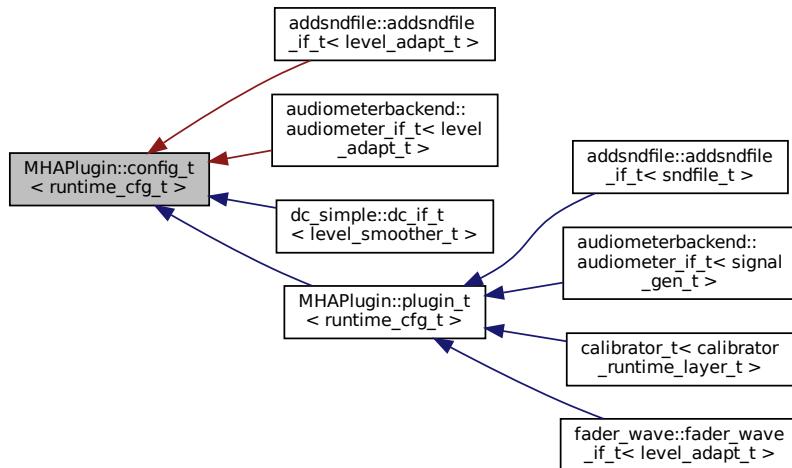
The documentation for this class was generated from the following file:

- **mha\_plugin.hh**

### 5.331 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. 1146) for housekeeping.*
- **void remove\_all\_cfg ()**  
*To be called on Plugin destruction, will delete all runtime configuration list nodes and objects regardless of their in\_use flag.*

## Protected Attributes

- runtime\_cfg\_t \* **cfg**

*Pointer to the runtime configuration currently used by the signal processing thread.*

## Private Attributes

- std::atomic< MHAPlugin::cfg\_node\_t< runtime\_cfg\_t > \* > **cfg\_root**  
*Start of a singly linked list of runtime configuration objects.*
- MHAPlugin::cfg\_node\_t< runtime\_cfg\_t > \* **cfg\_node\_current**  
*Pointer to the currently used plugin runtime configurations.*

### 5.331.1 Detailed Description

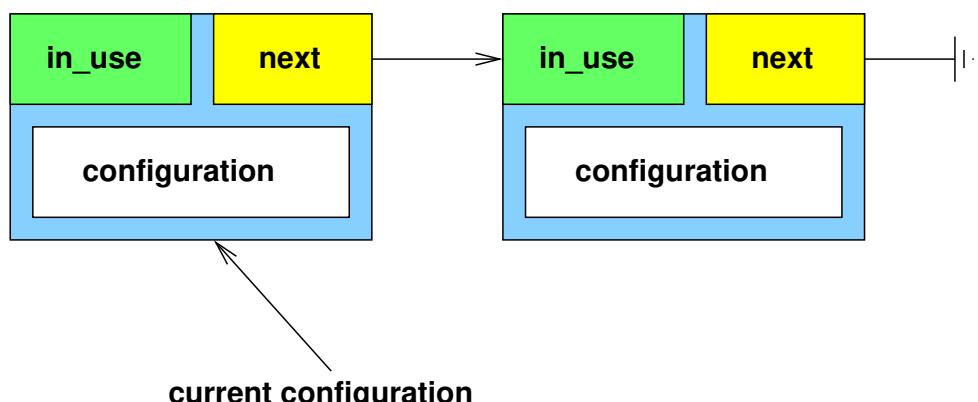
```
template<class runtime_cfg_t>
class MHAPlugin::config_t< runtime_cfg_t >
```

Template class for thread safe configuration.

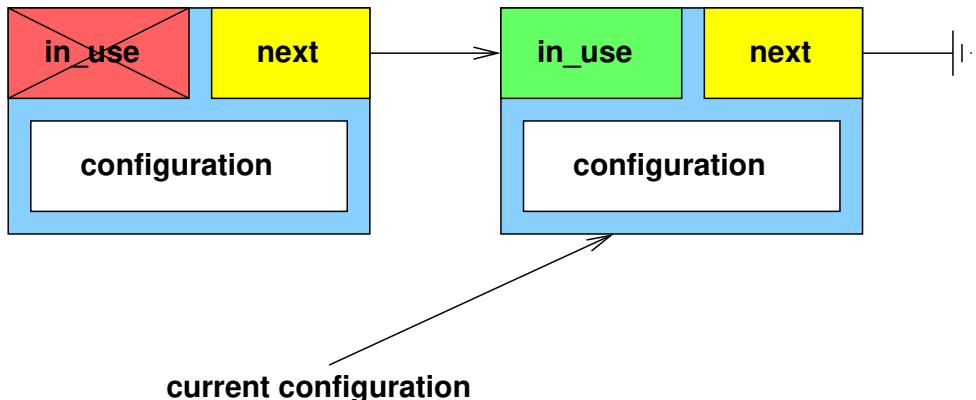
This template class provides a mechanism for the handling of thread safe configuration which is required for run time configuration changes of the openMHA plugins.

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. 1146). In the processing thread, the actual configuration can be received by a call of **poll\_config()** (p. 1145).



**Figure 5 Schematic drawing of runtime configuration update: configuration updated, but not used yet.**



**Figure 6 Schematic drawing of runtime configuration update: configuration in use.**

To ensure lock-free thread safety, we use C++ atomics and rely on the C++ memory model. We only use store-release and load-acquire operations by using C++ atomics with the default memory ordering. The semantics of these are:

The store-release operation atomically writes to an atomic variable, while the load-acquire operation atomically reads from an atomic variable.

The C++ memory model guarantees that all previous writes to memory performed by the thread doing the store-release are visible to other threads when they see the new value in the shared atomic variable when that value is read by the other thread with a load-acquire operation.

An important precondition of this synchronization scheme is that there is only ever one audio thread and one configuration thread per plugin, i.e. there is only one thread doing the `push_config` and one thread doing the `poll_config` for each instance of `config_t` (p. 1142).

For more details on atomics, refer to the C++11 or later documentation, or to these conference talks by Sutter:

- Atomic Weapons, 2012
- Lock-Free Programming, 2014

### 5.331.2 Constructor & Destructor Documentation

**5.331.2.1 config\_t()** `template<class runtime_cfg_t >`  
`MHAPPlugin::config_t< runtime_cfg_t >:: config_t`

```
5.331.2.2 ~config_t() template<class runtime_cfg_t >
MHAPlugin::config_t< runtime_cfg_t >::~ config_t
```

### 5.331.3 Member Function Documentation

```
5.331.3.1 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

<b>MHA_Error</b> (p. <a href="#">760</a> )	if the resulting runtime configuration is NULL. This usually means that no push_config has occurred.
--	--

```
5.331.3.2 peek_config() template<class runtime_cfg_t >
runtime_cfg_t * MHAPlugin::config_t< runtime_cfg_t >::peek_config [protected]
```

Receive the latest run time configuration without changing the configuration pointer.

This function retrieves the latest run time configuration. Returns a pointer to the latest runtime configuration without updating the data member cfg. For use in the configuration thread when

creation of a new runtime configuration object needs access to the previously created runtime configuration object. Should normally not be used because it introduces synchronization requirements between configuration thread and signal processing thread.

```
5.331.3.3 push_config() template<class runtime_cfg_t >
void MHAPPlugin::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. 1146) 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 <b>config_t</b> (p. 1142) instance.
-------------	---

```
5.331.3.4 cleanup_unused_cfg() template<class runtime_cfg_t >
void MHAPPlugin::config_t< runtime_cfg_t >::cleanup_unused_cfg [protected]
```

To be called by the **push\_config()** (p. 1146) for housekeeping.

Will delete any no longer used runtime configuration objects.

```
5.331.3.5 remove_all_cfg() template<class runtime_cfg_t >
void MHAPPlugin::config_t< runtime_cfg_t >::remove_all_cfg [protected]
```

To be called on Plugin destruction, will delete all runtime configuration list nodes and objects regardless of their in\_use flag.

### 5.331.4 Member Data Documentation

```
5.331.4.1 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. 1145) on this object.

```
5.331.4.2 cfg_root template<class runtime_cfg_t >
std::atomic< MHAPlugin::cfg_node_t<runtime_cfg_t> *> MHAPlugin::config_t< runtime_cfg_t >::cfg_root [private]
```

Start of a singly linked list of runtime configuration objects.

cfg\_root points to the oldest still existing node of that list. After object creation this pointer is updated by the configuration thread and then read by the signal processing thread. To ensure proper order of memory accesses for this transfer between threads, it needs to be atomic, this ensures that the start of the singly linked list of runtime configurations will be properly visible to the signal processing on startup.

```
5.331.4.3 cfg_node_current template<class runtime_cfg_t >
MHAPlugin::cfg_node_t<runtime_cfg_t>* MHAPlugin::config_t< runtime_cfg_t >::cfg_node_current [private]
```

Pointer to the currently used plugin runtime configurations.

Used as a hint for poll\_config where to start looking for the newest node. This optimization allows poll\_config to scale better with the number of nodes not yet cleaned up by push\_config. Does not need to be atomic because it is only used within the signal processing thread.

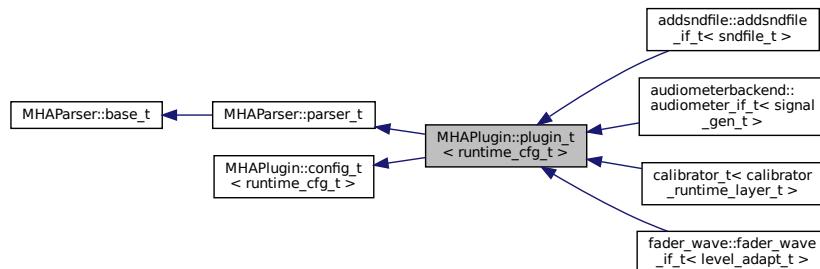
The documentation for this class was generated from the following file:

- **mha\_plugin.hh**

## 5.332 MHAPlugin::plugin\_t< runtime\_cfg\_t > Class Template Reference

The template class for C++ openMHA plugins.

Inheritance diagram for MHAPlugin::plugin\_t< runtime\_cfg\_t >:



## Public Member Functions

- **plugin\_t** (const std::string &, const **algo\_comm\_t** &)
 

*Constructor of plugin template.*
- virtual ~**plugin\_t** ()
- virtual void **prepare** ( **mhaconfig\_t** &)=0
- virtual void **release** ()
- void **prepare\_** ( **mhaconfig\_t** &)
- void **release\_** ()
- bool **is\_prepared** () const
 

*Flag, if the prepare method is successfully called (or currently evaluated)*
- **mhaconfig\_t input\_cfg** () const
 

*Current input channel configuration.*
- **mhaconfig\_t output\_cfg** () const
 

*Current output channel configuration.*

## Protected Attributes

- **mhaconfig\_t tftype**

*Member for storage of plugin interface configuration.*
- **algo\_comm\_t ac**

*AC handle of the chain.*

## Private Attributes

- bool **is\_prepared\_**
- **mhaconfig\_t input\_cfg\_**
- **mhaconfig\_t output\_cfg\_**
- **MHAParser::mhaconfig\_mon\_t mhaconfig\_in**
- **MHAParser::mhaconfig\_mon\_t mhaconfig\_out**

## Additional Inherited Members

### 5.332.1 Detailed Description

```
template<class runtime_cfg_t>
class MHAParser::plugin_t< runtime_cfg_t >
```

The template class for C++ openMHA plugins.

### Template Parameters

<i>runtime_</i> <i>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. 1620).

The template parameter `runtime_cfg_t` should be the runtime configuration of the plugin.

See **MHAPlugin::config\_t** (p. 1142) for details on the thread safe communication update mechanism.

### 5.332.2 Constructor & Destructor Documentation

```
5.332.2.1 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).

```
5.332.2.2 ~plugin_t() template<class runtime_cfg_t >
MHAPlugin::plugin_t< runtime_cfg_t >::~ plugin_t [virtual]
```

### 5.332.3 Member Function Documentation

```
5.332.3.1 prepare() template<class runtime_cfg_t >
virtual void MHAPplugin::plugin_t< runtime_cfg_t >::prepare (
    mhaconfig_t & ) [pure virtual]
```

Implemented in **calibrator\_t** (p. 339), **dc::dc\_if\_t** (p. 381), **fftfbpow::fftfbpow\_interface\_t** (p. 491), **save\_spec\_t** (p. 1419), **save\_wave\_t** (p. 1421), **wave2spec\_if\_t** (p. 1489), **windnoise::if\_t** (p. 1510), **attenuate20\_t** (p. 314), **matlab\_wrapper::matlab\_wrapper\_t** (p. 696), **example3\_t** (p. 471), **example4\_t** (p. 475), **droptect\_t** (p. 437), **example1\_t** (p. 464), **example2\_t** (p. 467), **mconv::MConv** (p. 715), **gtfb\_simple\_t** (p. 568), **plingploing::if\_t** (p. 1338), **plugins::hoertech::acrec::acrec\_t** (p. 1376), **wavrec\_t** (p. 1499), **altconfig\_t** (p. 297), **gtfb\_simd\_t** (p. 559), **bbcalib\_interface\_t** (p. 334), **addsndfile::addsndfile\_if\_t** (p. 251), **adm\_if\_t** (p. 272), **testplugin::if\_t** (p. 1482), **analysisispath\_if\_t** (p. 311), **osc2ac\_t** (p. 1318), **dbasync\_native::db\_if\_t** (p. 374), **ac2isl::ac2isl\_t** (p. 163), **audiometerbackend::audiometer\_if\_t** (p. 316), **noise\_psd\_estimator::noise\_psd\_estimator\_if\_t** (p. 1309), **isl2ac::isl2ac\_t** (p. 679), **fftfilter::interface\_t** (p. 498), **smooth\_cepstrum::smooth\_cepstrum\_if\_t** (p. 1436), **rohBeam::rohBeam** (p. 1397), **multibandcompressor::interface\_t** (p. 1301), **dc\_simple::dc\_if\_t** (p. 393), **combc\_if\_t** (p. 356), **gtfb\_analyzer::gtfb\_analyzer\_t** (p. 551), **rmslevel::rmslevel\_if\_t** (p. 1389), **coherence::cohfilt\_if\_t** (p. 348), **plugin\_interface\_t** (p. 1351), **smoothgains\_bridge::overlapadd\_if\_t** (p. 1450), **example6\_t** (p. 479), **cpupload::cpupload\_if\_t** (p. 367), **noise\_t** (p. 1315), **MHAPplugin\_Resampling::resampling\_if\_t** (p. 1154), **shadowfilter\_end::shadowfilter\_end\_t** (p. 1428), **ac2wave\_if\_t** (p. 187), **adaptive\_feedback\_canceller** (p. 242), **nlims\_t** (p. 1306), **fshift\_hilbert::frequency\_translator\_t** (p. 513), **spec2wave\_if\_t** (p. 1462), **level\_matching::level\_matching\_t** (p. 652), **fader\_wave::fader\_wave\_if\_t** (p. 486), **overlapadd::overlapadd\_if\_t** (p. 1328), **acsave::acsave\_t** (p. 220), **mhachain::chain\_base\_t** (p. 840), **complex\_scale\_channel\_t** (p. 362), **doasvm\_feature\_extraction** (p. 424), **shadowfilter\_begin::shadowfilter\_begin\_t** (p. 1424), **fshift::fshift\_t** (p. 509), **lpc\_bl\_predictor** (p. 662), **lpc\_burglattice** (p. 668), **acPooling\_wave** (p. 212), **delaysum::delaysum\_wave\_if\_t** (p. 410), **sine\_t** (p. 1432), **steerbf** (p. 1471), **db\_if\_t** (p. 370), **fader\_if\_t** (p. 484), **lpc** (p. 658), **acConcat\_wave** (p. 200), **gain::gain\_if\_t** (p. 531), **acSteer** (p. 229), **acTransform\_wave** (p. 235), **doasvm\_classification** (p. 419), **fftfilterbank::fftfb\_interface\_t** (p. 501), **route::interface\_t** (p. 1409), **matrixmixer::matmix\_t** (p. 711), **equalize::freqgains\_t** (p. 461), **altplugs\_t** (p. 301), **delaysum\_spec::delaysum\_spec\_if\_t** (p. 415), **softclip\_t** (p. 1455), **ac2osc\_t** (p. 182), **ac\_proc::interface\_t** (p. 197), **gsc\_adaptive\_stage::gsc\_adaptive\_stage\_if** (p. 543), **acmon::acmon\_t** (p. 208), **example7\_t** (p. 482), **dropgen\_t** (p. 434), **identity\_t** (p. 573), **levelmeter\_t** (p. 655), **delay::interface\_t** (p. 408), **ds\_t** (p. 441), and **us\_t** (p. 1486).

```
5.332.3.2 release() template<class runtime_cfg_t >
void MHAPplugin::plugin_t< runtime_cfg_t >::release [virtual]
```

Reimplemented in **windnoise::if\_t** (p. 1511), **attenuate20\_t** (p. 314), **smooth\_cepstrum::smooth\_cepstrum\_if\_t** (p. 1436), **rohBeam::rohBeam** (p. 1397), **adaptive\_feedback\_canceller** (p. 242), **level\_matching::level\_matching\_t** (p. 652), **doasvm\_feature\_extraction** (p. 425), **fshift::fshift\_t** (p. 510), **example3\_t** (p. 471), **example4\_t** (p. 476), **lpc\_bl\_predictor** (p. 662), **lpc\_burglattice** (p. 668), **acPooling\_wave** (p. 213),

**steerbf** (p. 1472), **droptect\_t** (p. 438), **lpc** (p. 659), **acConcat\_wave** (p. 201), **ac←Steer** (p. 230), **acTransform\_wave** (p. 236), **example2\_t** (p. 468), **doasvm\_classification** (p. 419), **example1\_t** (p. 464), **gsc\_adaptive\_stage::gsc\_adaptive\_stage\_if** (p. 543), **example7\_t** (p. 482), **bbcalib\_interface\_t** (p. 335), **calibrator\_t** (p. 339), **gtfb\_simple\_t** (p. 568), **addsndfile::addsndfile\_if\_t** (p. 252), **adm\_if\_t** (p. 273), **analysisispath\_if\_t** (p. 311), **ac2lsl::ac2lsl\_t** (p. 164), **osc2ac\_t** (p. 1318), **dbasync\_native::db\_if\_t** (p. 375), **matlab\_wrapper::matlab\_wrapper\_t** (p. 697), **lsl2ac::lsl2ac\_t** (p. 679), **multibandcompressor::interface\_t** (p. 1301), **wave2spec\_if\_t** (p. 1489), **dc\_simple::dc\_if\_t** (p. 393), **plugins::hoertech::acrec::acrec\_t** (p. 1376), **coherence::cohflt\_if\_t** (p. 348), **smoothgains\_bridge::overlapadd\_if\_t** (p. 1450), **MHAPlugin\_Resampling::resampling\_if\_t** (p. 1155), **ac2wave\_if\_t** (p. 187), **nlms\_t** (p. 1306), **fshift\_hilbert::frequency\_translator\_t** (p. 513), **spec2wave\_if\_t** (p. 1462), **fader\_wave::fader\_wave\_if\_t** (p. 486), **overlapadd::overlapadd\_if\_t** (p. 1328), **acsave::acsave\_t** (p. 220), **mhachain::chain\_base\_t** (p. 840), **delaysum::delaysum\_wave\_if\_t** (p. 411), **wavrec\_t** (p. 1499), **altconfig\_t** (p. 297), **db\_if\_t** (p. 370), **gain::gain\_if\_t** (p. 531), **fftfilterbank::fftfb\_interface\_t** (p. 501), **route::interface\_t** (p. 1410), **ac2osc\_t** (p. 182), **altplugs\_t** (p. 301), **mconv::MConv** (p. 715), **ac\_proc::interface\_t** (p. 197), **acmon::acmon\_t** (p. 208), **dropgen\_t** (p. 434), **identity\_t** (p. 573), **ds\_t** (p. 441), and **us\_t** (p. 1486).

### 5.332.3.3 **prepare\_()** template<class runtime\_cfg\_t >

```
void MHAPlugin::plugin_t< runtime_cfg_t >::prepare_
    (mhaconfig_t & cf )
```

### 5.332.3.4 **release\_()** template<class runtime\_cfg\_t >

```
void MHAPlugin::plugin_t< runtime_cfg_t >::release_
```

### 5.332.3.5 **is\_prepared()** template<class runtime\_cfg\_t >

```
bool MHAPlugin::plugin_t< runtime_cfg_t >::is_prepared ( ) const [inline]
```

Flag, if the prepare method is successfully called (or currently evaluated)

### 5.332.3.6 **input\_cfg()** template<class runtime\_cfg\_t >

```
mhaconfig_t MHAPlugin::plugin_t< runtime_cfg_t >::input_cfg ( ) const [inline]
```

Current input channel configuration.

---

**5.332.3.7 `output_cfg()`** template<class runtime\_cfg\_t >  
`mhaconfig_t MHAPlugin::plugin_t< runtime_cfg_t >::output_cfg( ) const [inline]`

Current output channel configuration.

#### 5.332.4 Member Data Documentation

**5.332.4.1 `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. 847) are stored in this member in the **prepare()** (p. 1149) method.

##### Note

This member is likely to be removed in later versions, use **input\_cfg()** (p. 1151) and **output\_cfg()** (p. 1151) instead.

**5.332.4.2 `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.

**5.332.4.3 `is_prepared_`** template<class runtime\_cfg\_t >  
`bool MHAPlugin::plugin_t< runtime_cfg_t >::is_prepared_ [private]`

**5.332.4.4 `input_cfg_`** template<class runtime\_cfg\_t >  
`mhaconfig_t MHAPlugin::plugin_t< runtime_cfg_t >::input_cfg_ [private]`

## **5.333 MHAPlugin\_Resampling::resampling\_if\_t Class Reference**

**5.332.4.5 output\_cfg\_** template<class runtime\_cfg\_t >  
mhaconfig\_t MHAPlugin::plugin\_t< runtime\_cfg\_t >::output\_cfg\_ [private]

**5.332.4.6 mhaconfig\_in** template<class runtime\_cfg\_t >  
MHAParser::mhaconfig\_mon\_t MHAPlugin::plugin\_t< runtime\_cfg\_t >::mhaconfig\_in [private]

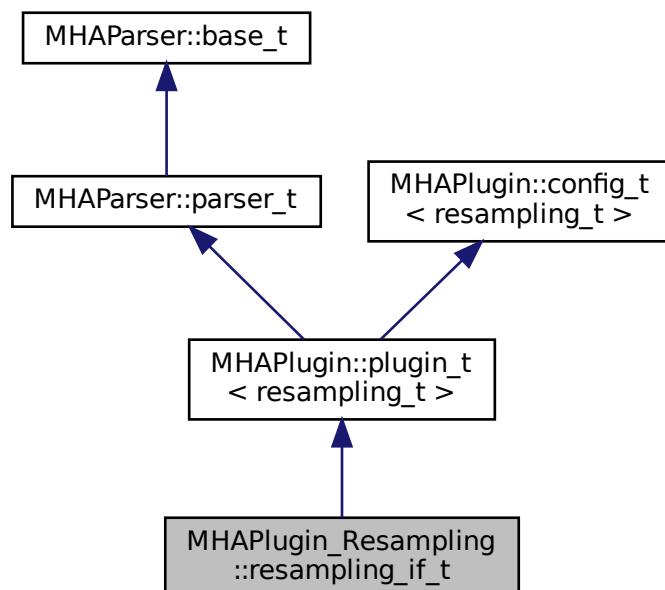
**5.332.4.7 mhaconfig\_out** template<class runtime\_cfg\_t >  
MHAParser::mhaconfig\_mon\_t MHAPlugin::plugin\_t< runtime\_cfg\_t >::mhaconfig\_out  
[private]

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

- **mha\_plugin.hh**

## **5.333 MHAPlugin\_Resampling::resampling\_if\_t Class Reference**

Inheritance diagram for MHAPlugin\_Resampling::resampling\_if\_t:



## Public Member Functions

- **resampling\_if\_t** (**algo\_comm\_t** iac, const std::string &configured\_name)
- **mha\_wave\_t \* process** (**mha\_wave\_t** \*)
- **void prepare** (**mhaconfig\_t** &)
- **void release** ()

## Private Attributes

- **MHAParser::float\_t srate**
- **MHAParser::int\_t fragsize**
- **MHAParser::float\_t nyquist\_ratio**
- **MHAParser::float\_t irslen\_outer2inner**
- **MHAParser::float\_t irslen\_inner2outer**
- **MHAParser::mhapluginloader\_t plugloader**
- std::string **algo**

## Additional Inherited Members

### 5.333.1 Constructor & Destructor Documentation

**5.333.1.1 resampling\_if\_t()** MHAPlugin\_Resampling::resampling\_if\_t::resampling\_if\_t  
(  
    **algo\_comm\_t** iac,  
    const std::string & configured\_name )

### 5.333.2 Member Function Documentation

**5.333.2.1 process()** **mha\_wave\_t** \* MHAPlugin\_Resampling::resampling\_if\_t::process (   
    **mha\_wave\_t** \* s )

## **5.333 MHAPlugin\_Resampling::resampling\_if\_t Class Reference**

**5.333.2.2 `prepare()`** `void MHAPlugin_Resampling::resampling_if_t::prepare ( mhaconfig_t & conf ) [virtual]`

Implements `MHAPlugin::plugin_t< resampling_t >` (p. 1149).

**5.333.2.3 `release()`** `void MHAPlugin_Resampling::resampling_if_t::release ( ) [virtual]`

Reimplemented from `MHAPlugin::plugin_t< resampling_t >` (p. 1150).

### **5.333.3 Member Data Documentation**

**5.333.3.1 `srate`** `MHAParser::float_t MHAPlugin_Resampling::resampling_if_t::srate [private]`

**5.333.3.2 `fragsize`** `MHAParser::int_t MHAPlugin_Resampling::resampling_if_t::fragsize [private]`

**5.333.3.3 `nyquist_ratio`** `MHAParser::float_t MHAPlugin_Resampling::resampling_if_t::nyquist_ratio [private]`

**5.333.3.4 `irslen_outer2inner`** `MHAParser::float_t MHAPlugin_Resampling::resampling_if_t::irslen_outer2inner [private]`

**5.333.3.5 `irslen_inner2outer`** `MHAParser::float_t MHAPlugin_Resampling::resampling_if_t::irslen_inner2outer [private]`

**5.333.3.6 plugloader** `MHAParser::mhapluginloader_t` `MHAParser::resampling_if_t::plugloader` [private]

**5.333.3.7 algo** `std::string` `MHAParser::resampling_if_t::algo` [private]

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

- `resampling.cpp`

## 5.334 MHAParser::resampling\_t Class Reference

### Public Member Functions

- `resampling_t` (unsigned int `outer_fragsize`, float `outer_srate`, unsigned int `inner_fragsize`, float `inner_srate`, unsigned int `nch_in`, float `filter_length_in`, unsigned int `nch_out`, float `filter_length_out`, float `nyquist_ratio`, `MHAParser::mhapluginloader_t &plug`)
- `mha_wave_t * process ( mha_wave_t *)`

### Private Attributes

- unsigned `outer_fragsize`
- unsigned `inner_fragsize`
- float `outer_srate`
- float `inner_srate`
- unsigned `nchannels_in`
- unsigned `nchannels_out`
- `MHAFilter::blockprocessing_polyphase_resampling_t outer2inner_resampling`
- `MHAFilter::blockprocessing_polyphase_resampling_t inner2outer_resampling`
- `MHAParser::mhapluginloader_t & plugloader`
- `MHASignal::waveform_t inner_signal`
- `MHASignal::waveform_t output_signal`

### 5.334.1 Constructor & Destructor Documentation

**5.334.1.1 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::mha_pluginloader_t & plug )
```

## **5.334.2 Member Function Documentation**

**5.334.2.1 process()** `mha_wave_t * MHAPlugin_Resampling::resampling_t::process (`

```
mha_wave_t * s )
```

## **5.334.3 Member Data Documentation**

**5.334.3.1 outer\_fragsize** `unsigned MHAPlugin_Resampling::resampling_t::outer_fragsize`  
[private]

**5.334.3.2 inner\_fragsize** `unsigned MHAPlugin_Resampling::resampling_t::inner_fragsize`  
[private]

**5.334.3.3 outer\_srate** `float MHAPlugin_Resampling::resampling_t::outer_srate` [private]

**5.334.3.4 inner\_srate** `float MHAPlugin_Resampling::resampling_t::inner_srate` [private]

**5.334.3.5 nchannels\_in** `unsigned MHAPlugIn_Resampling::resampling_t::nchannels_in [private]`

**5.334.3.6 nchannels\_out** `unsigned MHAPlugIn_Resampling::resampling_t::nchannels_out [private]`

**5.334.3.7 outer2inner\_resampling** `MHAFilter::blockprocessing_polyphase_resampling_t MHAPlugIn_Resampling::resampling_t::outer2inner_resampling [private]`

**5.334.3.8 inner2outer\_resampling** `MHAFilter::blockprocessing_polyphase_resampling_t MHAPlugIn_Resampling::resampling_t::inner2outer_resampling [private]`

**5.334.3.9 plugloader** `MHAParser::mhapluginloader_t& MHAPlugIn_Resampling::resampling_t::plugloader [private]`

**5.334.3.10 inner\_signal** `MHASignal::waveform_t MHAPlugIn_Resampling::resampling_t::inner_signal [private]`

**5.334.3.11 output\_signal** `MHASignal::waveform_t MHAPlugIn_Resampling::resampling_t::output_signal [private]`

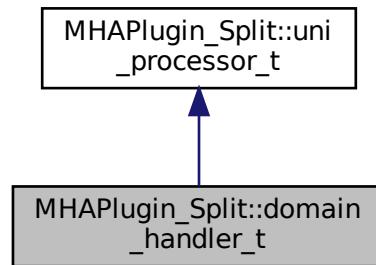
The documentation for this class was generated from the following file:

- `resampling.cpp`

## 5.335 MHAPlugin\_Split::domain\_handler\_t Class Reference

Handles domain-specific partial input and output signal.

Inheritance diagram for MHAPlugin\_Split::domain\_handler\_t:



### Public Member Functions

- void **set\_input\_domain** (const **mhaconfig\_t** &settings\_in)  
*Set parameters of input signal.*
- void **set\_output\_domain** (const **mhaconfig\_t** &settings\_out)  
*Set output signal parameters.*
- void **deallocate\_domains** ()  
*Deallocate domain indicators and signal holders.*
- **domain\_handler\_t** (const **mhaconfig\_t** &settings\_in, const **mhaconfig\_t** &settings\_out, **PluginLoader::fourway\_processor\_t** \*processor)  
*Construct a new domain handler once the domains and dimensions of input and output signal of one of the child plugins of split are known.*
- virtual ~**domain\_handler\_t** ()  
*Deallocation of signal holders.*
- unsigned **put\_signal** (**mha\_wave\_t** \*s\_in, unsigned start\_channel)  
*Store the relevant channels from the input signal for processing.*
- unsigned **put\_signal** (**mha\_spec\_t** \*s\_in, unsigned start\_channel)  
*Store the relevant channels from the input signal for processing.*
- unsigned **get\_signal** (**MHASignal::waveform\_t** \*s\_out, unsigned start\_channel)  
*Store all partial signal output channels in the combined waveform signal with the given channel offset.*
- unsigned **get\_signal** (**MHASignal::spectrum\_t** \*s\_out, unsigned start\_channel)  
*Store all partial signal output channels in the combined spectrum signal with the given channel offset.*
- void **process** ()  
*Call the processing method of the processor with configured input/output signal domains.*

## Public Attributes

- **MHASignal::waveform\_t \* wave\_in**  
*Partial wave input signal.*
- **mha\_wave\_t \*\* wave\_out**  
*Partial wave output signal.*
- **MHASignal::spectrum\_t \* spec\_in**  
*Partial spec input signal.*
- **mha\_spec\_t \*\* spec\_out**  
*Partial spec input signal.*
- **PluginLoader::fourway\_processor\_t \* processor**  
*The domain-specific signal processing methods are implemented here.*

## Private Member Functions

- **domain\_handler\_t (const domain\_handler\_t &)**  
*Disallow copy constructor.*
- **domain\_handler\_t & operator= (const domain\_handler\_t &)**  
*Disallow assignment operator.*

### 5.335.1 Detailed Description

Handles domain-specific partial input and output signal.

### 5.335.2 Constructor & Destructor Documentation

```
5.335.2.1 domain_handler_t() [1/2] MHAPlugin_Split::domain_handler_t::domain_←
handler_t (
    const domain_handler_t & ) [private]
```

Disallow copy constructor.

```
5.335.2.2 domain_handler_t() [2/2] MHAPlugin_Split::domain_handler_t::domain_handler_t (
    const mhaconfig_t & settings_in,
    const mhaconfig_t & settings_out,
    PluginLoader::fourway_processor_t * processor ) [inline]
```

Construct a new domain handler once the domains and dimensions of input and output signal of one of the child plugins of split are known.

```
5.335.2.3 ~domain_handler_t() virtual MHAPlugin_Split::domain_handler_t::~domain_handler_t ( ) [inline], [virtual]
```

Deallocation of signal holders.

### 5.335.3 Member Function Documentation

```
5.335.3.1 operator=() domain_handler_t& MHAPlugin_Split::domain_handler_t::operator=
(
    const domain_handler_t & ) [private]
```

Disallow assignment operator.

```
5.335.3.2 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
--------------------	---

```
5.335.3.3 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
---------------------	--

```
5.335.3.4 deallocate_domains() void MHAPlugin_Split::domain_handler_t::deallocate<-
_domains ( ) [inline]
```

Deallocate domain indicators and signal holders.

```
5.335.3.5 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. 1164).

#### 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

```
5.335.3.6 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. 1165).

#### 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

```
5.335.3.7 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. 1164) 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

```
5.335.3.8 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. 1165) 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

**5.335.3.9 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. 1162) before this method may be called.

Implements **MHAPlugin\_Split::uni\_processor\_t** (p. 1188).

### 5.335.4 Member Data Documentation

**5.335.4.1 wave\_in** MHASignal::waveform\_t\* MHAPlugin\_Split::domain\_handler\_t::wave\_in

Partial wave input signal.

**5.335.4.2 wave\_out mha\_wave\_t\*\* MHAPlugin\_Split::domain\_handler\_t::wave\_out**

Partial wave output signal.

**5.335.4.3 spec\_in MHASignal::spectrum\_t\* MHAPlugin\_Split::domain\_handler\_t::spec\_in**

Partial spec input signal.

**5.335.4.4 spec\_out mha\_spec\_t\*\* MHAPlugin\_Split::domain\_handler\_t::spec\_out**

Partial spec input signal.

**5.335.4.5 processor PluginLoader::fourway\_processor\_t\* MHAPlugin\_Split::domain\_handler\_t::processor**

The domain-specific signal processing methods are implemented here.

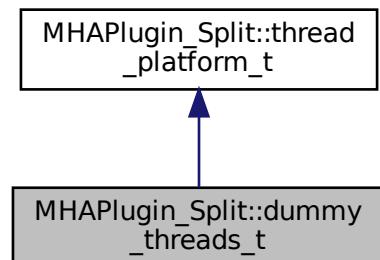
The documentation for this class was generated from the following file:

- **split.cpp**

**5.336 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

### 5.336.1 Detailed Description

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

### 5.336.2 Constructor & Destructor Documentation

**5.336.2.1 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

### 5.336.3 Member Function Documentation

**5.336.3.1 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).

**5.336.3.2 catch\_thread()** void MHAPlugin\_Split::dummy\_threads\_t::catch\_thread ( ) [inline], [virtual]

No implementation needed: Processing has been completed during dummy\_threads\_t::kick\_thread.

Implements **MHAPlugin\_Split::thread\_platform\_t** (p. 1186).

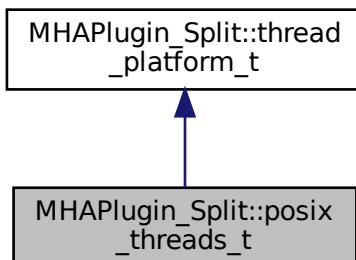
The documentation for this class was generated from the following file:

- **split.cpp**

## 5.337 MHAPlugin\_Split::posix\_threads\_t Class Reference

Posix threads specification of thread platform.

Inheritance diagram for MHAPlugin\_Split::posix\_threads\_t:



## Public Member Functions

- void **kick\_thread** ()
 

*Start signal processing in separate thread.*
- void **catch\_thread** ()
 

*Wait for signal processing to finish.*
- **posix\_threads\_t** ( **uni\_processor\_t** \*proc, const std::string &thread\_scheduler, int thread\_priority)
 

*Constructor.*
- **~posix\_threads\_t** ()
 

*Terminate thread.*
- void **main** ()
 

*Thread main loop. Wait for process/termination trigger, then act.*

## Static Public Member Functions

- static void \* **thread\_start** (void \*thr)
 

*Thread start function.*
- static std::string **current\_thread\_scheduler** ()
- static int **current\_thread\_priority** ()

## Private Attributes

- pthread\_mutex\_t **mutex**

*The mutex.*
- pthread\_cond\_t **kick\_condition**

*The condition for signalling the kicking and termination.*
- pthread\_cond\_t **catch\_condition**

*The condition for signalling the processing is finished.*
- pthread\_attr\_t **attr**

*Thread attributes.*
- struct sched\_param **priority**

*Thread scheduling priority.*
- int **scheduler**
- pthread\_t **thread**

*The thread object.*
- bool **kicked**

*A flag that is set to true by kick\_thread and to false by the thread after it has woken up from the kicking.*
- bool **processing\_done**

*A flag that is set to true by the thread when it returns from processing and to false by catch\_thread after it has waited for that return.*
- bool **termination\_request**

*Set to true by the destructor.*

## Additional Inherited Members

### 5.337.1 Detailed Description

Posix threads specification of thread platform.

### 5.337.2 Constructor & Destructor Documentation

```
5.337.2.1 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.

```
5.337.2.2 ~posix_threads_t() MHAPlugin_Split::posix_threads_t::~posix_threads_t (
) [inline]
```

Terminate thread.

### 5.337.3 Member Function Documentation

```
5.337.3.1 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).

**5.337.3.2 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).

**5.337.3.3 thread\_start()** static void\* MHAPlugin\_Split::posix\_threads\_t::thread\_start  
(  
    void \* thr ) [inline], [static]

Thread start function.

**5.337.3.4 main()** void MHAPlugin\_Split::posix\_threads\_t::main ( ) [inline]

Thread main loop. Wait for process/termination trigger, then act.

**5.337.3.5 current\_thread\_scheduler()** static std::string MHAPlugin\_Split::posix\_←  
threads\_t::current\_thread\_scheduler ( ) [inline], [static]

**5.337.3.6 current\_thread\_priority()** static int MHAPlugin\_Split::posix\_threads\_t←  
::current\_thread\_priority ( ) [inline], [static]

## 5.337.4 Member Data Documentation

**5.337.4.1 mutex** pthread\_mutex\_t MHAPlugin\_Split::posix\_threads\_t::mutex [private]

The mutex.

**5.337.4.2 kick\_condition** pthread\_cond\_t MHAPlugin\_Split::posix\_threads\_t::kick←  
condition [private]

The condition for signalling the kicking and termination.

**5.337.4.3 catch\_condition** pthread\_cond\_t MHAPlugin\_Split::posix\_threads\_t::catch←  
\_condition [private]

The condition for signalling the processing is finished.

**5.337.4.4 attr** pthread\_attr\_t MHAPlugin\_Split::posix\_threads\_t::attr [private]

Thread attributes.

**5.337.4.5 priority** struct sched\_param MHAPlugin\_Split::posix\_threads\_t::priority  
[private]

Thread scheduling priority.

**5.337.4.6 scheduler** int MHAPlugin\_Split::posix\_threads\_t::scheduler [private]

**5.337.4.7 thread** pthread\_t MHAPlugin\_Split::posix\_threads\_t::thread [private]

The thread object.

**5.337.4.8 kicked** bool MHAPlugin\_Split::posix\_threads\_t::kicked [private]

A flag that is set to true by kick\_thread and to false by the thread after it has woken up from the kicking.

**5.337.4.9 processing\_done** bool MHAParser::base\_t::processing\_done  
[private]

A flag that is set to true by the thread when it returns from processing and to false by catch\_←  
thread after it has waited for that return.

**5.337.4.10 termination\_request** bool MHAParser::base\_t::termination←  
\_request [private]

Set to true by the destructor.

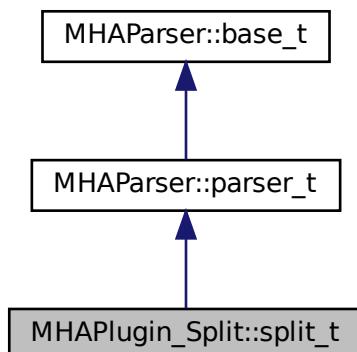
The documentation for this class was generated from the following file:

- **split.cpp**

## 5.338 MHAParser::base\_t Class Reference

Implements split plugin.

Inheritance diagram for MHAParser::base\_t:



## Public Member Functions

- **split\_t ( algo\_comm\_t iac, const std::string &configured\_name)**  
*Plugin constructor.*
- **~split\_t ()**  
*Plugin destructor. Unloads nested plugins.*
- **void prepare\_ ( mhaconfig\_t &)**  
*Check signal parameters, prepare chains, and allocate output signal holders.*
- **void release\_ ()**  
*Delete output signal holder and release chains.*
- template<class SigTypeIn , class SigTypeOut >  
**void process (SigTypeIn \*, SigTypeOut \*\*)**  
*Let the parallel plugins process channel groups of the input signal.*

## Private Member Functions

- **void update ()**  
*Load plugins in response to a value change in the algos variable.*
- **void clear\_chains ()**  
*Unload the plugins.*
- **mha\_wave\_t \* copy\_output\_wave ()**
- **mha\_spec\_t \* copy\_output\_spec ()**
- template<class SigType >  
**void trigger\_processing (SigType \*s\_in)**  
*Split the argument input signal to groups of channels for the plugins and initiate signal processing.*
- template<class SigType >  
**void collect\_result (SigType \*s\_out)**  
*Combine the output signal from the plugins.*
- **MHASignal::waveform\_t \* signal\_out ( mha\_wave\_t \*\*)**  
*Waveform domain output signal structure accessor.*
- **MHASignal::spectrum\_t \* signal\_out ( mha\_spec\_t \*\*)**  
*Spectrum domain output signal structure. Parameter is ignored.*

## Private Attributes

- **MHAEvents::patchbay\_t< split\_t > patchbay**  
*Reload plugins when the algos variable changes.*
- **MHAParser::vstring\_t algos**  
*Vector of plugins to load in parallel.*
- **MHAParser::vint\_t channels**  
*Number of channels to route through each plugin.*
- **MHAParser::kw\_t thread\_platform**  
*Thread platform chooser.*

- **MHAParser::kw\_t worker\_thread\_scheduler**  
*Scheduler used for worker threads.*
- **MHAParser::int\_t worker\_thread\_priority**  
*Priority of worker threads.*
- **MHAParser::string\_mon\_t framework\_thread\_scheduler**  
*Scheduler of the signal processing thread.*
- **MHAParser::int\_mon\_t framework\_thread\_priority**  
*Priority of signal processing thread.*
- **MHAParser::bool\_t delay**  
*Switch to activate parallel processing of plugins at the cost of one block of additional delay.*
- std::vector<  **splitted\_part\_t \* > chains**  
*Interfaces to parallel plugins.*
- **MHASignal::waveform\_t \* wave\_out**  
*Combined output waveforms structure.*
- **MHASignal::spectrum\_t \* spec\_out**  
*Combined output spectra structure.*

## Additional Inherited Members

### 5.338.1 Detailed Description

Implements split plugin.

An instance of class **split\_t** (p. 1172) implements the split plugin functionality: The audio channels are splitted and groups of audio channels are processed by different plugins in parallel.

### 5.338.2 Constructor & Destructor Documentation

```
5.338.2.1 split_t() MHAPlugIn_Split::split_t::split_t (
    algo_comm_t iac,
    const std::string & configured_name )
```

Plugin constructor.

```
5.338.2.2 ~split_t() MHAPlugIn_Split::split_t::~split_t ( )
```

Plugin destructor. Unloads nested plugins.

### 5.338.3 Member Function Documentation

**5.338.3.1 `prepare_()`** `void MHAPlugin_Split::split_t::prepare_ ( mhaconfig_t & signal_parameters )`

Check signal parameters, prepare chains, and allocate output signal holders.

**5.338.3.2 `release_()`** `void MHAPlugin_Split::split_t::release_ ( )`

Delete output signal holder and release chains.

**5.338.3.3 `process()`** `template<class SigTypeIn , class SigTypeOut >`  
`void MHAPlugin_Split::split_t::process (`  
    `SigTypeIn * s_in,`  
    `SigTypeOut ** s_out )`

Let the parallel plugins process channel groups of the input signal.

**5.338.3.4 `update()`** `void MHAPlugin_Split::split_t::update ( ) [private]`

Load plugins in response to a value change in the algos variable.

**5.338.3.5 `clear_chains()`** `void MHAPlugin_Split::split_t::clear_chains ( ) [private]`

Unload the plugins.

**5.338.3.6 `copy_output_wave()`** `mha_wave_t* MHAPlugin_Split::split_t::copy_output_<wave ( ) [private]`

---

**5.338.3.7 `copy_output_spec()`** `mha_spec_t* MHAPlugin_Split::split_t::copy_output_`←  
`spec ( ) [private]`

**5.338.3.8 `trigger_processing()`** `template<class SigType >`  
`void MHAPlugin_Split::split_t::trigger_processing (`  
`SigType * s_in ) [private]`

Split the argument input signal to groups of channels for the plugins and initiate signal processing.

**5.338.3.9 `collect_result()`** `template<class SigType >`  
`void MHAPlugin_Split::split_t::collect_result (`  
`SigType * s_out ) [private]`

Combine the output signal from the plugins.

**5.338.3.10 `signal_out()` [1/2]** `MHASignal::waveform_t* MHAPlugin_Split::split_t::`←  
`:signal_out (`  
`mha_wave_t ** ) [inline], [private]`

Waveform domain output signal structure accessor.

Parameter is only for domain disambiguation and is ignored.

**5.338.3.11 `signal_out()` [2/2]** `MHASignal::spectrum_t* MHAPlugin_Split::split_t::`←  
`:signal_out (`  
`mha_spec_t ** ) [inline], [private]`

Spectrum domain output signal structure. Parameter is ignored.

## 5.338.4 Member Data Documentation

**5.338.4.1 patchbay** `MHAEEvents::patchbay_t< split_t> MHAPlugin_Split::split_t::patchbay [private]`

Reload plugins when the algos variable changes.

**5.338.4.2 algos** `MHAParser::vstring_t MHAPlugin_Split::split_t::algos [private]`

Vector of plugins to load in parallel.

**5.338.4.3 channels** `MHAParser::vint_t MHAPlugin_Split::split_t::channels [private]`

Number of channels to route through each plugin.

**5.338.4.4 thread\_platform** `MHAParser::kw_t MHAPlugin_Split::split_t::thread_platform [private]`

Thread platform chooser.

**5.338.4.5 worker\_thread\_scheduler** `MHAParser::kw_t MHAPlugin_Split::split_t::worker_thread_scheduler [private]`

Scheduler used for worker threads.

**5.338.4.6 worker\_thread\_priority** `MHAParser::int_t MHAPlugin_Split::split_t::worker_thread_priority [private]`

Priority of worker threads.

**5.338.4.7 framework\_thread\_scheduler** `MHAParser::string_mon_t MHAParser::string_mon_t MHAPlugin_Split::split_t::framework_thread_scheduler [private]`

Scheduler of the signal processing thread.

**5.338.4.8 framework\_thread\_priority** `MHAParser::int_mon_t MHAPlugin_Split::split_t::framework_thread_priority [private]`

Priority of signal processing thread.

**5.338.4.9 delay** `MHAParser::bool_t MHAPlugin_Split::split_t::delay [private]`

Switch to activate parallel processing of plugins at the cost of one block of additional delay.

**5.338.4.10 chains** `std::vector< splitted_part_t*> MHAPlugin_Split::split_t::chains [private]`

Interfaces to parallel plugins.

**5.338.4.11 wave\_out** `MHASignal::waveform_t* MHAPlugin_Split::split_t::wave_out [private]`

Combined output waveforms structure.

**5.338.4.12 spec\_out** `MHASignal::spectrum_t* MHAPlugin_Split::split_t::spec_out [private]`

Combined output spectra structure.

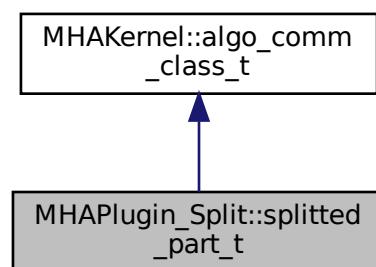
The documentation for this class was generated from the following file:

- `split.cpp`

## 5.339 MHAPlugin\_Split::splitted\_part\_t Class Reference

The **splitted\_part\_t** (p. 1179) instance manages the plugin that performs processing on the reduced set of channels.

Inheritance diagram for MHAPlugin\_Split::splitted\_part\_t:



### Public Member Functions

- **splitted\_part\_t** (const std::string &plugname, **MHAParser::parser\_t** \*parent)  
*Load the plugin for this partial signal path.*
- **splitted\_part\_t** ( **PluginLoader::fourway\_processor\_t** \*plugin)  
*Create the handler for the partial signal.*
- ~**splitted\_part\_t** () throw ()  
*Destructor. Deletes the plugin **plug** (p. 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. 1159) instance.*
- void **release** ()  
*Delegates the release method to the plugin and deletes the **MHAPlugin\_Split::domain\_handler\_t** (p. 1159) 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

### 5.339.1 Detailed Description

The **splitted\_part\_t** (p. 1179) instance manages the plugin that performs processing on the reduced set of channels.

The signal is split by channels by this instance, but the signal is combined again by the calling class.

### 5.339.2 Constructor & Destructor Documentation

#### 5.339.2.1 **splitted\_part\_t()** [1/3] `MHAParser_Split::splitted_part_t::splitted_part_t ( const splitted_part_t & ) [private]`

Disallow copy constructor.

#### 5.339.2.2 **splitted\_part\_t()** [2/3] `MHAParser_Split::splitted_part_t::splitted_part_t ( const std::string & plugname, MHAParser::parser_t * parent )`

Load the plugin for this partial signal path.

Loads the MHA plugin for a signal path of these audio channels.

### Parameters

<i>plugname</i>	The name of the MHA plugin, optionally followed by a colon and the algorithm name.
<i>parent</i>	The parser node where the configuration of the new plugin is inserted. The plugin's parser name is the configured name (colon syntax).

**5.339.2.3 `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

<i>plugin</i>	The plugin used for processing the signal. The new <code>splitted_part_t</code> (p. 1180) instance will take ownership of this instance and release it in the destructor.
---------------	---

**5.339.2.4 `~splitted_part_t()`** `MHAPlugin_Split::splitted_part_t::~splitted_part_t ( ) throw ( )`

Destructor. Deletes the plugin `plug` (p. 1183).

## 5.339.3 Member Function Documentation

**5.339.3.1 `operator=()`** `splitted_part_t& MHAPlugin_Split::splitted_part_t::operator= ( const splitted_part_t & ) [private]`

Disallow assignment operator.

```
5.339.3.2 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. 1159) instance.

Prepare the loaded plugin.

Plugin preparation.

#### Parameters

<i>signal_parameters</i>	The signal description parameters for this path.
<i>thread_platform</i>	The name of the thread platform to use. Possible values: "posix", "win32", "dummy".
<i>thread_scheduler</i>	The name of the scheduler to use. Posix threads support "SCHED_OTHER", "SCHED_RR", "SCHED_FIFO". The other thread platforms do not support different thread schedulers. This value is not used for platforms other than "posix".
<i>thread_priority</i>	The new thread priority. Interpretation and permitted range depend on the thread platform and possibly on the scheduler.

```
5.339.3.3 release() void MHAPlugIn_Split::splitted_part_t::release ( )
```

Delegates the release method to the plugin and deletes the **MHAPlugIn\_Split::domain\_handler\_t** (p. 1159) instance.

Release the loaded plugin.

Plugin release.

```
5.339.3.4 parse() std::string MHAPlugIn_Split::splitted_part_t::parse (
    const std::string & str ) [inline]
```

Delegates parser invocation to plugin.

```
5.339.3.5 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

**5.339.3.6 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

**5.339.4 Member Data Documentation**

**5.339.4.1 plug PluginLoader::fourway\_processor\_t\* MHAPlugin\_Split::splitted\_part\_t::plug [private]**

The plugin that performs the signal processing on the prepared channels.

### 5.339.4.2 domain `domain_handler_t*` `MHAPlugin_Split::splitted_part_t::domain` [private]

The domain specific signal handler, allocated from prepare when input and output domains and signal parameters are known.

### 5.339.4.3 thread `thread_platform_t*` `MHAPlugin_Split::splitted_part_t::thread` [private]

The platform-dependent thread synchronization implementation.

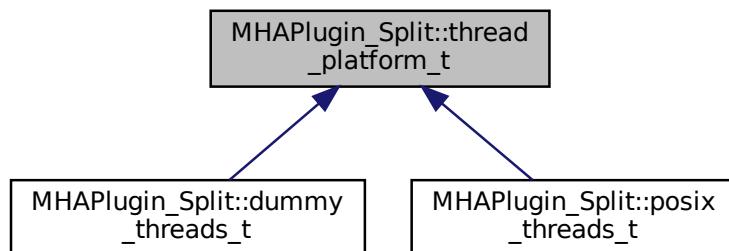
The documentation for this class was generated from the following file:

- `split.cpp`

## 5.340 MHAPlugin\_Split::thread\_platform\_t Class Reference

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

Inheritance diagram for MHAPlugin\_Split::thread\_platform\_t:



### Public Member Functions

- **thread\_platform\_t ( `uni_processor_t` \*proc)**  
*Constructor.*
- **virtual ~thread\_platform\_t ()**  
*Make derived classes destructable via pointer to this base class.*
- **virtual void kick\_thread ()=0**  
*Derived classes notify their processing thread that it should call processor->process().*
- **virtual void catch\_thread ()=0**  
*Derived classes wait for their signal processing thread to return from the call to part->process().*

## Protected Attributes

- `uni_processor_t * processor`

*A pointer to the plugin loader that processes the sound data in the channels for which this thread was created.*

## Private Member Functions

- `thread_platform_t (const thread_platform_t &)`  
*Disallow copy constructor.*
- `thread_platform_t & operator= (const thread_platform_t &)`  
*Disallow assignment operator.*

### 5.340.1 Detailed Description

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

Derived classes specialize in the actual thread platform.

### 5.340.2 Constructor & Destructor Documentation

```
5.340.2.1 thread_platform_t() [1/2] MHAPlugin_Split::thread_platform_t::thread_←
platform_t (
    const thread_platform_t & ) [private]
```

Disallow copy constructor.

```
5.340.2.2 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 <b>MHAPlugIn_Split::splitted_part_t</b> (p. 1179) instance.
-------------	--

**5.340.2.3 ~thread\_platform\_t()** virtual MHAPlugIn\_Split::thread\_platform\_t::~thread\_platform\_t ( ) [inline], [virtual]

Make derived classes destructable via pointer to this base class.

Derived classes' destructors notify the thread that it should terminate itself, and wait for the termination to occur.

## 5.340.3 Member Function Documentation

**5.340.3.1 operator=()** thread\_platform\_t& MHAPlugIn\_Split::thread\_platform\_t::operator= ( const thread\_platform\_t & ) [private]

Disallow assignment operator.

**5.340.3.2 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. 1169), and **MHAPlugIn\_Split::dummy\_threads\_t** (p. 1166).

**5.340.3.3 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. 1169), and **MHAPlugin\_Split::dummy\_threads\_t** (p. 1167).

#### 5.340.4 Member Data Documentation

**5.340.4.1 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).

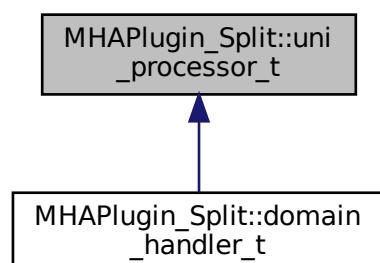
The documentation for this class was generated from the following file:

- **split.cpp**

#### 5.341 MHAPlugin\_Split::uni\_processor\_t Class Reference

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

Inheritance diagram for MHAPlugin\_Split::uni\_processor\_t:



## Public Member Functions

- virtual void **process ()=0**  
*This method uses some input signal, performs processing and stores the output signal somewhere.*
- virtual ~**uni\_processor\_t ()**  
*Classes containing virtual methods need virtual destructors.*

### 5.341.1 Detailed Description

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

No signal transfer occurs through this interface, because the signal transfer is performed in another thread than the processing.

### 5.341.2 Constructor & Destructor Documentation

**5.341.2.1 ~uni\_processor\_t()** virtual MHAPlugin\_Split::uni\_processor\_t::~uni\_processor\_t ( ) [inline], [virtual]

Classes containing virtual methods need virtual destructors.

### 5.341.3 Member Function Documentation

**5.341.3.1 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. [1164](#)).

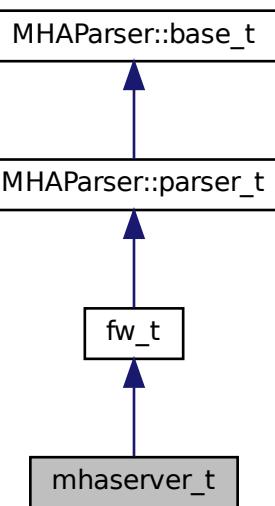
The documentation for this class was generated from the following file:

- **split.cpp**

## 5.342 mhaserver\_t Class Reference

MHA Framework listening on TCP port for commands.

Inheritance diagram for mhaserver\_t:



### Classes

- class **tcp\_server\_t**

### Public Member Functions

- **mhaserver\_t** (const std::string &ao, const std::string &af, const std::string &lf, bool b\_← interactive\_)
- **~mhaserver\_t ()**
- virtual std::string **on\_received\_line** (const std::string &line)
 

*A line of text was received from network client.*
- virtual void **acceptor\_started ()**

*Notification: "TCP port is open".*
- virtual void **send\_port\_announcement ()**

*sends an announcement which port this MHA is listening on to the creator of the process.*
- virtual void **start\_stdin\_thread ()**

*Starts a separate thread that reads lines from stdin and forwards these lines over TCP to the MHA configuration thread which multiplexes multiple TCP connections.*

- virtual void **set\_announce\_port** (unsigned short **announce\_port**)  
*If set to nonzero, the spawning process has asked to be notified of the TCP port used by this process.*
- void **logstring** (const std::string &)  
*Log a message to log file.*
- int **run** (unsigned short **port**, const std::string &\_interface)  
*Accept network connections and act on commands.*

## Public Attributes

- **MHAParser::int\_t port**

## Private Attributes

- std::shared\_ptr< **tcp\_server\_t** > **tcpserver**
- std::string **ack\_ok**
- std::string **ack\_fail**
- std::string **logfile**
- unsigned short **announce\_port**
- bool **b\_interactive**
- **MHAParser::int\_mon\_t pid\_mon**

## Additional Inherited Members

### 5.342.1 Detailed Description

MHA Framework listening on TCP port for commands.

### 5.342.2 Constructor & Destructor Documentation

#### 5.342.2.1 **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.

**5.342.2.2 ~mhaserver\_t()** `mhaserver_t::~mhaserver_t ( )`**5.342.3 Member Function Documentation****5.342.3.1 on\_received\_line()** `std::string mhaserver_t::on_received_line ( const std::string & line ) [virtual]`

A line of text was received from network client.

**5.342.3.2 acceptor\_started()** `void mhaserver_t::acceptor_started ( ) [virtual]`

Notification: "TCP port is open".

**5.342.3.3 send\_port\_announcement()** `void mhaserver_t::send_port_announcement ( ) [virtual]`

sends an announcement which port this MHA is listening on to the creator of the process.

See command line option –announce

**5.342.3.4 start\_stdin\_thread()** `void mhaserver_t::start_stdin_thread ( ) [virtual]`

Starts a separate thread that reads lines from stdin and forwards these lines over TCP to the MHA configuration thread which multiplexes multiple TCP connections.

Enables users to type mha configuration language commands directly into the terminal where MHA was started, without the need to use third-party tools like nc or putty.

**5.342.3.5 `set_announce_port()`** `void mhaserver_t::set_announce_port (`  
`unsigned short announce_port ) [virtual]`

If set to nonzero, the spawning process has asked to be notified of the TCP port used by this process.

**5.342.3.6 `logstring()`** `void mhaserver_t::logstring (`  
`const std::string & s ) [inline]`

Log a message to log file.

**5.342.3.7 `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

### 5.342.4 Member Data Documentation

**5.342.4.1 `tcpserver`** `std::shared_ptr< tcp_server_t> mhaserver_t::tcpserver [private]`

**5.342.4.2 `ack_ok`** `std::string mhaserver_t::ack_ok [private]`

**5.342.4.3 ack\_fail** std::string mhaserver\_t::ack\_fail [private]

**5.342.4.4 logfile** std::string mhaserver\_t::logfile [private]

**5.342.4.5 announce\_port** unsigned short mhaserver\_t::announce\_port [private]

**5.342.4.6 b\_interactive** bool mhaserver\_t::b\_interactive [private]

**5.342.4.7 pid\_mon** MHAParser::int\_mon\_t mhaserver\_t::pid\_mon [private]

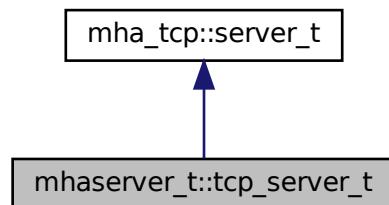
**5.342.4.8 port** MHAParser::int\_t mhaserver\_t::port

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

- **mhamain.cpp**

## 5.343 mhaserver\_t::tcp\_server\_t Class Reference

Inheritance diagram for mhaserver\_t::tcp\_server\_t:



## Public Member Functions

- **tcp\_server\_t** (const std::string &interface, uint16\_t **port**, **mhaserver\_t** \* **mha**)
- virtual bool **on\_received\_line** (std::shared\_ptr< **mha\_tcp::buffered\_socket\_t** > **c**, const std::string &**l**) override

*This method is invoked when a line of text is received on one of the accepted connections.*

## Private Attributes

- **mhaserver\_t** \* **mha**

### 5.343.1 Constructor & Destructor Documentation

```
5.343.1.1 tcp_server_t() mhaserver_t::tcp_server_t::tcp_server_t (
    const std::string & interface,
    uint16_t port,
    mhaserver_t * mha ) [inline]
```

### 5.343.2 Member Function Documentation

```
5.343.2.1 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

<b>c</b>	the connection that has received this line
<b>l</b>	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. 818).

### 5.343.3 Member Data Documentation

#### 5.343.3.1 mha\_mhaserver\_t\* mhaserver\_t::tcp\_server\_t::mha [private]

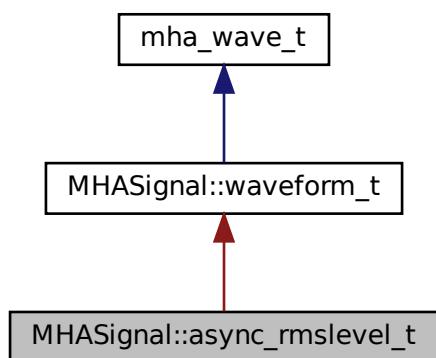
The documentation for this class was generated from the following file:

- [mhamain.cpp](#)

## 5.344 MHASignal::async\_rmslevel\_t Class Reference

Class for asynchronous level metering.

Inheritance diagram for MHASignal::async\_rmslevel\_t:



## Public Member Functions

- **async\_rmslevel\_t** (unsigned int *frames*, unsigned int **channels**)  
*Constructor for level metering class.*
- std::vector< float > **rmslevel** () const  
*Read-only function for querying the current RMS level.*
- std::vector< float > **peaklevel** () const  
*Read-only function for querying the current peak level.*
- void **process** ( **mha\_wave\_t** \**s*)  
*Function to store a chunk of audio in the level meter.*

## Private Attributes

- unsigned int **pos**
- unsigned int **filled**

## Additional Inherited Members

### 5.344.1 Detailed Description

Class for asynchronous level metering.

### 5.344.2 Constructor & Destructor Documentation

#### 5.344.2.1 **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.

### 5.344.3 Member Function Documentation

**5.344.3.1 rmslevel()** `std::vector< float > MHASignal::async_rmslevel_t::rmslevel ( ) const`

Read-only function for querying the current RMS level.

#### Returns

Vector of floats, one value for each channel, containing the RMS level in dB (SPL if calibrated properly).

**5.344.3.2 peaklevel()** `std::vector< float > MHASignal::async_rmslevel_t::peaklevel ( ) const`

Read-only function for querying the current peak level.

#### Returns

Vector of floats, one value for each channel, containing the peak level in dB (SPL if calibrated properly).

**5.344.3.3 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).
----------------	---

### 5.344.4 Member Data Documentation

**5.344.4.1 pos** unsigned int MHASignal::async\_rmslevel\_t::pos [private]

**5.344.4.2 filled** unsigned int MHASignal::async\_rmslevel\_t::filled [private]

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

- [mha\\_signal.hh](#)
- [mha\\_signal.cpp](#)

## 5.345 MHASignal::delay\_spec\_t Class Reference

### Public Member Functions

- [delay\\_spec\\_t](#) (unsigned int **delay**, unsigned int frames, unsigned int **channels**)
- [~delay\\_spec\\_t \(\)](#)
- [mha\\_spec\\_t \\* process \( mha\\_spec\\_t \\*\)](#)

### Private Attributes

- unsigned int **delay**
- [MHASignal::spectrum\\_t \\*\\* buffer](#)
- unsigned int **pos**

### 5.345.1 Constructor & Destructor Documentation

**5.345.1.1 delay\_spec\_t()** MHASignal::delay\_spec\_t::delay\_spec\_t (

```
    unsigned int delay,
    unsigned int frames,
    unsigned int channels )
```

**5.345.1.2 ~delay\_spec\_t()** MHASignal::delay\_spec\_t::~delay\_spec\_t ( )

### 5.345.2 Member Function Documentation

**5.345.2.1 process()** `mha_spec_t * MHASignal::delay_spec_t::process ( mha_spec_t * s )`

### 5.345.3 Member Data Documentation

**5.345.3.1 delay** `unsigned int MHASignal::delay_spec_t::delay [private]`

**5.345.3.2 buffer** `MHASignal::spectrum_t** MHASignal::delay_spec_t::buffer [private]`

**5.345.3.3 pos** `unsigned int MHASignal::delay_spec_t::pos [private]`

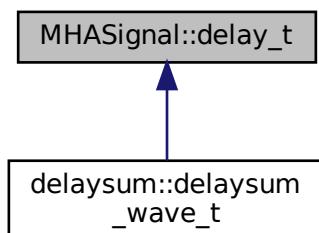
The documentation for this class was generated from the following files:

- `mha_signal.hh`
- `mha_signal.cpp`

## 5.346 MHASignal::delay\_t Class Reference

Class to realize a simple delay of waveform streams.

Inheritance diagram for MHASignal::delay\_t:



## Public Member Functions

- **delay\_t** (`std::vector< int > delays, unsigned int channels`)  
*Constructor.*
- **mha\_wave\_t \* process** (`mha_wave_t *s`)  
*Processing method.*
- **~delay\_t ()**
- `std::string inspect () const`

## Private Attributes

- `unsigned int channels`
- `unsigned int * delays`
- `unsigned int * pos`
- `mha_real_t ** buffer`

### 5.346.1 Detailed Description

Class to realize a simple delay of waveform streams.

### 5.346.2 Constructor & Destructor Documentation

**5.346.2.1 `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.

**5.346.2.2 `~delay_t()`** `MHASignal::delay_t::~delay_t ( )`

### 5.346.3 Member Function Documentation

**5.346.3.1 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.

**5.346.3.2 inspect()** `std::string MHASignal::delay_t::inspect ( ) const [inline]`

### 5.346.4 Member Data Documentation

**5.346.4.1 channels** `unsigned int MHASignal::delay_t::channels [private]`

**5.346.4.2 delays** `unsigned int* MHASignal::delay_t::delays [private]`

**5.346.4.3 pos** `unsigned int* MHASignal::delay_t::pos [private]`

### **5.346.4.4 buffer `mha_real_t** MHASignal::delay_t::buffer` [private]**

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

- `mha_signal.hh`
- `mha_signal.cpp`

## **5.347 MHASignal::delay\_wave\_t Class Reference**

Delayline containing wave fragments.

### **Public Member Functions**

- `delay_wave_t` (unsigned int `delay`, unsigned int `frames`, unsigned int `channels`)
- `~delay_wave_t ()`
- `mha_wave_t * process ( mha_wave_t *)`

### **Private Attributes**

- unsigned int `delay`
- `MHASignal::waveform_t ** buffer`
- unsigned int `pos`

### **5.347.1 Detailed Description**

Delayline containing wave fragments.

The delayline contains waveform fragments. The delay can be configured in integer fragments (sample delay or sub-sample delay is not possible).

### **5.347.2 Constructor & Destructor Documentation**

#### **5.347.2.1 `delay_wave_t()` `MHASignal::delay_wave_t::delay_wave_t (`**

```
unsigned int delay,
unsigned int frames,
unsigned int channels )
```

**5.347.2.2 ~delay\_wave\_t()** MHASignal::delay\_wave\_t::~delay\_wave\_t ( )

### 5.347.3 Member Function Documentation

**5.347.3.1 process()** mha\_wave\_t \* MHASignal::delay\_wave\_t::process ( mha\_wave\_t \* s )

### 5.347.4 Member Data Documentation

**5.347.4.1 delay** unsigned int MHASignal::delay\_wave\_t::delay [private]

**5.347.4.2 buffer** MHASignal::waveform\_t\*\* MHASignal::delay\_wave\_t::buffer [private]

**5.347.4.3 pos** unsigned int MHASignal::delay\_wave\_t::pos [private]

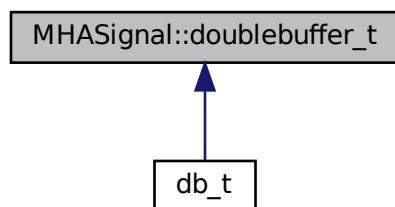
The documentation for this class was generated from the following files:

- **mha\_signal.hh**
- **mha\_signal.cpp**

## 5.348 MHASignal::doublebuffer\_t Class Reference

Double-buffering class.

Inheritance diagram for MHASignal::doublebuffer\_t:



## Public Member Functions

- **doublebuffer\_t** (unsigned int nchannels\_in, unsigned int nchannels\_out, unsigned int outer\_fragsize, unsigned int inner\_fragsize)  
*Constructor of double buffer.*
- virtual ~**doublebuffer\_t** ()
- **mha\_wave\_t \* outer\_process ( mha\_wave\_t \*s)**  
*Method to pass audio fragments into the inner layer.*

## Protected Member Functions

- virtual **mha\_wave\_t \* inner\_process ( mha\_wave\_t \*s)=0**  
*Method to realize inner processing callback.*

## Private Member Functions

- unsigned int **min** (unsigned int a, unsigned int b)

## Private Attributes

- **waveform\_t outer\_out**
- **mha\_wave\_t this\_outer\_out**
- **waveform\_t inner\_in**
- **waveform\_t inner\_out**
- unsigned int **k\_inner**
- unsigned int **k\_outer**
- unsigned int **ch**

### 5.348.1 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.

### 5.348.2 Constructor & Destructor Documentation

```
5.348.2.1 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)

**5.348.2.2 ~doublebuffer\_t()** `MHASignal::doublebuffer_t::~doublebuffer_t ( ) [virtual]`

**5.348.3 Member Function Documentation**

**5.348.3.1 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.

**5.348.3.2 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**

<b>s</b>	Pointer to input waveform fragment.
----------	-------------------------------------

**Returns**

Pointer to output waveform fragment.

Implemented in **db\_t** (p. 372).

**5.348.3.3 min()** `unsigned int MHASignal::doublebuffer_t::min (`  
`unsigned int a,`  
`unsigned int b ) [inline], [private]`

**5.348.4 Member Data Documentation**

**5.348.4.1 outer\_out waveform\_t** `MHASignal::doublebuffer_t::outer_out [private]`

**5.348.4.2 this\_outer\_out mha\_wave\_t** `MHASignal::doublebuffer_t::this_outer_out [private]`

**5.348.4.3 inner\_in waveform\_t** `MHASignal::doublebuffer_t::inner_in [private]`

**5.348.4.4 inner\_out waveform\_t** `MHASignal::doublebuffer_t::inner_out [private]`

**5.348.4.5 k\_inner** unsigned int MHASignal::doublebuffer\_t::k\_inner [private]

**5.348.4.6 k\_outer** unsigned int MHASignal::doublebuffer\_t::k\_outer [private]

**5.348.4.7 ch** unsigned int MHASignal::doublebuffer\_t::ch [private]

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

- [mha\\_signal.hh](#)
- [mha\\_signal.cpp](#)

## 5.349 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**

## 5.349.1 Constructor & Destructor Documentation

**5.349.1.1 `fft_t()`** MHASignal::fft\_t::fft\_t ( const unsigned int & n )

**5.349.1.2 `~fft_t()`** MHASignal::fft\_t::~fft\_t ( )

## 5.349.2 Member Function Documentation

**5.349.2.1 `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.

```
5.349.2.2 spec2wave() [1/2] void MHASignal::fft_t::spec2wave (
    const mha_spec_t * spec,
    mha_wave_t * wave )
```

```
5.349.2.3 spec2wave() [2/2] void MHASignal::fft_t::spec2wave (
    const mha_spec_t * spec,
    mha_wave_t * wave,
    unsigned int offset )
```

wave may have fewer number of frames than needed for a complete iFFT.

Only as many frames are written into wave as fit, starting with offset offset of the complete iFFT.

```
5.349.2.4 forward() void MHASignal::fft_t::forward (
    mha_spec_t * sIn,
    mha_spec_t * sOut )
```

```
5.349.2.5 backward() void MHASignal::fft_t::backward (
    mha_spec_t * sIn,
    mha_spec_t * sOut )
```

```
5.349.2.6 wave2spec_scale() void MHASignal::fft_t::wave2spec_scale (
    const mha_wave_t * wave,
    mha_spec_t * spec,
    bool swap )
```

```
5.349.2.7 spec2wave_scale() void MHASignal::fft_t::spec2wave_scale (
    const mha_spec_t * spec,
    mha_wave_t * wave )
```

---

**5.349.2.8 forward\_scale()** void MHASignal::fft\_t::forward\_scale (   
     mha\_spec\_t \* sIn,  
     mha\_spec\_t \* sOut )

**5.349.2.9 backward\_scale()** void MHASignal::fft\_t::backward\_scale (   
     mha\_spec\_t \* sIn,  
     mha\_spec\_t \* sOut )

**5.349.2.10 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 –].

**5.349.2.11 sort\_spec2fftw()** void MHASignal::fft\_t::sort\_spec2fftw (   
     fftw\_real \* s\_fftw,  
     const mha\_spec\_t \* s\_spec,  
     unsigned int ch ) [private]

Arrange the order of an internal spectrum to the fftw order.

### 5.349.3 Member Data Documentation

**5.349.3.1 nfft** unsigned int MHASignal::fft\_t::nfft [private]

**5.349.3.2 n\_re** unsigned int MHASignal::fft\_t::n\_re [private]

**5.349.3.3 n\_im** unsigned int MHASignal::fft\_t::n\_im [private]

**5.349.3.4 scale** mha\_real\_t MHASignal::fft\_t::scale [private]

**5.349.3.5 buf\_in** mha\_real\_t\* MHASignal::fft\_t::buf\_in [private]

**5.349.3.6 buf\_out** mha\_real\_t\* MHASignal::fft\_t::buf\_out [private]

**5.349.3.7 fftw\_plan\_wave2spec** rffftw\_plan MHASignal::fft\_t::fftw\_plan\_wave2spec [private]

**5.349.3.8 fftw\_plan\_spec2wave** rffftw\_plan MHASignal::fft\_t::fftw\_plan\_spec2wave [private]

**5.349.3.9 fftw\_plan\_fft** fftw\_plan MHASignal::fft\_t::fftw\_plan\_fft [private]

**5.349.3.10 fftw\_plan\_ifft** fftw\_plan MHASignal::fft\_t::fftw\_plan\_ifft [private]

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

- **mha\_signal\_fft.h**
- **mha\_signal.cpp**

## 5.350 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**

#### 5.350.1 Constructor & Destructor Documentation

**5.350.1.1 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
------------	------------

**5.350.1.2 ~hilbert\_fftw\_t()** MHASignal::hilbert\_fftw\_t::~hilbert\_fftw\_t ( )

D'tor of **hilbert\_fftw\_t** (p. 1212).

## 5.350.2 Member Function Documentation

**5.350.2.1 `hilbert()`** void MHASignal::hilbert\_fftw\_t::hilbert ( const mha\_wave\_t \* s\_in, mha\_wave\_t \* s\_out )

## 5.350.3 Member Data Documentation

**5.350.3.1 `n`** unsigned int MHASignal::hilbert\_fftw\_t::n [private]

**5.350.3.2 `p1`** rfftw\_plan MHASignal::hilbert\_fftw\_t::p1 [private]

**5.350.3.3 `p2`** fftw\_plan MHASignal::hilbert\_fftw\_t::p2 [private]

**5.350.3.4 `buf_r_in`** fftw\_real\* MHASignal::hilbert\_fftw\_t::buf\_r\_in [private]

**5.350.3.5 `buf_r_out`** fftw\_real\* MHASignal::hilbert\_fftw\_t::buf\_r\_out [private]

**5.350.3.6 `buf_c_in`** fftw\_complex\* MHASignal::hilbert\_fftw\_t::buf\_c\_in [private]

**5.350.3.7 buf\_c\_out** fftw\_complex\* MHASignal::hilbert\_fftw\_t::buf\_c\_out [private]

**5.350.3.8 sc** mha\_real\_t MHASignal::hilbert\_fftw\_t::sc [private]

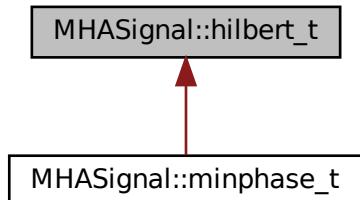
The documentation for this class was generated from the following file:

- [mha\\_signal.cpp](#)

## 5.351 MHASignal::hilbert\_t Class Reference

Hilbert transformation of a waveform segment.

Inheritance diagram for MHASignal::hilbert\_t:



### Public Member Functions

- **hilbert\_t** (unsigned int len)
- **~hilbert\_t ()**
- **void operator()** (const mha\_wave\_t \*, mha\_wave\_t \*)

*Apply Hilbert transformation on a waveform segment.*

### Private Attributes

- void \* **h**

### 5.351.1 Detailed Description

Hilbert transformation of a waveform segment.

Returns the imaginary part of the inverse Fourier transformation of the Fourier transformed input signal with negative frequencies set to zero.

### 5.351.2 Constructor & Destructor Documentation

**5.351.2.1 hilbert\_t()** MHASignal::hilbert\_t::hilbert\_t ( unsigned int len )

Parameters

<i>len</i>	Length of waveform segment
------------	----------------------------

**5.351.2.2 ~hilbert\_t()** MHASignal::hilbert\_t::~hilbert\_t ( )

### 5.351.3 Member Function Documentation

**5.351.3.1 operator()** void MHASignal::hilbert\_t::operator() ( const mha\_wave\_t \* s\_in, mha\_wave\_t \* s\_out )

Apply Hilbert transformation on a waveform segment.

### 5.351.4 Member Data Documentation

### 5.351.4.1 **h** void\* MHASignal::hilbert\_t::h [private]

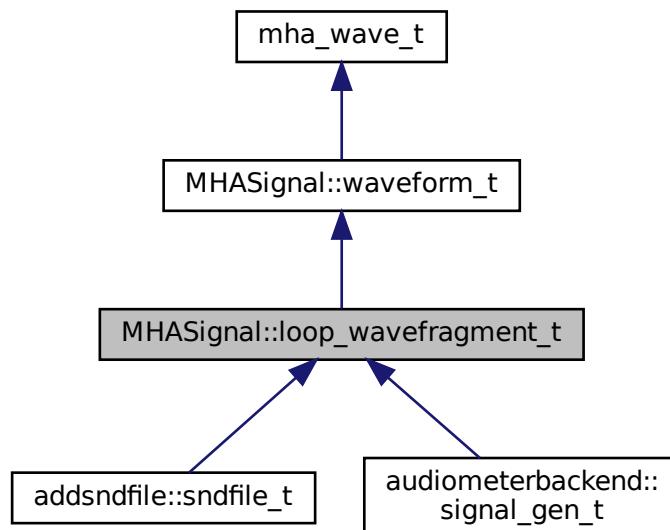
The documentation for this class was generated from the following files:

- **mha\_signal.hh**
- **mha\_signal.cpp**

## 5.352 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

### 5.352.1 Detailed Description

Copy a fixed waveform fragment to a series of waveform fragments of other size.

This class is designed to continuously play back a waveform to an output stream, with variable output block size.

### 5.352.2 Member Enumeration Documentation

#### 5.352.2.1 **level\_mode\_t** enum **MHASignal::loop\_wavefragment\_t::level\_mode\_t**

Switch for playback level mode.

**Enumerator**

relative	The nominal level is applied as a gain to the source signal.
peak	The nominal level is the peak level of source signal in Pascal.
rms	The nominal level is the RMS level of the source signal in Pascal.
rms_limit40	

**5.352.2.2 playback\_mode\_t enum MHASignal::loop\_wavefragment\_t::playback\_mode\_t**

Switch for playback mode.

**Enumerator**

add	Add source signal to output stream.
replace	Replace output stream by source signal.
input	Do nothing, keep output stream (source position is unchanged).
mute	Mute output stream (source position is unchanged).

**5.352.3 Constructor & Destructor Documentation**

```
5.352.3.1 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 <b>MHASignal::loop_wavefragment_t::level_mode_t</b> (p. 1217) for details)
<i>channels</i>	Mapping of input to output channels.
<i>startpos</i>	Starting position

## 5.352.4 Member Function Documentation

**5.352.4.1 get\_mapping()** `std::vector< int > MHASignal::loop_wavefragment_t::get_mapping ( unsigned int channels )`

**5.352.4.2 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

**5.352.4.3 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.

**5.352.4.4 `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).

**5.352.4.5 `set_level_lin()`** `void MHASignal::loop_wavefragment_t::set_level_lin ( mha_real_t l )`

**5.352.4.6 `set_level_db()`** `void MHASignal::loop_wavefragment_t::set_level_db ( mha_real_t l )`

**5.352.4.7 `rewind()`** `void MHASignal::loop_wavefragment_t::rewind ( )` [inline]

**5.352.4.8 `locate_end()`** `void MHASignal::loop_wavefragment_t::locate_end ( )` [inline]

**5.352.4.9 `is_playback_active()`** `bool MHASignal::loop_wavefragment_t::is_playback_active ( ) const` [inline]

## 5.352.5 Member Data Documentation

**5.352.5.1 playback\_channels** std::vector<int> MHASignal::loop\_wavefragment\_t::playback\_channels [private]

**5.352.5.2 b\_loop** bool MHASignal::loop\_wavefragment\_t::b\_loop [private]

**5.352.5.3 pos** unsigned int MHASignal::loop\_wavefragment\_t::pos [private]

**5.352.5.4 intern\_level** MHASignal::waveform\_t MHASignal::loop\_wavefragment\_t::intern\_level [private]

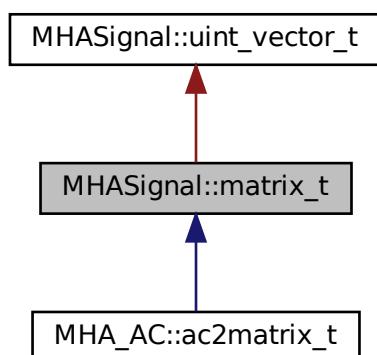
The documentation for this class was generated from the following files:

- [mha\\_signal.hh](#)
- [mha\\_signal.cpp](#)

## 5.353 MHASignal::matrix\_t Class Reference

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

Inheritance diagram for MHASignal::matrix\_t:



## Public Member Functions

- **matrix\_t** (unsigned int nRows, unsigned int nCols, bool b\_is\_complex=true)  
*Create a two-dimensional matrix.*
- **matrix\_t** (const **mha\_spec\_t** &spec)  
*Create a two-dimensional matrix from a spectrum, copy values.*
- **matrix\_t** (const **MHASignal::uint\_vector\_t** & size, bool b\_is\_complex=true)  
*Create n-dimensional matrix, described by size argument.*
- **matrix\_t** (const **MHASignal::matrix\_t** &)
- **matrix\_t** (const uint8\_t \*buf, unsigned int len)  
*Construct from memory area.*
- **~matrix\_t ()**
- **MHASignal::matrix\_t** & **operator=** (const **MHASignal::matrix\_t** &)  
**MHASignal::matrix\_t** & **operator=** (const **comm\_var\_t** &v)  
*Fill matrix with data of an AC variable object.*
- **comm\_var\_t get\_comm\_var ()**  
*Return a AC communication variable pointing to the data of the current matrix.*
- unsigned int **dimension () const**  
*Return the dimension of the matrix.*
- unsigned int **size** (unsigned int k) const  
*Return the size of the matrix.*
- unsigned int **get\_nelements () const**  
*Return total number of elements.*
- bool **is\_same\_size** (const **MHASignal::matrix\_t** &)  
*Test if matrix has same size as other.*
- bool **iscomplex () const**  
*Return information about complexity.*
- **mha\_real\_t** & **real** (const **MHASignal::uint\_vector\_t** &index)  
*Access real part of an element in a n-dimensional matrix.*
- **mha\_real\_t** & **imag** (const **MHASignal::uint\_vector\_t** &index)  
*Access imaginary part of an element in a n-dimensional matrix.*
- **mha\_complex\_t** & **operator()** (const **MHASignal::uint\_vector\_t** &index)  
*Access complex value of an element in a n-dimensional matrix.*
- const **mha\_real\_t** & **real** (const **MHASignal::uint\_vector\_t** &index) const  
*Access real part of an element in a n-dimensional matrix.*
- const **mha\_real\_t** & **imag** (const **MHASignal::uint\_vector\_t** &index) const  
*Access imaginary part of an element in a n-dimensional matrix.*
- const **mha\_complex\_t** & **operator()** (const **MHASignal::uint\_vector\_t** &index) const  
*Access complex value of an element in a n-dimensional matrix.*
- **mha\_real\_t** & **real** (unsigned int row, unsigned int col)  
*Access real part of an element in a two-dimensional matrix.*
- **mha\_real\_t** & **imag** (unsigned int row, unsigned int col)  
*Access imaginary part of an element in a two-dimensional matrix.*
- **mha\_complex\_t** & **operator()** (unsigned int row, unsigned int col)  
*Access complex value of an element in a two-dimensional matrix.*

- const **mha\_real\_t & real** (unsigned int row, unsigned int col) const  
*Access real part of an element in a two-dimensional matrix.*
- const **mha\_real\_t & imag** (unsigned int row, unsigned int col) const  
*Access imaginary part of an element in a two-dimensional matrix.*
- const **mha\_complex\_t & operator()** (unsigned int row, unsigned int col) const  
*Access complex value of an element in a two-dimensional matrix.*
- unsigned int **get\_nreals** () const
- unsigned int **get\_index** (unsigned int row, unsigned int col) const
- unsigned int **get\_index** (const **MHASignal::uint\_vector\_t &index**) const
- unsigned int **numbytes** () const  
*Return number of bytes needed to store into memory.*
- unsigned int **write** (uint8\_t \*buf, unsigned int len) const  
*Copy to memory area.*
- const **mha\_real\_t \* get\_rdata** () const  
*Return pointer of real data.*
- const **mha\_complex\_t \* get\_cdata** () const  
*Return pointer of complex data.*

## Private Attributes

- uint32\_t **complex\_ofs**
- uint32\_t **nelements**
- union {
  - mha\_real\_t \* rdata**
  - mha\_complex\_t \* cdata**
- };

## Additional Inherited Members

### 5.353.1 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.

### 5.353.2 Constructor & Destructor Documentation

**5.353.2.1 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

**5.353.2.2 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
-------------	---------------------------

**5.353.2.3 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

**5.353.2.4 matrix\_t() [4/5]** `MHASignal::matrix_t::matrix_t (`  
`const MHASignal::matrix_t & src )`

**5.353.2.5 matrix\_t() [5/5]** `MHASignal::matrix_t::matrix_t (`  
`const uint8_t * buf,`  
`unsigned int len )`

Construct from memory area.

## Warning

This constructor is not real time safe

### **5.353.2.6 ~matrix\_t()** `MHASignal::matrix_t::~matrix_t ( )`

## 5.353.3 Member Function Documentation

### **5.353.3.1 operator=() [1/2]** `matrix_t & MHASignal::matrix_t::operator= ( const MHASignal::matrix_t & src )`

### **5.353.3.2 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 ( <a href="#">comm_var_t (p. 359)</a> )
----------------	--

#### Note

The type and dimension of the AC variable must match the type and dimension of the matrix.

### **5.353.3.3 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 \(p. 359\)](#)), valid for the life time of the matrix.

**5.353.3.4 dimension()** `unsigned int MHASignal::matrix_t::dimension ( ) const [inline]`

Return the dimension of the matrix.

**Returns**

Dimension of the matrix

**5.353.3.5 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`

**5.353.3.6 get\_nelements()** `unsigned int MHASignal::matrix_t::get_nelements ( ) const`

Return total number of elements.

**5.353.3.7 is\_same\_size()** `bool MHASignal::matrix_t::is_same_size ( const MHASignal::matrix_t & src )`

Test if matrix has same size as other.

**5.353.3.8 iscomplex()** `bool MHASignal::matrix_t::iscomplex ( ) const [inline]`

Return information about complexity.

**5.353.3.9 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
--------------	--------------

**5.353.3.10 `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
--------------	--------------

**5.353.3.11 `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
--------------	--------------

**5.353.3.12 `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
--------------	--------------

**5.353.3.13 `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
--------------	--------------

```
5.353.3.14 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
--------------	--------------

```
5.353.3.15 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

```
5.353.3.16 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

**5.353.3.17 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

**5.353.3.18 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

**5.353.3.19 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

```
5.353.3.20 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

```
5.353.3.21 get_nreals() unsigned int MHASignal::matrix_t::get_nreals () const [inline]
```

```
5.353.3.22 get_index() [1/2] unsigned int MHASignal::matrix_t::get_index (
    unsigned int row,
    unsigned int col ) const
```

```
5.353.3.23 get_index() [2/2] unsigned int MHASignal::matrix_t::get_index (
    const MHASignal::uint_vector_t & index ) const
```

```
5.353.3.24 nbytes() unsigned int MHASignal::matrix_t::nbytes () const
```

Return number of bytes needed to store into memory.

```
5.353.3.25 write() unsigned int MHASignal::matrix_t::write (
    uint8_t * buf,
    unsigned int len ) const
```

Copy to memory area.

**5.353.3.26 `get_rdata()`** const `mha_real_t*` MHASignal::matrix\_t::get\_rdata ( ) const [inline]

Return pointer of real data.

**5.353.3.27 `get_cdata()`** const `mha_complex_t*` MHASignal::matrix\_t::get\_cdata ( ) const [inline]

Return pointer of complex data.

## 5.353.4 Member Data Documentation

**5.353.4.1 `complex_ofs`** uint32\_t MHASignal::matrix\_t::complex\_ofs [private]

**5.353.4.2 `nelements`** uint32\_t MHASignal::matrix\_t::nelements [private]

**5.353.4.3 `rdata`** `mha_real_t*` MHASignal::matrix\_t::rdata

**5.353.4.4 `cdata`** `mha_complex_t*` MHASignal::matrix\_t::cdata

**5.353.4.5 "@1** union { ... } [private]

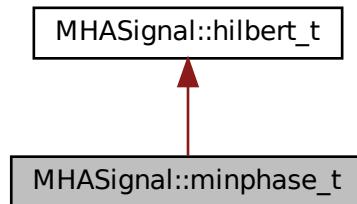
The documentation for this class was generated from the following files:

- `mha_signal.hh`
- `mha_signal.cpp`

## 5.354 MHASignal::minphase\_t Class Reference

Minimal phase function.

Inheritance diagram for MHASignal::minphase\_t:



### Public Member Functions

- **minphase\_t** (unsigned int fftlen, unsigned int ch)  
*Constructor.*
- void **operator()** ( **mha\_spec\_t** \*s)  
*Transform input spectrum to a minimal-phase spectrum, discarding the original phase.*

### Private Attributes

- **MHASignal::waveform\_t phase**

### Additional Inherited Members

#### 5.354.1 Detailed Description

Minimal phase function.

The output spectrum  $Y(f)$  is

$$Y(f) = |X(f)|e^{i\mathcal{H}\{\log|X(f)|\}},$$

with the input spectrum  $X(f)$  and the Hilbert transformation  $\mathcal{H}\{\dots\}$ .

#### 5.354.2 Constructor & Destructor Documentation

##### 5.354.2.1 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

### 5.354.3 Member Function Documentation

**5.354.3.1 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.
----------	-------------------------

### 5.354.4 Member Data Documentation

**5.354.4.1 phase** `MHASignal::waveform_t MHASignal::minphase_t::phase [private]`

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

- `mha_signal.hh`
- `mha_signal.cpp`

## 5.355 MHASignal::quantizer\_t Class Reference

Simple simulation of fixpoint quantization.

### Public Member Functions

- **quantizer\_t** (`unsigned int num_bits`)
   
*Constructor.*
- **void operator()** (`mha_wave_t &s`)
   
*Quantization of a waveform fragment.*

## Private Attributes

- bool `limit`
- `mha_real_t upscale`
- `mha_real_t downscale`
- `mha_real_t up_limit`

### 5.355.1 Detailed Description

Simple simulation of fixpoint quantization.

### 5.355.2 Constructor & Destructor Documentation

#### 5.355.2.1 `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.
-----------------------	--

### 5.355.3 Member Function Documentation

#### 5.355.3.1 `operator()()` `void MHASignal::quantizer_t::operator() (` `mha_wave_t & s )`

Quantization of a waveform fragment.

##### Parameters

<code>s</code>	Waveform fragment to be quantized.
----------------	------------------------------------

## 5.355.4 Member Data Documentation

**5.355.4.1 limit** `bool MHASignal::quantizer_t::limit` [private]

**5.355.4.2 upscale** `mha_real_t MHASignal::quantizer_t::upscale` [private]

**5.355.4.3 downscale** `mha_real_t MHASignal::quantizer_t::downscale` [private]

**5.355.4.4 up\_limit** `mha_real_t MHASignal::quantizer_t::up_limit` [private]

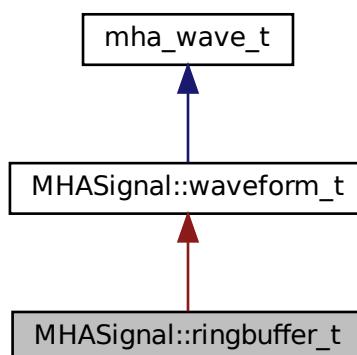
The documentation for this class was generated from the following files:

- `mha_signal.hh`
- `mha_signal.cpp`

## 5.356 MHASignal::ringbuffer\_t Class Reference

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

Inheritance diagram for MHASignal::ringbuffer\_t:



## Public Member Functions

- **ringbuffer\_t** (unsigned frames, unsigned **channels**, unsigned prefilled\_frames)  
*Creates new ringbuffer for time domain signal.*
- unsigned **contained\_frames** () const  
*number of currently contained frames*
- **mha\_real\_t & value** (unsigned frame, unsigned channel)  
*Access to value stored in ringbuffer.*
- void **discard** (unsigned frames)  
*Discards the oldest frames.*
- void **write** ( **mha\_wave\_t** &signal)  
*Copies the contents of the signal into the ringbuffer if there is enough space.*

## Private Attributes

- unsigned **next\_read\_frame\_index**  
*Index of oldest frame in underlying storage for the ringbuffer.*
- unsigned **next\_write\_frame\_index**  
*Index of first free frame in underlying storage.*

## Additional Inherited Members

### 5.356.1 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.

### 5.356.2 Constructor & Destructor Documentation

#### 5.356.2.1 **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. 760) if *prefilled\_frames > frames*

## 5.356.3 Member Function Documentation

**5.356.3.1 contained\_frames()** `unsigned MHASignal::ringbuffer_t::contained_frames () const [inline]`

number of currently contained frames

**5.356.3.2 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**

<b>MHA_Error</b> (p. 760)	if channel or frame out of bounds.
---------------------------	------------------------------------

**5.356.3.3 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**

<b>MHA_Error</b> (p. 760)	if frames > <b>contained_frames</b> (p. 1238)
---------------------------	---

**5.356.3.4 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**

<b>MHA_Error</b> (p. 760)	if there is not enough space or if the channel count mismatches. Nothing is copied if the space is insufficient.
---------------------------	---

## 5.356.4 Member Data Documentation

**5.356.4.1 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.

**5.356.4.2 next\_write\_frame\_index** `unsigned MHASignal::ringbuffer_t::next_write_<→frame_index [private]`

Index of first free frame in underlying storage.

Next frame to be stored will be placed here.

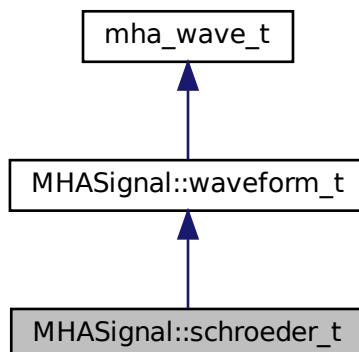
The documentation for this class was generated from the following files:

- [mha\\_signal.hh](#)
- [mha\\_signal.cpp](#)

## 5.357 MHASignal::schroeder\_t Class Reference

Schroeder tone complex class.

Inheritance diagram for MHASignal::schroeder\_t:



## Public Types

- enum **sign\_t** { **up**, **down** }  
*Enumerator for sign of Schroeder tone complex sweep direction.*
- typedef float(\*) **groupdelay\_t** (float f, float fmin, float fmax)  
*Function type for group delay definition.*

## Public Member Functions

- schroeder\_t** (unsigned int len, unsigned int **channels**=1, **schroeder\_t::sign\_t** sign=**up**, **mha\_real\_t** speed=1)  
*Constructor.*
- schroeder\_t** (unsigned int len, unsigned int **channels**=1, **schroeder\_t::groupdelay\_t** freqfun= **MHASignal::schroeder\_t::identity**, float fmin=0, float fmax=1, float eps=1e-10)  
*Construct create Schroeder tone complex from a given frequency function.*

## Static Public Member Functions

- static float **identity** (float x, float, float)
- static float **log\_up** (float x, float fmin, float fmax)
- static float **log\_down** (float x, float fmin, float fmax)

## Additional Inherited Members

### 5.357.1 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,  $\sigma$  is the sign of the sweep (-1 for up sweep, +1 for down sweep),  $\tau$  is the sweep duration in samples,  $f_s$  is the sampling rate in Hz and  $\alpha$  is the relative sweep speed.

### 5.357.2 Member Typedef Documentation

#### 5.357.2.1 **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.

## 5.357.3 Member Enumeration Documentation

### 5.357.3.1 `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 ( $\sigma = +1$ )

## 5.357.4 Constructor & Destructor Documentation

### 5.357.4.1 `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 $\tau$ of the Schroeder tone complex in samples
<i>channels</i>	Number of channels
<i>sign</i>	Sign $\sigma$ of Schroeder sweep
<i>speed</i>	Relative speed $\alpha$ (curvature of phase function)

**5.357.4.2 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 $\tau$ of the Schroeder tone complex in samples.
<i>channels</i>	Number of channels.
<i>freqfun</i>	Frequency function $g(f)$ .
<i>fmin</i>	Start frequency (relative to Nyquist frequency).
<i>fmax</i>	End frequency (relative to Nyquist frequency).
<i>eps</i>	Stability constant for frequency ranges not covered by Schroeder tone complex.

#### 5.357.5 Member Function Documentation

**5.357.5.1 identity()** `static float MHASignal::schroeder_t::identity (`

```
    float x,
    float ,
    float ) [inline], [static]
```

**5.357.5.2 `log_up()`** static float MHASignal::schroeder\_t::log\_up ( float *x*, float *fmin*, float *fmax* ) [inline], [static]

**5.357.5.3 `log_down()`** static float MHASignal::schroeder\_t::log\_down ( float *x*, float *fmin*, float *fmax* ) [inline], [static]

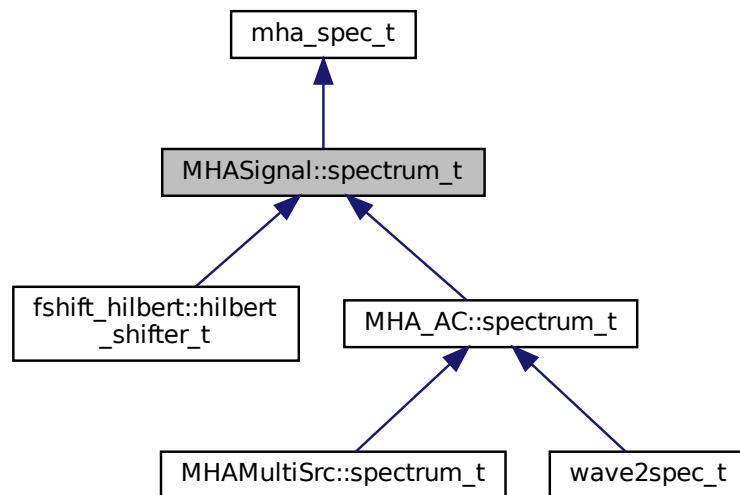
The documentation for this class was generated from the following files:

- `mha_signal.hh`
- `mha_signal.cpp`

## 5.358 MHASignal::spectrum\_t Class Reference

a signal processing class for spectral data (based on `mha_spec_t` (p. 790))

Inheritance diagram for MHASignal::spectrum\_t:



## Public Member Functions

- **spectrum\_t** (const unsigned int &frames, const unsigned int & **channels**)  
*constructor of spectrum class*
- **spectrum\_t** (const **mha\_spec\_t** &)  
*Copy constructor.*
- **spectrum\_t** (const **MHASignal::spectrum\_t** &)  
*Copy constructor.*
- **spectrum\_t** (const std::vector< **mha\_complex\_t** > &)
- virtual ~**spectrum\_t** (void)
- **mha\_complex\_t** & **operator()** (unsigned int f, unsigned int ch)  
*Access to element.*
- **mha\_complex\_t** & **operator[]** (unsigned int k)  
*Access to a single element, direct index into data buffer.*
- **mha\_complex\_t** & **value** (unsigned int f, unsigned int ch)  
*Access to element.*
- void **copy** (const **mha\_spec\_t** &)  
*copy all elements from a spectrum*
- void **copy\_channel** (const **mha\_spec\_t** &s, unsigned sch, unsigned dch)  
*Copy one channel of a given spectrum signal to a target channel.*
- void **export\_to** ( **mha\_spec\_t** &)  
*copy elements to spectrum structure*
- void **scale** (const unsigned int &, const unsigned int &, const unsigned int &, const **mha\_real\_t** &)  
*scale section [a,b) in channel "ch" by "val"*
- void **scale\_channel** (const unsigned int &, const **mha\_real\_t** &)  
*scale all elements in one channel*

## Additional Inherited Members

### 5.358.1 Detailed Description

a signal processing class for spectral data (based on **mha\_spec\_t** (p. 790))

### 5.358.2 Constructor & Destructor Documentation

#### 5.358.2.1 **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

**5.358.2.2 `spectrum_t()` [2/4]** `spectrum_t::spectrum_t ( const mha_spec_t & src )` [explicit]

Copy constructor.

**5.358.2.3 `spectrum_t()` [3/4]** `spectrum_t::spectrum_t ( const MHASignal::spectrum_t & src )`

Copy constructor.

**5.358.2.4 `spectrum_t()` [4/4]** `spectrum_t::spectrum_t ( const std::vector< mha_complex_t > & src )`

**5.358.2.5 `~spectrum_t()`** `spectrum_t::~spectrum_t ( void )` [virtual]

Reimplemented in **MHA\_AC::spectrum\_t** (p. 732).

### 5.358.3 Member Function Documentation

**5.358.3.1 `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

**5.358.3.2 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

**5.358.3.3 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

---

**5.358.3.4 `copy()`** `void spectrum_t::copy (`  
`const mha_spec_t & src )`

copy all elements from a spectrum

Parameters

<i>src</i>	input spectrum
------------	----------------

**5.358.3.5 `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

**5.358.3.6 `export_to()`** `void spectrum_t::export_to (`  
`mha_spec_t & dest )`

copy elements to spectrum structure

Parameters

<i>dest</i>	destination spectrum structure
-------------	--------------------------------

**5.358.3.7 `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

```
5.358.3.8 scale_channel() void spectrum_t::scale_channel (
    const unsigned int & ch,
    const mha_real_t & src )
```

scale all elements in one channel

### Parameters

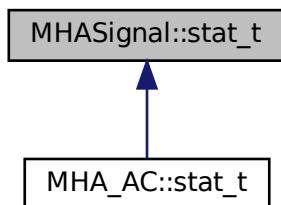
<i>ch</i>	channel number
<i>src</i>	scale factor

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

- **mha\_signal.hh**
- **mha\_signal.cpp**

## 5.359 MHASignal::stat\_t Class Reference

Inheritance diagram for MHASignal::stat\_t:



## Public Member Functions

- **stat\_t** (const unsigned int &frames, const unsigned int & **channels**)
- void **mean** ( **mha\_wave\_t** &m)
- void **mean\_std** ( **mha\_wave\_t** &m, **mha\_wave\_t** &s)
- void **push** (const **mha\_wave\_t** &)
- void **push** (const **mha\_real\_t** &x, const unsigned int &k, const unsigned int &ch)

## Private Attributes

- **MHASignal::waveform\_t n**
- **MHASignal::waveform\_t sum**
- **MHASignal::waveform\_t sum2**

### 5.359.1 Constructor & Destructor Documentation

**5.359.1.1 stat\_t()** `MHASignal::stat_t::stat_t (`  
    `const unsigned int & frames,`  
    `const unsigned int & channels )`

### 5.359.2 Member Function Documentation

**5.359.2.1 mean()** `void MHASignal::stat_t::mean (`  
    `mha_wave_t & m )`

**5.359.2.2 mean\_std()** `void MHASignal::stat_t::mean_std (`  
    `mha_wave_t & m,`  
    `mha_wave_t & s )`

**5.359.2.3 `push()` [1/2]** `void MHASignal::stat_t::push (`  
`const mha_wave_t & x )`

**5.359.2.4 `push()` [2/2]** `void MHASignal::stat_t::push (`  
`const mha_real_t & x,`  
`const unsigned int & k,`  
`const unsigned int & ch )`

### 5.359.3 Member Data Documentation

**5.359.3.1 `n`** `MHASignal::waveform_t MHASignal::stat_t::n` [private]

**5.359.3.2 `sum`** `MHASignal::waveform_t MHASignal::stat_t::sum` [private]

**5.359.3.3 `sum2`** `MHASignal::waveform_t MHASignal::stat_t::sum2` [private]

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

- `mha_signal.hh`
- `mha_signal.cpp`

## 5.360 `MHASignal::subsample_delay_t` Class Reference

implements subsample delay in spectral domain.

### Public Member Functions

- **`subsample_delay_t`** (`const std::vector< float > &subsample_delay, unsigned fftlen)`  
*Constructor computes complex phase factors to apply to achieve subsample delay.*
- **`void process ( mha_spec_t *s )`**  
*Apply the phase\_gains to s to achieve the subsample delay.*
- **`void process ( mha_spec_t *s, unsigned idx )`**  
*Apply the pase gains to channel idx in s to achieve the subsample delay in channel idx.*

## Public Attributes

- **spectrum\_t phase\_gains**

*The complex factors to apply to achieve the necessary phase shift.*

## Private Attributes

- unsigned **last\_complex\_bin**

*index of the last complex fft bin for the used fft length.*

### 5.360.1 Detailed Description

implements subsample delay in spectral domain.

When transformed back to the time domain, the signal is delayed by the configured fraction of a sample. This operation must not be used in a smoothgains bracket.

### 5.360.2 Constructor & Destructor Documentation

**5.360.2.1 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. 760) if the parameters are out of range

### 5.360.3 Member Function Documentation

**5.360.3.1 process()** [1/2] void MHASignal::subsample\_delay\_t::process (   
`mha_spec_t * s )`

Apply the phase\_gains to s to achieve the subsample delay.

**5.360.3.2 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

<b>MHA_Error</b> (p. 760)	if idx >= s->num_channels
---------------------------	---------------------------

### 5.360.4 Member Data Documentation

**5.360.4.1 phase\_gains** `spectrum_t` MHASignal::subsample\_delay\_t::phase\_gains

The complex factors to apply to achieve the necessary phase shift.

**5.360.4.2 last\_complex\_bin** `unsigned` MHASignal::subsample\_delay\_t::last\_complex\_bin  
[private]

index of the last complex fft bin for the used fft length.

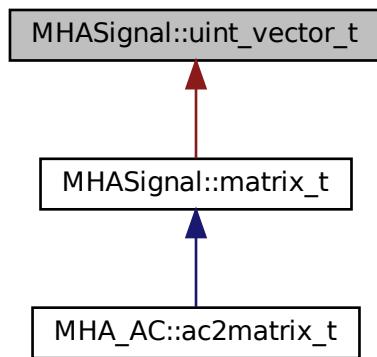
The documentation for this class was generated from the following files:

- **mha\_signal.hh**
- **mha\_signal.cpp**

## 5.361 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`

### 5.361.1 Detailed Description

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

### 5.361.2 Constructor & Destructor Documentation

**5.361.2.1 `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.
------------------	-------------------

**5.361.2.2 `uint_vector_t()` [2/3]** `MHASignal::uint_vector_t::uint_vector_t (`  
`const uint_vector_t & src )`

**5.361.2.3 `uint_vector_t()` [3/3]** `MHASignal::uint_vector_t::uint_vector_t (`  
`const uint8_t * buf,`  
`unsigned int len )`

Construct from memory area.

#### Warning

This constructor is not real time safe

**5.361.2.4 ~uint\_vector\_t()** MHASignal::uint\_vector\_t::~uint\_vector\_t ( )**5.361.3 Member Function Documentation****5.361.3.1 operator==( )** bool MHASignal::uint\_vector\_t::operator== ( const uint\_vector\_t & src ) const

Check for equality.

**5.361.3.2 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.

**5.361.3.3 get\_length()** unsigned int MHASignal::uint\_vector\_t::get\_length ( ) const [inline]

Return the length of the vector.

**5.361.3.4 operator[]( ) [1/2]** const uint32\_t& MHASignal::uint\_vector\_t::operator[] ( unsigned int k ) const [inline]

Read-only access to elements.

**5.361.3.5 operator[]( ) [2/2]** `uint32_t& MHASignal::uint_vector_t::operator[ ] ( unsigned int k ) [inline]`

Access to elements.

**5.361.3.6 nbytes()** `unsigned int MHASignal::uint_vector_t::nbytes ( ) const`

Return number of bytes needed to store into memory.

**5.361.3.7 write()** `unsigned int MHASignal::uint_vector_t::write ( uint8_t * buf, unsigned int len ) const`

Copy to memory area.

**5.361.3.8 getdata()** `const uint32_t* MHASignal::uint_vector_t::getdata ( ) const [inline]`

Return pointer to the data field.

## 5.361.4 Member Data Documentation

**5.361.4.1 length** `uint32_t MHASignal::uint_vector_t::length [protected]`

**5.361.4.2 data** `uint32_t* MHASignal::uint_vector_t::data [protected]`

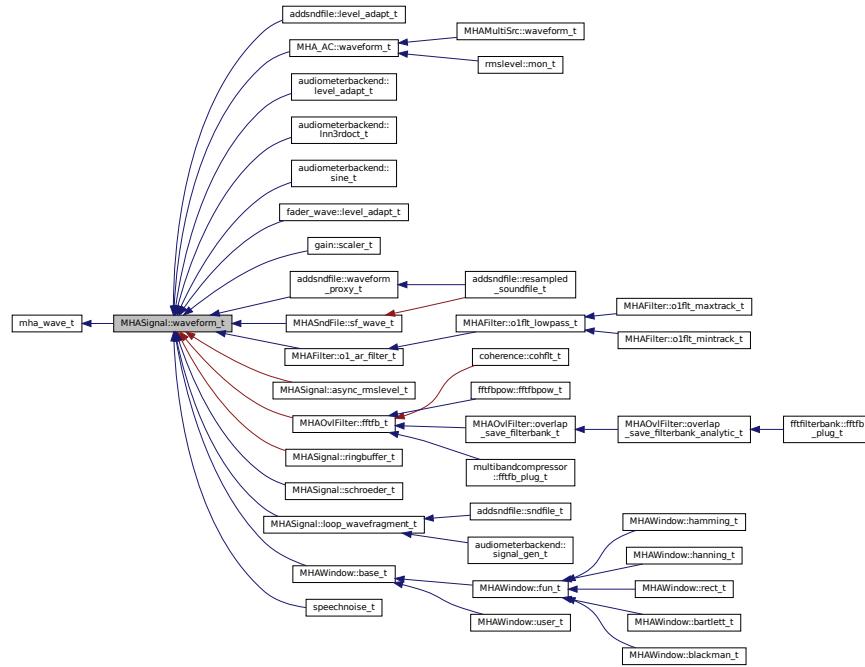
The documentation for this class was generated from the following files:

- **mha\_signal.hh**
- **mha\_signal.cpp**

## 5.362 MHASignal::waveform\_t Class Reference

signal processing class for waveform data (based on `mha_wave_t` (p. 836))

## 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. 836) 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. 836) structure*
  - void **limit** (const **mha\_real\_t** & min, const **mha\_real\_t** & max)
 

*limit target to range [min,max]*
  - void **power** (const **waveform\_t** &)
 

*transform waveform signal (in Pa) to squared signal (in W/m<sup>2</sup>)*
  - void **powspec** (const **mha\_spec\_t** &)
 

*get the power spectrum (in W/m<sup>2</sup>) from a complex spectrum*
  - void **scale** (const unsigned int &a, const unsigned int &b, const unsigned int &ch, const **mha\_real\_t** &val)
 

*scale section [a,b] in channel "ch" by "val"*
  - void **scale** (const unsigned int &k, const unsigned int &ch, const **mha\_real\_t** &val)

- `scale one element`
- void **scale\_channel** (const unsigned int &, const **mha\_real\_t** &)  
*scale one channel of target with a scalar*
- void **scale\_frame** (const unsigned int &, const **mha\_real\_t** &)
- unsigned int **get\_size** () const

## Additional Inherited Members

### 5.362.1 Detailed Description

signal processing class for waveform data (based on **mha\_wave\_t** (p. 836))

### 5.362.2 Constructor & Destructor Documentation

**5.362.2.1 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

**5.362.2.2 waveform\_t() [2/5]** waveform\_t::waveform\_t (

const <b>mhaconfig_t</b> & <i>cf</i> ) [explicit]
---

Constructor to create a waveform from plugin configuration.

#### Parameters

<i>cf</i>	Plugin configuration
-----------	----------------------

**5.362.2.3 `waveform_t()` [3/5]** `waveform_t::waveform_t (`  
    `const mha_wave_t & src )` [explicit]

Copy contructor for `mha_wave_t` (p. [836](#)) source.

**5.362.2.4 `waveform_t()` [4/5]** `waveform_t::waveform_t (`  
    `const MHASignal::waveform_t & src )`

Copy contructor.

**5.362.2.5 `waveform_t()` [5/5]** `waveform_t::waveform_t (`  
    `const std::vector< mha_real_t > & src )`

Copy contructor for `std::vector<mha_real_t>` source.

A waveform structure with a single channel is created, the length is equal to the number of elements in the source vector.

**5.362.2.6 `~waveform_t()`** `waveform_t::~waveform_t (`  
    `void )` [virtual]

Reimplemented in `MHA_AC::waveform_t` (p. [736](#)).

### 5.362.3 Member Function Documentation

**5.362.3.1 `flatten()`** `std::vector< mha_real_t > waveform_t::flatten ( ) const`

**5.362.3.2 operator std::vector< mha\_real\_t >()** MHASignal::waveform\_t::operator std::vector< mha\_real\_t > ( ) const [explicit]

**5.362.3.3 operator=()** void MHASignal::waveform\_t::operator= ( const mha\_real\_t & v ) [inline]

**5.362.3.4 operator[](1/2)** mha\_real\_t& MHASignal::waveform\_t::operator[ ] ( unsigned int k ) [inline]

**5.362.3.5 operator[](2/2)** const mha\_real\_t& MHASignal::waveform\_t::operator[ ] ( unsigned int k ) const [inline]

**5.362.3.6 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

**5.362.3.7 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

**5.362.3.8 `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

**5.362.3.9 `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

**5.362.3.10 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

**5.362.3.11 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

**5.362.3.12 sum() [3/3]** `mha_real_t waveform_t::sum ( )`

sum of all elements

**Returns**

sum of all elements

**5.362.3.13 sumsqr()** `mha_real_t waveform_t::sumsqr ( )`

sum of square of all elements

**Returns**

sum of square of all elements

**5.362.3.14 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

**5.362.3.15 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

**5.362.3.16 assign() [2/2]** void waveform\_t::assign ( const mha\_real\_t & val )

set all elements to value

Parameters

val	new value
-----	-----------

**5.362.3.17 assign\_frame()** void waveform\_t::assign\_frame ( const unsigned int & k, const mha\_real\_t & val )

assign value "val" to frame k in all channels

Parameters

k	frame number
val	new value

**5.362.3.18 assign\_channel()** void waveform\_t::assign\_channel ( const unsigned int & ch, const mha\_real\_t & val )

assign value "val" to channel ch in all frames

Parameters

ch	channel number
val	new value

**5.362.3.19 copy() [1/3]** void waveform\_t::copy ( const std::vector< mha\_real\_t > & v )

**5.362.3.20 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)
------------	---

**5.362.3.21 copy()** [3/3] `void waveform_t::copy (`  
`const mha_wave_t * src )`

**5.362.3.22 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

**5.362.3.23 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

**5.362.3.24 `export_to()`** `void waveform_t::export_to (`  
 `mha_wave_t & dest )`

copy data into allocated **mha\_wave\_t** (p. 836) structure

**Parameters**

<i>dest</i>	destination structure
-------------	-----------------------

**5.362.3.25 `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

**5.362.3.26 `power()`** `void waveform_t::power (`  
 `const waveform_t & src )`

transform waveform signal (in Pa) to squared signal (in W/m<sup>2</sup>)

**Parameters**

<i>src</i>	linear waveform signal (in Pa)
------------	--------------------------------

**5.362.3.27 powspec()** void waveform\_t::powspec ( const mha\_spec\_t & src )

get the power spectrum (in W/m<sup>2</sup>) from a complex spectrum

#### Parameters

<i>src</i>	complex spectrum (normalized to Pa)
------------	-------------------------------------

**5.362.3.28 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

**5.362.3.29 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

```
5.362.3.30 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

```
5.362.3.31 scale_frame() void waveform_t::scale_frame (
    const unsigned int & frame,
    const mha_real_t & val )
```

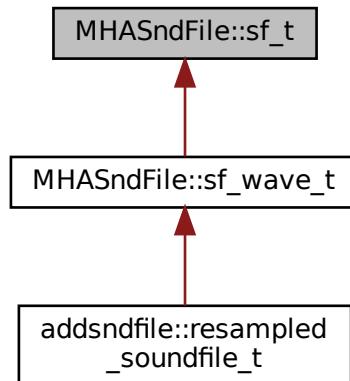
```
5.362.3.32 get_size() unsigned int MHASignal::waveform_t::get_size ( ) const [inline]
```

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

- **mha\_signal.hh**
- **mha\_signal.cpp**

### 5.363 MHASndFile::sf\_t Class Reference

Inheritance diagram for MHASndFile::sf\_t:



#### Public Member Functions

- **sf\_t** (const std::string &fname)
- **~sf\_t** ()

#### Public Attributes

- **SNDFILE \* sf**

#### 5.363.1 Constructor & Destructor Documentation

**5.363.1.1 sf\_t()** MHASndFile::sf\_t::sf\_t ( const std::string & fname )

**5.363.1.2 ~sf\_t()** MHASndFile::sf\_t::~sf\_t ( )

### 5.363.2 Member Data Documentation

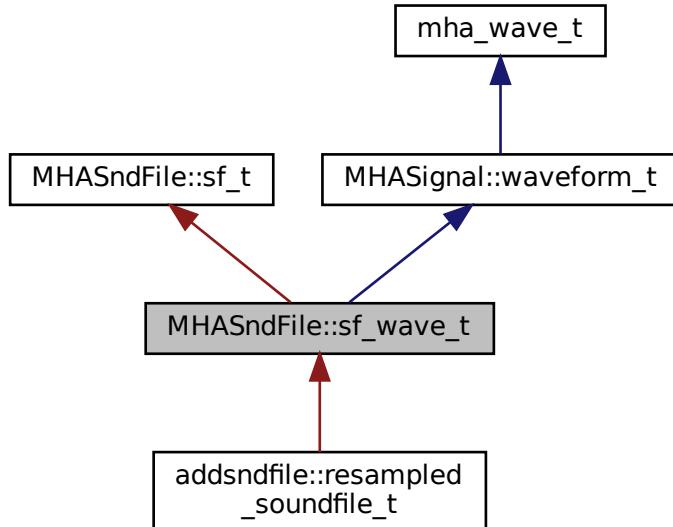
#### 5.363.2.1 sf SNDFILE\* MHASndFile::sf\_t::sf

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

- **mhasndfile.h**
- **mhasndfile.cpp**

### 5.364 MHASndFile::sf\_wave\_t Class Reference

Inheritance diagram for MHASndFile::sf\_wave\_t:



### Public Member Functions

- **sf\_wave\_t** (const std::string &fname, **mha\_real\_t** peaklevel\_db, unsigned int maxlen=std::numeric\_limits< unsigned int >:: **max()**, unsigned int startpos=0, std::vector< int > channel\_map=std::vector< int >())

## Additional Inherited Members

### 5.364.1 Constructor & Destructor Documentation

```
5.364.1.1 sf_wave_t() MHASndFile::sf_wave_t::sf_wave_t (
    const std::string & fname,
    mha_real_t peaklevel_db,
    unsigned int maxlen = std::numeric_limits<unsigned int>:: max(),
    unsigned int startpos = 0,
    std::vector< int > channel_map = std::vector<int>() )
```

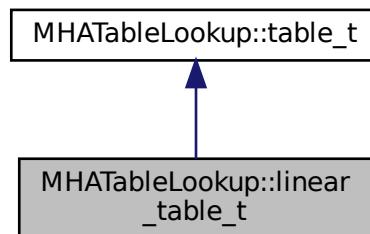
The documentation for this class was generated from the following files:

- **mhasndfile.h**
- **mhasndfile.cpp**

## 5.365 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

### 5.365.1 Detailed Description

Class for interpolation with equidistant x values.

This class can be used for linear interpolation tasks where the mesh points are known for equidistant x values.

Before the class can be used for interpolation, it has to be filled with the y values for the mesh points, the x range has to be specified, and when all values are given, the prepare method has to be called so that the object can determine the distance between x values from the range and the number of mesh points given.

Only after prepare has returned, the object may be used for interpolation.

## 5.365.2 Constructor & Destructor Documentation

**5.365.2.1 `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.

**5.365.2.2 `~linear_table_t()`** `linear_table_t::~linear_table_t (`  
    `void )`

destructor

## 5.365.3 Member Function Documentation

**5.365.3.1 `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  $\text{xmin}$  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).

**5.365.3.2 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](#)).

**5.365.3.3 set\_xmin()** `void linear_table_t::set_xmin ( mha_real_t xmin )`

set the x value for the first mesh point.

Must be called before prepare can be called.

**5.365.3.4 add\_entry()** `void linear_table_t::add_entry ( mha_real_t y )`

set the y value for the next mesh point.

Must be called at least twice before prepare can be called.

**5.365.3.5 set\_xmax()** `void linear_table_t::set_xmax ( mha_real_t xmax )`

this sets the x value for a past-the-end, not added mesh point.

Example:

```
t.set_xmin(100);
t.add_entry(0); // mesh point {100,0}
t.add_entry(1); // mesh point {110,1}
// the next mesh point would be at x=120, but we do not add this
t.set_xmax(120); // the x where the next mesh point would be
t.prepare();
```

now, t.interp(100) == 0; t.interp(110) == 1; t.interp(105) == 0.5;

---

**5.365.3.6 `prepare()`** `void linear_table_t::prepare (`  
`void )`

`prepare` computes the x distance of the mesh points based on the values given to `set_xmin`, `set_xmax`, and the number of times that `add_entry` was called.

#### Precondition

`set_xmin`, `set_xmax`, `add_entry` functions must have been called before calling `prepare`, `add_entry` must have been called at least twice.

Only after this method has been called, `interp` or `lookup` may be called.

**5.365.3.7 `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](#)).

### 5.365.4 Member Data Documentation

**5.365.4.1 `vy`** `mha_real_t* MHATableLookup::linear_table_t::vy [protected]`

**5.365.4.2 `len`** `unsigned int MHATableLookup::linear_table_t::len [protected]`

**5.365.4.3 `vec_y`** `vector< mha_real_t> MHATableLookup::linear_table_t::vec_y [private]`

**5.365.4.4 xmin** `mha_real_t` MHATableLookup::linear\_table\_t::xmin [private]

**5.365.4.5 xmax** `mha_real_t` MHATableLookup::linear\_table\_t::xmax [private]

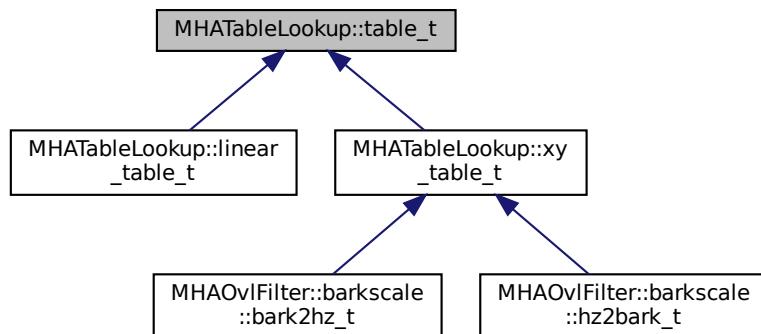
**5.365.4.6 scalefac** `mha_real_t` MHATableLookup::linear\_table\_t::scalefac [private]

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

- `mha_tablelookup.hh`
- `mha_tablelookup.cpp`

## 5.366 MHATableLookup::table\_t Class Reference

Inheritance diagram for MHATableLookup::table\_t:



### Public Member Functions

- `table_t (void)`
- `virtual ~table_t (void)`
- `virtual mha_real_t lookup ( mha_real_t) const =0`
- `virtual mha_real_t interp ( mha_real_t) const =0`

## Protected Member Functions

- virtual void **clear** (void)=0

### 5.366.1 Constructor & Destructor Documentation

**5.366.1.1 `table_t()`** `table_t::table_t (`  
    `void )`

**5.366.1.2 `~table_t()`** `table_t::~table_t (`  
    `void ) [virtual]`

### 5.366.2 Member Function Documentation

**5.366.2.1 `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).

**5.366.2.2 `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).

```
5.366.2.3 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).

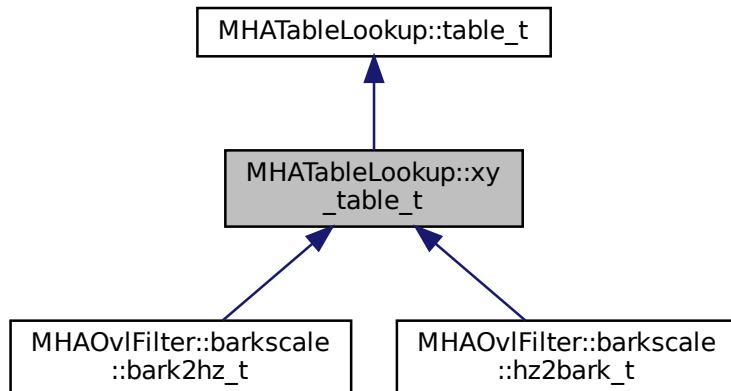
The documentation for this class was generated from the following files:

- **mha\_tablelookup.hh**
- **mha\_tablelookup.cpp**

## 5.367 MHATableLookup::xy\_table\_t Class Reference

Class for interpolation with non-equidistant x values.

Inheritance diagram for MHATableLookup::xy\_table\_t:



### Public Member Functions

- **xy\_table\_t ()**
- **mha\_real\_t lookup ( mha\_real\_t x) const**

*Return the y-value at the position of the nearest x value below input.*
- **mha\_real\_t interp ( mha\_real\_t x) const**

*Linear interpolation function.*
- **void add\_entry ( mha\_real\_t x, mha\_real\_t y)**

*Add a single x-y pair entry.*
- **void add\_entry ( mha\_real\_t \*pVX, mha\_real\_t \*pVY, unsigned int len)**

- Add multiple entries at once.
- void **clear ()**  
Clear the table and transformation functions.
- void **set\_xfun** (float(\*pXFun)(float))  
Set transformation function for x values.
- void **set\_yfun** (float(\*pYFun)(float))  
Set transformation function for y values during insertion.
- void **set\_xyfun** (float(\*pYFun)(float, float))  
Set transformation function for y values during insertion, based on x and y values.
- std::pair< **mha\_real\_t**, **mha\_real\_t** > **get\_xlimits () const**  
returns the min and max x of all mesh points that are stored in the lookup table, i.e.

## Private Attributes

- std::map< **mha\_real\_t**, **mha\_real\_t** > **mXY**
- float(\* **xfun** )(float)
- float(\* **yfun** )(float)
- float(\* **xyfun** )(float, float)

## Additional Inherited Members

### 5.367.1 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.

### 5.367.2 Constructor & Destructor Documentation

#### 5.367.2.1 **xy\_table\_t()** `xy_table_t::xy_table_t ( )`

### 5.367.3 Member Function Documentation

#### 5.367.3.1 **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).

**5.367.3.2 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).

**5.367.3.3 add\_entry() [1/2]** `void xy_table_t::add_entry (mha_real_t x, mha_real_t y)`

Add a single x-y pair entry.

**Parameters**

x	x value
y	corresponding y value

---

**5.367.3.4 add\_entry()** [2/2] void xy\_table\_t::add\_entry (

<i>mha_real_t</i> *	<i>pVX</i> ,
<i>mha_real_t</i> *	<i>pVY</i> ,
unsigned int	<i>uLength</i> )

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

**5.367.3.5 clear()** void xy\_table\_t::clear ( ) [virtual]

Clear the table and transformation functions.

Implements **MHATableLookup::table\_t** (p. 1280).

**5.367.3.6 set\_xfun()** void xy\_table\_t::set\_xfun (

float(*)(float)	<i>fun</i> )
-----------------	--------------

Set transformation function for x values.

#### Parameters

<i>fun</i>	Transformation function.
------------	--------------------------

**5.367.3.7 set\_yfun()** void xy\_table\_t::set\_yfun (

float(*)(float)	<i>fun</i> )
-----------------	--------------

Set transformation function for y values during insertion.

#### Parameters

<i>fun</i>	Transformation function.
------------	--------------------------

**5.367.3.8 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.
------------	--------------------------

**5.367.3.9 get\_xlimits()** std::pair< mha\_real\_t, mha\_real\_t> MHATableLookup::xy\_<-->  
table\_t::get\_xlimits ( ) const [inline]

returns the min and max x of all mesh points that are stored in the lookup table, i.e.

after transformation with xfun, if any. Not real-time safe

## 5.367.4 Member Data Documentation

**5.367.4.1 mXY** std::map< mha\_real\_t, mha\_real\_t> MHATableLookup::xy\_table\_t::mXY  
[private]

**5.367.4.2 xfun** float(\* MHATableLookup::xy\_table\_t::xfun) (float) [private]

**5.367.4.3 yfun** float(\* MHATableLookup::xy\_table\_t::yfun) (float) [private]

**5.367.4.4 xyfun** float(\* MHATableLookup::xy\_table\_t::xyfun) (float, float) [private]

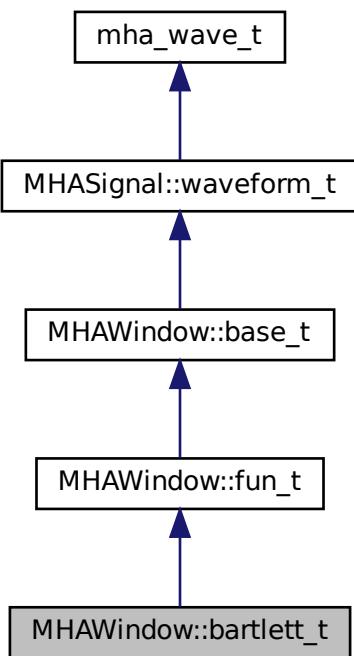
The documentation for this class was generated from the following files:

- **mha\_tablelookup.hh**
- **mha\_tablelookup.cpp**

## 5.368 MHAWindow::bartlett\_t Class Reference

Bartlett window.

Inheritance diagram for MHAWindow::bartlett\_t:



### Public Member Functions

- **bartlett\_t** (unsigned int n)

## Additional Inherited Members

### 5.368.1 Detailed Description

Bartlett window.

### 5.368.2 Constructor & Destructor Documentation

**5.368.2.1 bartlett\_t()** MHAWindow::bartlett\_t::bartlett\_t ( unsigned int n ) [inline]

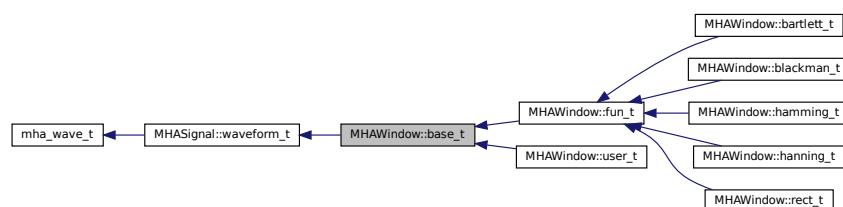
The documentation for this class was generated from the following file:

- **mha\_windowparser.h**

## 5.369 MHAWindow::base\_t Class Reference

Common base for window types.

Inheritance diagram for MHAWindow::base\_t:



## Public Member Functions

- **base\_t** (unsigned int len)  
*Constructor.*
- **base\_t** (const MHAWindow::base\_t &src)  
*Copy constructor.*
- void **operator()** ( mha\_wave\_t &) const  
*Apply window to waveform segment (reference)*
- void **operator()** ( mha\_wave\_t \* ) const  
*Apply window to waveform segment (pointer)*
- void **ramp\_begin** ( mha\_wave\_t &) const  
*Apply a ramp at the begining.*
- void **ramp\_end** ( mha\_wave\_t &) const  
*Apply a ramp at the end.*

## Additional Inherited Members

### 5.369.1 Detailed Description

Common base for window types.

### 5.369.2 Constructor & Destructor Documentation

**5.369.2.1 `base_t()` [1/2]** `MHAWindow::base_t::base_t (`  
`unsigned int len )`

Constructor.

#### Parameters

<code>len</code>	Window length in samples.
------------------	---------------------------

**5.369.2.2 `base_t()` [2/2]** `MHAWindow::base_t::base_t (`  
`const MHAWindow::base_t & src )`

Copy constructor.

#### Parameters

<code>src</code>	Source to be copied
------------------	---------------------

### 5.369.3 Member Function Documentation

**5.369.3.1 `operator()()` [1/2]** `void MHAWindow::base_t::operator() (`  
`mha_wave_t & s ) const`

Apply window to waveform segment (reference)

**5.369.3.2 operator()()** [2/2] void MHAWindow::base\_t::operator() (   
 mha\_wave\_t \* s ) const

Apply window to waveform segment (pointer)

**5.369.3.3 ramp\_begin()** void MHAWindow::base\_t::ramp\_begin (   
 mha\_wave\_t & s ) const

Apply a ramp at the begining.

**5.369.3.4 ramp\_end()** void MHAWindow::base\_t::ramp\_end (   
 mha\_wave\_t & s ) const

Apply a ramp at the end.

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

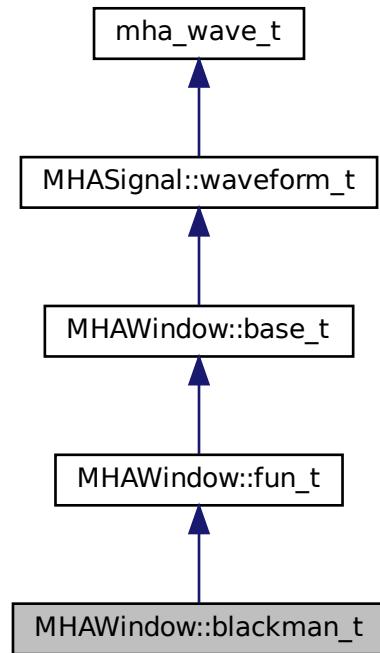
- **mha\_windowparser.h**
- **mha\_windowparser.cpp**

## 5.370 MHAWindow::blackman\_t Class Reference

---

Blackman window.

Inheritance diagram for MHAWindow::blackman\_t:



## Public Member Functions

- **blackman\_t** (unsigned int n)

## Additional Inherited Members

### 5.370.1 Detailed Description

Blackman window.

### 5.370.2 Constructor & Destructor Documentation

**5.370.2.1 blackman\_t()** MHAWindow::blackman\_t ( unsigned int n ) [inline]

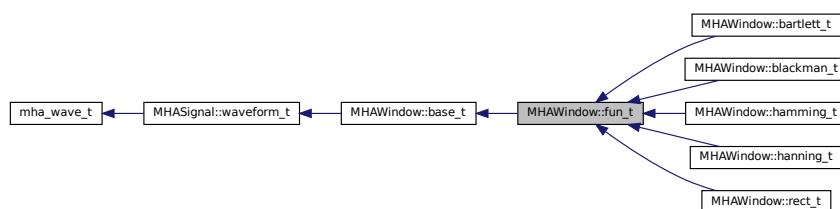
The documentation for this class was generated from the following file:

- **mha\_windowparser.h**

## 5.371 MHAWindow::fun\_t Class Reference

Generic window based on a generator function.

Inheritance diagram for MHAWindow::fun\_t:



### Public Member Functions

- **fun\_t** (unsigned int n, float(\*fun)(float), float xmin=-1, float xmax=1, bool min\_included=true, bool max\_included=false)
- Constructor.*

### Additional Inherited Members

#### 5.371.1 Detailed Description

Generic window based on a generator function.

The generator function should return a valid window function in the interval [-1,1[.

#### 5.371.2 Constructor & Destructor Documentation

**5.371.2.1 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. <b>MHAWindow::hanning()</b> (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

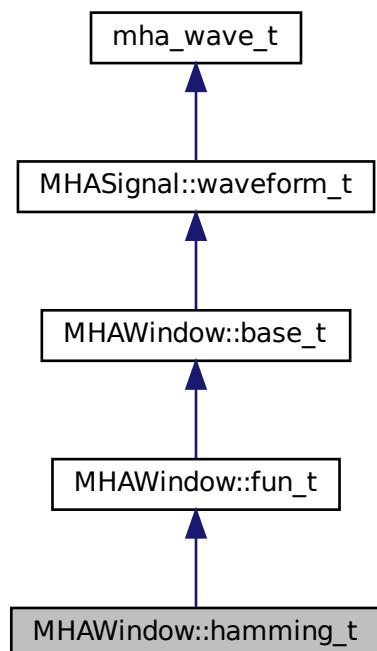
The documentation for this class was generated from the following files:

- **mha\_windowparser.h**
- **mha\_windowparser.cpp**

## 5.372 MHAWindow::hamming\_t Class Reference

Hamming window.

Inheritance diagram for MHAWindow::hamming\_t:



## Public Member Functions

- **hamming\_t** (unsigned int n)

## Additional Inherited Members

### 5.372.1 Detailed Description

Hamming window.

### 5.372.2 Constructor & Destructor Documentation

#### 5.372.2.1 **hamming\_t()** MHAWindow::hanning\_t::hamming\_t (

```
unsigned int n ) [inline]
```

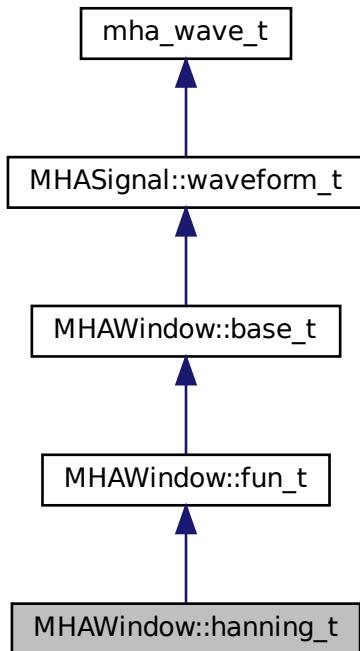
The documentation for this class was generated from the following file:

- **mha\_windowparser.h**

## 5.373 MHAWindow::hanning\_t Class Reference

von-Hann window

Inheritance diagram for MHAWindow::hanning\_t:



## Public Member Functions

- **hanning\_t** (unsigned int n)

## Additional Inherited Members

### 5.373.1 Detailed Description

von-Hann window

### 5.373.2 Constructor & Destructor Documentation

**5.373.2.1 hanning\_t()** `MHAWindow::hanning_t::hanning_t ( unsigned int n ) [inline]`

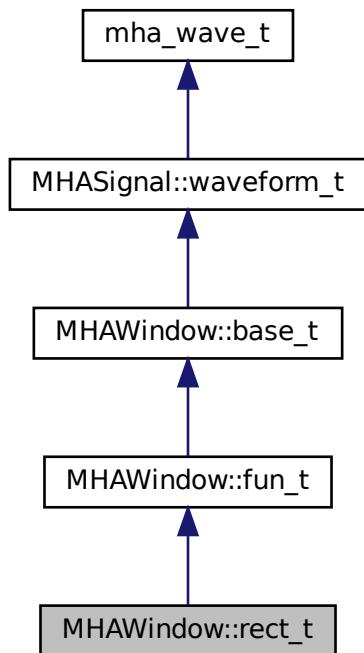
The documentation for this class was generated from the following file:

- `mha_windowparser.h`

## 5.374 MHAWindow::rect\_t Class Reference

Rectangular window.

Inheritance diagram for MHAWindow::rect\_t:



### Public Member Functions

- `rect_t (unsigned int n)`

### Additional Inherited Members

#### 5.374.1 Detailed Description

Rectangular window.

### 5.374.2 Constructor & Destructor Documentation

**5.374.2.1 rect\_t()** MHAWindow::rect\_t::rect\_t ( unsigned int n ) [inline]

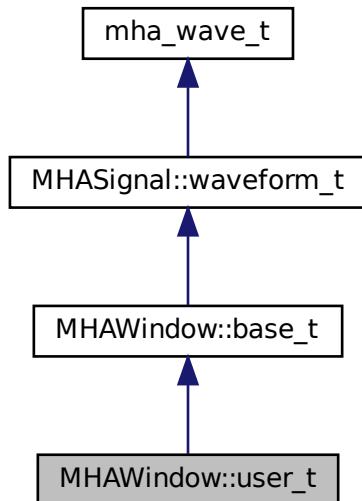
The documentation for this class was generated from the following file:

- **mha\_windowparser.h**

### 5.375 MHAWindow::user\_t Class Reference

User defined window.

Inheritance diagram for MHAWindow::user\_t:



#### Public Member Functions

- **user\_t** (const std::vector< **mha\_real\_t** > &wnd)  
*Constructor.*

## Additional Inherited Members

### 5.375.1 Detailed Description

User defined window.

### 5.375.2 Constructor & Destructor Documentation

**5.375.2.1 user\_t()** MHAWindow::user\_t::user\_t (const std::vector< mha\_real\_t > & wnd )

Constructor.

#### Parameters

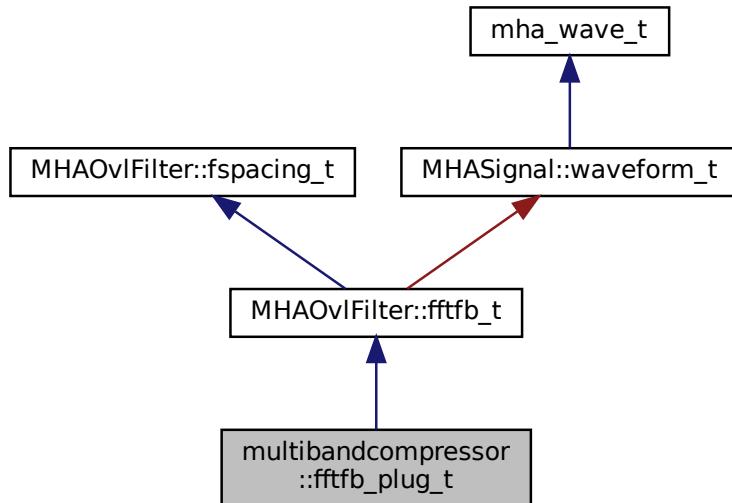
<i>wnd</i>	User defined window
------------	---------------------

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

- **mha\_windowparser.h**
- **mha\_windowparser.cpp**

## 5.376 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

#### 5.376.1 Constructor & Destructor Documentation

```
5.376.1.1 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 )
```

## 5.376.2 Member Function Documentation

5.376.2.1 **insert()** void multibandcompressor::fftfb\_plug\_t::insert ( )

## 5.376.3 Member Data Documentation

5.376.3.1 **cfv** MHA\_AC::waveform\_t multibandcompressor::fftfb\_plug\_t::cfv [private]

vector of nominal center frequencies / Hz

5.376.3.2 **efv** MHA\_AC::waveform\_t multibandcompressor::fftfb\_plug\_t::efv [private]

vector of edge frequencies / Hz

5.376.3.3 **bwv** MHA\_AC::waveform\_t multibandcompressor::fftfb\_plug\_t::bwv [private]

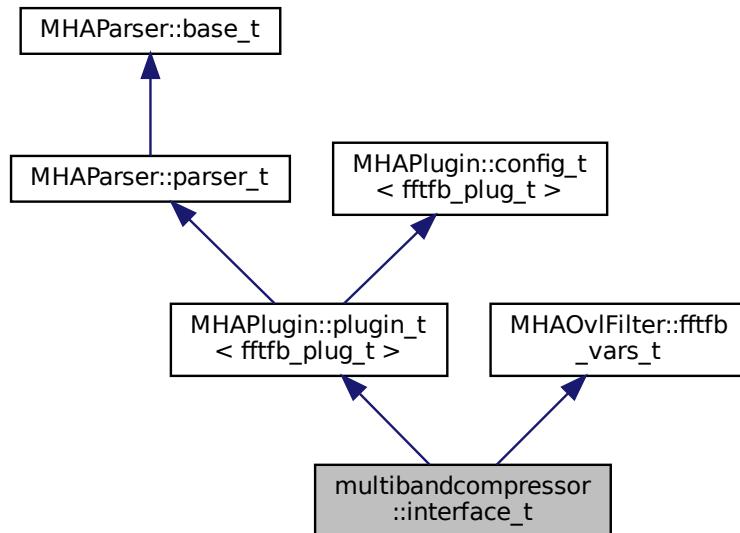
vector of band-weights (sum of squared fft-bin-weights)/num\_frames

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

- **multibandcompressor.cpp**

### 5.377 multibandcompressor::interface\_t Class Reference

Inheritance diagram for multibandcompressor::interface\_t:



#### Public Member Functions

- **interface\_t ( algo\_comm\_t iac, const std::string &configured\_name)**
- **void prepare ( mhaconfig\_t &)**
- **void release ()**
- **mha\_spec\_t \* process ( mha\_spec\_t \*)**

#### Private Member Functions

- **void update\_cfg ()**

#### Private Attributes

- **MHA\_AC::int\_t num\_channels**
- **DynComp::dc\_afterburn\_t burn**
- **MHAEvents::patchbay\_t< interface\_t > patchbay**
- **std::string algo**
- **MHParse::mhapluginloader\_t plug**
- **plugin\_signals\_t \* plug\_sigs**

## Additional Inherited Members

### 5.377.1 Constructor & Destructor Documentation

**5.377.1.1 `interface_t()`** `multibandcompressor::interface_t::interface_t ( algo_comm_t iac,  
const std::string & configured_name )`

Default values are set and MHA configuration variables registered into the parser.

#### Parameters

<code>ac</code>	algorithm communication handle
<code>—</code>	
<code>th</code>	chain name
<code>al</code>	algorithm name

### 5.377.2 Member Function Documentation

**5.377.2.1 `prepare()`** `void multibandcompressor::interface_t::prepare ( mhaconfig_t & tf ) [virtual]`

Implements `MHAPlugin::plugin_t< fftfb_plug_t >` (p. [1149](#)).

**5.377.2.2 `release()`** `void multibandcompressor::interface_t::release ( ) [virtual]`

Reimplemented from `MHAPlugin::plugin_t< fftfb_plug_t >` (p. [1150](#)).

**5.377.2.3 `process()`** `mha_spec_t * multibandcompressor::interface_t::process ( mha_spec_t * s )`

**5.377.2.4 update\_cfg()** void multibandcompressor::interface\_t::update\_cfg ( ) [private]

### 5.377.3 Member Data Documentation

**5.377.3.1 num\_channels** MHA\_AC::int\_t multibandcompressor::interface\_t::num\_channels [private]

**5.377.3.2 burn** DynComp::dc\_afterburn\_t multibandcompressor::interface\_t::burn [private]

**5.377.3.3 patchbay** MHAEVENTS::patchbay\_t< interface\_t> multibandcompressor::interface\_t::patchbay [private]

**5.377.3.4 algo** std::string multibandcompressor::interface\_t::algo [private]

**5.377.3.5 plug** MHAPARSER::mhapluginloader\_t multibandcompressor::interface\_t::plug [private]

**5.377.3.6 plug\_sigs** plugin\_signals\_t\* multibandcompressor::interface\_t::plug\_sigs [private]

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

- **multibandcompressor.cpp**

## **5.378 multibandcompressor::plugin\_signals\_t Class Reference**

### **Public Member Functions**

- **plugin\_signals\_t** (unsigned int **channels**, unsigned int bands)
- void **update\_levels** ( **MHAOvlFilter::fftfb\_t** \*, **mha\_spec\_t** \***s\_in**)
- void **apply\_gains** ( **MHAOvlFilter::fftfb\_t** \*, **DynComp::dc\_afterburn\_t** &**burn**, **mha\_spec\_t** \***s\_out**)

### **Public Attributes**

- **mha\_wave\_t** \* **plug\_output**

### **Private Attributes**

- **MHASignal::waveform\_t** **plug\_level**
- **MHASignal::waveform\_t** **gain**

#### **5.378.1 Constructor & Destructor Documentation**

```
5.378.1.1 plugin_signals_t() multibandcompressor::plugin_signals_t::plugin_signals_t (
```

unsigned int *channels*,  
unsigned int *bands* )

#### **5.378.2 Member Function Documentation**

```
5.378.2.1 update_levels() void multibandcompressor::plugin_signals_t::update_levels (
```

**MHAOvlFilter::fftfb\_t** \* *pFb*,  
**mha\_spec\_t** \* *s\_in* )

**5.378.2.2 apply\_gains()** void multibandcompressor::plugin\_signals\_t::apply\_gains ( MHAOvlFilter::fftfb\_t \* pFb,  
DynComp::dc\_afterburn\_t & burn,  
mha\_spec\_t \* s\_out )

### 5.378.3 Member Data Documentation

**5.378.3.1 plug\_level** MHASignal::waveform\_t multibandcompressor::plugin\_signals\_t::plug\_level [private]

**5.378.3.2 gain** MHASignal::waveform\_t multibandcompressor::plugin\_signals\_t::gain [private]

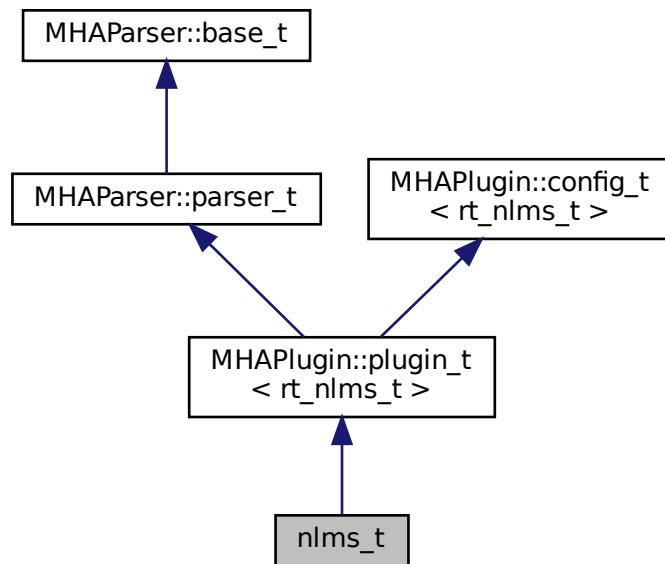
**5.378.3.3 plug\_output** mha\_wave\_t\* multibandcompressor::plugin\_signals\_t::plug\_output

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

- **multibandcompressor.cpp**

## 5.379 nlms\_t Class Reference

Inheritance diagram for nlms\_t:



### Public Member Functions

- `nlms_t ( algo_comm_t iac, const std::string &configured_name)`
- `void prepare ( mhaconfig_t &)`
- `void release ()`
- `mha_wave_t * process ( mha_wave_t *)`

### Private Member Functions

- `void update ()`

### Private Attributes

- `MHAParser::float_t rho`
- `MHAParser::float_t c`
- `MHAParser::int_t ntaps`
- `MHAParser::string_t name_u`
- `MHAParser::string_t name_d`

- **MHAParser::kw\_t normtype**
- **MHAParser::kw\_t estimtype**
- **MHAParser::float\_t lambda\_smoothing\_power**
- **MHAParser::string\_t name\_e**
- **MHAParser::string\_t name\_f**
- **MHAParser::int\_t n\_no\_update**
- std::string algo
- **MHAEvents::patchbay\_t< nlms\_t > patchbay**

## Additional Inherited Members

### 5.379.1 Constructor & Destructor Documentation

**5.379.1.1 nlms\_t()** nlms\_t::nlms\_t (

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

### 5.379.2 Member Function Documentation

**5.379.2.1 prepare()** void nlms\_t::prepare (

```
mhaconfig_t & cf ) [virtual]
```

Implements **MHAPlugin::plugin\_t< rt\_nlms\_t >** (p. 1149).

**5.379.2.2 release()** void nlms\_t::release ( ) [virtual]

Reimplemented from **MHAPlugin::plugin\_t< rt\_nlms\_t >** (p. 1150).

**5.379.2.3 process()** mha\_wave\_t \* nlms\_t::process (

```
mha_wave_t * s )
```

**5.379.2.4 update()** void nlms\_t::update ( ) [private]

### 5.379.3 Member Data Documentation

**5.379.3.1 rho** MHAParser::float\_t nlms\_t::rho [private]

**5.379.3.2 c** MHAParser::float\_t nlms\_t::c [private]

**5.379.3.3 ntaps** MHAParser::int\_t nlms\_t::ntaps [private]

**5.379.3.4 name\_u** MHAParser::string\_t nlms\_t::name\_u [private]

**5.379.3.5 name\_d** MHAParser::string\_t nlms\_t::name\_d [private]

**5.379.3.6 normtype** MHAParser::kw\_t nlms\_t::normtype [private]

**5.379.3.7 estimtype** MHAParser::kw\_t nlms\_t::estimtype [private]

**5.379.3.8 lambda\_smoothing\_power** MHAParser::float\_t nlms\_t::lambda\_smoothing←  
\_power [private]

**5.379.3.9 name\_e** `MHAParser::string_t nlms_t::name_e [private]`

**5.379.3.10 name\_f** `MHAParser::string_t nlms_t::name_f [private]`

**5.379.3.11 n\_no\_update** `MHAParser::int_t nlms_t::n_no_update [private]`

**5.379.3.12 algo** `std::string nlms_t::algo [private]`

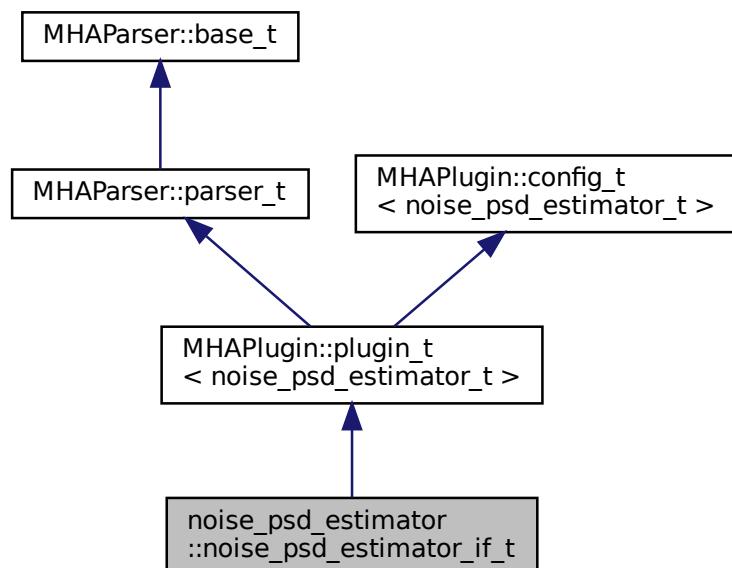
**5.379.3.13 patchbay** `MHAEVENTS::patchbay_t< nlms_t > nlms_t::patchbay [private]`

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

- `nlms_wave.cpp`

## 5.380 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 ( algo_comm_t iac, const std::string &configured_name)`
- `mha_spec_t * process ( mha_spec_t *)`
- `void prepare ( mhaconfig_t &)`

## Private Member Functions

- `void update_cfg ()`

## Private Attributes

- `MHAParser::float_t alphaPH1mean`
- `MHAParser::float_t alphaPSD`
- `MHAParser::float_t q`
- `MHAParser::float_t xiOptDb`
- `std::string name`
- `MHAEvents::patchbay_t< noise_psd_estimator_if_t > patchbay`

## Additional Inherited Members

### 5.380.1 Constructor & Destructor Documentation

```
5.380.1.1 noise_psd_estimator_if_t() noise_psd_estimator::noise_psd_estimator_if_t<→
t::noise_psd_estimator_if_t (
    algo_comm_t iac,
    const std::string & configured_name )
```

### 5.380.2 Member Function Documentation

```
5.380.2.1 process() mha_spec_t * noise_psd_estimator::noise_psd_estimator_if_t<→
::process (
    mha_spec_t * s )
```

**5.380.2.2 `prepare()`** `void noise_psd_estimator::noise_psd_estimator_if_t::prepare ( mhaconfig_t & cf ) [virtual]`

Implements `MHAPlugIn::plugin_t< noise_psd_estimator_t >` (p. [1149](#)).

**5.380.2.3 `update_cfg()`** `void noise_psd_estimator::noise_psd_estimator_if_t::update_cfg ( ) [private]`

### 5.380.3 Member Data Documentation

**5.380.3.1 `alphaPH1mean`** `MHAParser::float_t noise_psd_estimator::noise_psd_estimator_if_t::alphaPH1mean [private]`

**5.380.3.2 `alphaPSD`** `MHAParser::float_t noise_psd_estimator::noise_psd_estimator_if_t::alphaPSD [private]`

**5.380.3.3 `q`** `MHAParser::float_t noise_psd_estimator::noise_psd_estimator_if_t::q [private]`

**5.380.3.4 `xiOptDb`** `MHAParser::float_t noise_psd_estimator::noise_psd_estimator_if_t::xiOptDb [private]`

**5.380.3.5 `name`** `std::string noise_psd_estimator::noise_psd_estimator_if_t::name [private]`

**5.380.3.6 patchbay** `MHAEvents::patchbay_t< noise_psd_estimator_if_t> noise_psd_<→ estimator::noise_psd_estimator_if_t::patchbay [private]`

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

- `noise_psd_estimator.cpp`

## 5.381 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`

### 5.381.1 Constructor & Destructor Documentation

```
5.381.1.1 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 )
```

## 5.381.2 Member Function Documentation

```
5.381.2.1 process() void noise_psd_estimator::noise_psd_estimator_t::process (
    mha_spec_t * noisyDftFrame )
```

```
5.381.2.2 insert() void noise_psd_estimator::noise_psd_estimator_t::insert ( ) [inline]
```

## 5.381.3 Member Data Documentation

```
5.381.3.1 noisyPer MHASignal::waveform_t noise_psd_estimator::noise_psd_estimator<=
_t::noisyPer [private]
```

```
5.381.3.2 PH1mean MHASignal::waveform_t noise_psd_estimator::noise_psd_estimator<=
_t::PH1mean [private]
```

```
5.381.3.3 noisePow MHA_AC::waveform_t noise_psd_estimator::noise_psd_estimator<=
_t::noisePow [private]
```

**5.381.3.4 inputPow** `MHA_AC::waveform_t` `noise_psd_estimator::noise_psd_estimator_t::inputPow` [private]

**5.381.3.5 snrPost1Debug** `MHA_AC::waveform_t` `noise_psd_estimator::noise_psd_estimator_t::snrPost1Debug` [private]

**5.381.3.6 GLRDebug** `MHA_AC::waveform_t` `noise_psd_estimator::noise_psd_estimator_t::GLRDebug` [private]

**5.381.3.7 PH1Debug** `MHA_AC::waveform_t` `noise_psd_estimator::noise_psd_estimator_t::PH1Debug` [private]

**5.381.3.8 estimateDebug** `MHA_AC::waveform_t` `noise_psd_estimator::noise_psd_estimator_t::estimateDebug` [private]

**5.381.3.9 inputSpec** `MHA_AC::spectrum_t` `noise_psd_estimator::noise_psd_estimator_t::inputSpec` [private]

**5.381.3.10 alphaPH1mean\_** `float` `noise_psd_estimator::noise_psd_estimator_t::alphaPH1mean_` [private]

**5.381.3.11 alphaPSD\_** `float` `noise_psd_estimator::noise_psd_estimator_t::alphaPSD_` [private]

**5.381.3.12 priorFact** float noise\_psd\_estimator::noise\_psd\_estimator\_t::priorFact [private]

**5.381.3.13 xiOpt** float noise\_psd\_estimator::noise\_psd\_estimator\_t::xiOpt [private]

**5.381.3.14 logGLRFact** float noise\_psd\_estimator::noise\_psd\_estimator\_t::logGLRFact [private]

**5.381.3.15 GLRexp** float noise\_psd\_estimator::noise\_psd\_estimator\_t::GLRexp [private]

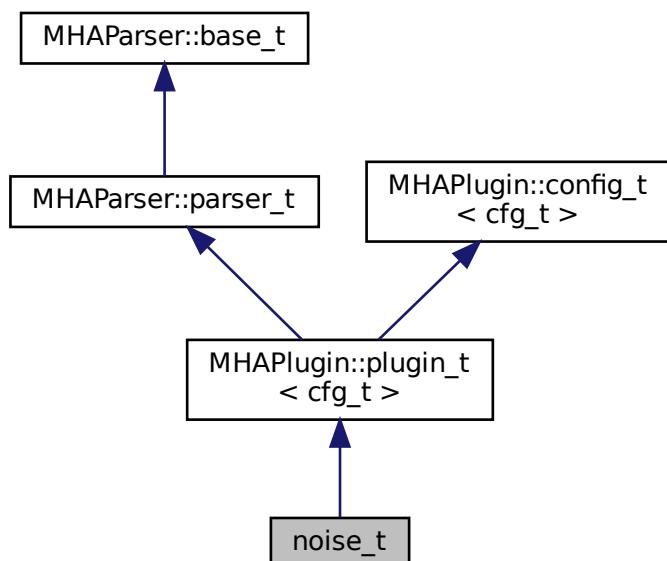
**5.381.3.16 frameno** int noise\_psd\_estimator::noise\_psd\_estimator\_t::frameno [private]

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

- `noise_psd_estimator.cpp`

## 5.382 noise\_t Class Reference

Inheritance diagram for noise\_t:



## Public Member Functions

- `noise_t ( algo_comm_t iac, const std::string &configured_name)`
- `mha_wave_t * process ( mha_wave_t *)`
- `mha_spec_t * process ( mha_spec_t *)`
- `void prepare ( mhaconfig_t &)`
- `void update_cfg ()`

## Private Attributes

- `MHAParser::float_t lev`
- `MHAParser::kw_t mode`
- `MHAParser::float_t frozennoise_length`
- `MHAParser::int_t seed`
- `MHAEvents::patchbay_t< noise_t > patchbay`

## Additional Inherited Members

### 5.382.1 Constructor & Destructor Documentation

**5.382.1.1 noise\_t()** `noise_t::noise_t (`  
    `algo_comm_t iac,`  
    `const std::string & configured_name )`

### 5.382.2 Member Function Documentation

**5.382.2.1 process() [1/2]** `mha_wave_t * noise_t::process (`  
    `mha_wave_t * s )`

**5.382.2.2 process() [2/2]** `mha_spec_t * noise_t::process (`  
    `mha_spec_t * s )`

**5.382.2.3 `prepare()`** `void noise_t::prepare ( mhaconfig_t & tf ) [virtual]`

Implements `MHAPlugIn::plugin_t< cfg_t >` (p. 1149).

**5.382.2.4 `update_cfg()`** `void noise_t::update_cfg ( )`

### 5.382.3 Member Data Documentation

**5.382.3.1 `lev`** `MHAParser::float_t noise_t::lev [private]`

**5.382.3.2 `mode`** `MHAParser::kw_t noise_t::mode [private]`

**5.382.3.3 `frozennoise_length`** `MHAParser::float_t noise_t::frozennoise_length [private]`

**5.382.3.4 `seed`** `MHAParser::int_t noise_t::seed [private]`

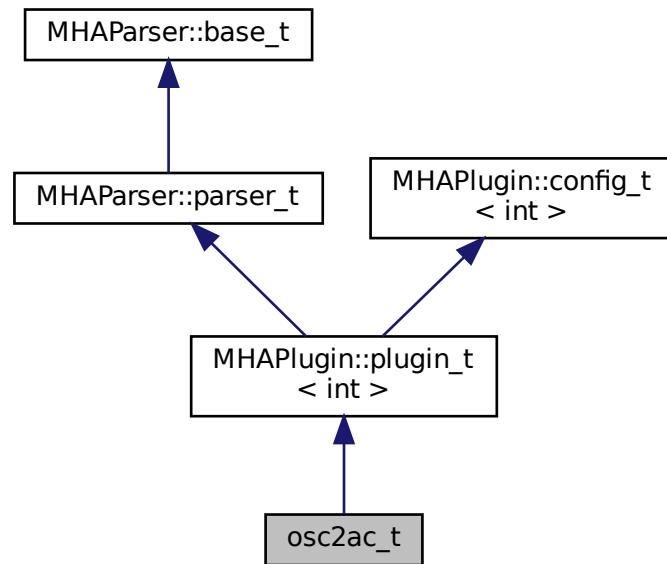
**5.382.3.5 `patchbay`** `MHAEvents::patchbay_t< noise_t > noise_t::patchbay [private]`

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

- `noise.cpp`

## 5.383 osc2ac\_t Class Reference

Inheritance diagram for osc2ac\_t:



### Public Member Functions

- `osc2ac_t ( algo_comm_t iac, const std::string &configured_name)`
- `void prepare ( mhaconfig_t &)`
- `void release ()`
- `mha_wave_t * process ( mha_wave_t *s)`
- `mha_spec_t * process ( mha_spec_t *s)`
- `void process ()`

### Private Member Functions

- `void setlock (bool b)`

### Private Attributes

- `MHAParser::string_t host`
- `MHAParser::string_t port`
- `MHAParser::vstring_t vars`
- `MHAParser::vint_t size`
- `MHAEvents::patchbay_t< osc2ac_t > patchbay`
- `std::unique_ptr< osc_server_t > srv`

## Additional Inherited Members

### 5.383.1 Constructor & Destructor Documentation

**5.383.1.1 `osc2ac_t()`** `osc2ac_t::osc2ac_t (`  
    `algo_comm_t iac,`  
    `const std::string & configured_name )`

### 5.383.2 Member Function Documentation

**5.383.2.1 `prepare()`** `void osc2ac_t::prepare (`  
    `mhaconfig_t & ) [virtual]`

Implements `MHAPlugin::plugin_t< int >` (p. [1149](#)).

**5.383.2.2 `release()`** `void osc2ac_t::release ( ) [virtual]`

Reimplemented from `MHAPlugin::plugin_t< int >` (p. [1150](#)).

**5.383.2.3 `process()` [1/3]** `mha_wave_t* osc2ac_t::process (`  
    `mha_wave_t * s ) [inline]`

**5.383.2.4 `process()` [2/3]** `mha_spec_t* osc2ac_t::process (`  
    `mha_spec_t * s ) [inline]`

**5.383.2.5 process()** [3/3] void osc2ac\_t::process ( )

**5.383.2.6 setlock()** void osc2ac\_t::setlock ( bool b ) [private]

### 5.383.3 Member Data Documentation

**5.383.3.1 host** MHAParser::string\_t osc2ac\_t::host [private]

**5.383.3.2 port** MHAParser::string\_t osc2ac\_t::port [private]

**5.383.3.3 vars** MHAParser::vstring\_t osc2ac\_t::vars [private]

**5.383.3.4 size** MHAParser::vint\_t osc2ac\_t::size [private]

**5.383.3.5 patchbay** MHAEEvents::patchbay\_t< osc2ac\_t> osc2ac\_t::patchbay [private]

**5.383.3.6 srv** std::unique\_ptr< osc\_server\_t> osc2ac\_t::srv [private]

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

- **osc2ac.cpp**

## 5.384 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`

### 5.384.1 Detailed Description

OSC receiver implemented using liblo.

### 5.384.2 Constructor & Destructor Documentation

```
5.384.2.1 osc_server_t() osc_server_t::osc_server_t (
    const std::string & multicast_addr,
    const std::string & port )
```

**5.384.2.2 ~osc\_server\_t()** `osc_server_t::~osc_server_t ( )`

### 5.384.3 Member Function Documentation

**5.384.3.1 server\_stop()** `void osc_server_t::server_stop ( )`

**5.384.3.2 server\_start()** `void osc_server_t::server_start ( )`

**5.384.3.3 insert\_variable()** `void osc_server_t::insert_variable (`  
    `const std::string & name,`  
    `unsigned int size,`  
    `algo_comm_t hAC )`

**5.384.3.4 sync\_osc2ac()** `void osc_server_t::sync_osc2ac ( )`

**5.384.3.5 ac\_insert()** `void osc_server_t::ac_insert ( )`

**5.384.3.6 error\_h()** `void osc_server_t::error_h (`  
    `int num,`  
    `const char * msg,`  
    `const char * path ) [static]`

### 5.384.4 Member Data Documentation

**5.384.4.1 pVars** std::vector<std::unique\_ptr<**osc\_variable\_t**> > osc\_server\_t::pVars [private]

**5.384.4.2 lost** lo\_server\_thread osc\_server\_t::lost [private]

**5.384.4.3 is\_running** bool osc\_server\_t::is\_running [private]

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

- **osc2ac.cpp**

## 5.385 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.*

### 5.385.1 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. 1322), one for each mapping of an OSC address to an AC variable.

### 5.385.2 Constructor & Destructor Documentation

#### 5.385.2.1 osc\_variable\_t() [1/2] osc\_variable\_t::osc\_variable\_t (

```
const osc_variable_t & ) [delete]
```

An instance of this class cannot safely be copied.

#### 5.385.2.2 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.

## 5.385.3 Member Function Documentation

### 5.385.3.1 sync\_osc2ac() void osc\_variable\_t::sync\_osc2ac ( ) [inline]

Copies the latest OSC data from the OSC storage to the AC storage.

To be executed during process callback of osc2ac plugin.

### 5.385.3.2 ac\_insert() void osc\_variable\_t::ac\_insert ( ) [inline]

Insert/Re-insert the AC variable into AC space.

Should be done in each process callback.

### 5.385.3.3 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>msg</i>	Unused.
<i>user_data</i>	Pointer to <b>osc_variable_t</b> (p. 1322) instance.

**Returns**

1 if the message was accepted, 0 if not.

**5.385.3.4 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**

<code>types</code>	The OSC data type indicator of the received message.
<code>argv</code>	Array of received OSC data.
<code>argc</code>	Number of elements in array of received OSC data.

**Returns**

1 if the message had correct length and contained only floats, 0 if not.

## 5.385.4 Member Data Documentation

**5.385.4.1 acname** `std::string osc_variable_t::acname` [private]

Name of the ac variable.

**5.385.4.2 oscaddr** `std::string osc_variable_t::oscaddr` [private]

OSC address.

**5.385.4.3 ac\_data** `MHA_AC::waveform_t` `osc_variable_t::ac_data` [private]

AC variable storage.

**5.385.4.4 osc\_data** `MHASignal::waveform_t` `osc_variable_t::osc_data` [private]

OSC variable storage.

**5.385.4.5 name\_** `std::string` `osc_variable_t::name_` [private]

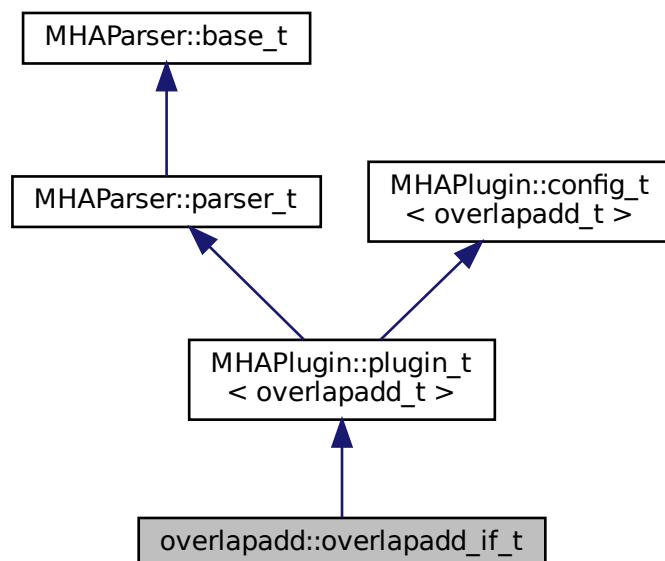
Name of AC variable and OSC address without the initial slash.

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

- `osc2ac.cpp`

## 5.386 overlapadd::overlapadd\_if\_t Class Reference

Inheritance diagram for overlapadd::overlapadd\_if\_t:



## Public Member Functions

- **overlapadd\_if\_t ( algo\_comm\_t iac, const std::string &configured\_name)**
- **~overlapadd\_if\_t ()=default**
- **void prepare ( mhaconfig\_t &)**
- **void release ()**
- **mha\_wave\_t \* process ( mha\_wave\_t \*)**

## Private Member Functions

- **void update ()**
- **void setlock (bool b)**

*Lock/Unlock all configuration variables.*

## Private Attributes

- **MHAParser::int\_t nfft**  
*FFT length to be used, zero-padding is FFT length-wndlength.*
- **MHAParser::int\_t nwnd**  
*Window length to be used (overlap is 1-fragsize/wndlength)*
- **MHAParser::float\_t wndpos**  
*Relative position of zero padding (0 end, 0.5 center, 1 start)*
- **MHAParser::window\_t window**
- **MHAParser::float\_t wndexp**
- **MHAParser::window\_t zerowindow**
- **MHAParser::bool\_t strict\_window\_ratio**  
*Disallow window sizes that are not a multiple of the hop size ("fragsize" in MHA) a by power of two.*
- **MHAParser::mhapluginloader\_t plugloader**
- **MHAParser::float\_mon\_t prescale**
- **MHAParser::float\_mon\_t postscale**
- **std::string algo**
- **mhaconfig\_t cf\_in**
- **mhaconfig\_t cf\_out**

## Additional Inherited Members

### 5.386.1 Constructor & Destructor Documentation

---

**5.386.1.1 `overlapadd_if_t()`** `overlapadd::overlapadd_if_t::overlapadd_if_t ( algo_comm_t iac,`  
`const std::string & configured_name )`

**5.386.1.2 `~overlapadd_if_t()`** `overlapadd::overlapadd_if_t::~overlapadd_if_t ( )`  
`[default]`

## 5.386.2 Member Function Documentation

**5.386.2.1 `prepare()`** `void overlapadd::overlapadd_if_t::prepare ( mhaconfig_t & t ) [virtual]`

Implements `MHAPlugin::plugin_t< overlapadd_t >` (p. [1149](#)).

**5.386.2.2 `release()`** `void overlapadd::overlapadd_if_t::release ( ) [virtual]`

Reimplemented from `MHAPlugin::plugin_t< overlapadd_t >` (p. [1150](#)).

**5.386.2.3 `process()`** `mha_wave_t * overlapadd::overlapadd_if_t::process ( mha_wave_t * wave_in )`

**5.386.2.4 `update()`** `void overlapadd::overlapadd_if_t::update ( ) [private]`

**5.386.2.5 `setlock()`** `void overlapadd::overlapadd_if_t::setlock ( bool b ) [inline], [private]`

Lock/Unlock all configuration variables.

**Parameters**

<i>b</i>	Desired lock state
----------	--------------------

**5.386.3 Member Data Documentation****5.386.3.1 nfft    MHAParser::int\_t overlapadd::overlapadd\_if\_t::nfft [private]**

FFT length to be used, zero-padding is FFT length-wndlength.

**5.386.3.2 nwnd    MHAParser::int\_t overlapadd::overlapadd\_if\_t::nwnd [private]**

Window length to be used (overlap is 1-fragsize/wndlength)

**5.386.3.3 wndpos    MHAParser::float\_t overlapadd::overlapadd\_if\_t::wndpos [private]**

Relative position of zero padding (0 end, 0.5 center, 1 start)

**5.386.3.4 window    MHAParser::window\_t overlapadd::overlapadd\_if\_t::window [private]****5.386.3.5 wndexp    MHAParser::float\_t overlapadd::overlapadd\_if\_t::wndexp [private]****5.386.3.6 zerowindow    MHAParser::window\_t overlapadd::overlapadd\_if\_t::zerowindow [private]**

**5.386.3.7 strict\_window\_ratio** `MHAParser::bool_t overlapadd::overlapadd_if_t::strict_window_ratio [private]`

Disallow window sizes that are not a multiple of the hop size ("fragsize" in MHA) a by power of two.

**5.386.3.8 plugloader** `MHAParser::mhaplugloader_t overlapadd::overlapadd_if_t::plugloader [private]`

**5.386.3.9 prescale** `MHAParser::float_mon_t overlapadd::overlapadd_if_t::prescale [private]`

**5.386.3.10 postscale** `MHAParser::float_mon_t overlapadd::overlapadd_if_t::postscale [private]`

**5.386.3.11 algo** `std::string overlapadd::overlapadd_if_t::algo [private]`

**5.386.3.12 cf\_in** `mhaconfig_t overlapadd::overlapadd_if_t::cf_in [private]`

**5.386.3.13 cf\_out** `mhaconfig_t overlapadd::overlapadd_if_t::cf_out [private]`

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

- `overlapadd.hh`
- `overlapadd.cpp`

## 5.387 overlapadd::overlapadd\_t Class Reference

### Public Member Functions

- `overlapadd_t ( mhaconfig_t spar_in, mhaconfig_t spar_out, float wexp, float wndpos, const MHAParser::window_t &window, const MHAParser::window_t &zerowindow, float &prescale_fac, float &postscale_fac)`
- `~overlapadd_t ()`
- `mha_spec_t * wave2spec ( mha_wave_t *)`
- `mha_wave_t * spec2wave ( mha_spec_t *)`

### Private Member Functions

- `void wave2spec_hop_forward ( mha_wave_t *)`
- `void wave2spec_apply_window (void)`
- `mha_spec_t * wave2spec_compute_fft (void)`

### Private Attributes

- `mha_fft_t fft`
- `MHAWindow::base_t prewnd`
- `MHAWindow::base_t postwnd`
- `MHASignal::waveform_t wave_in1`
- `MHASignal::waveform_t wave_out1`
- `MHASignal::spectrum_t spec_in`
- `MHASignal::waveform_t calc_out`
- `MHASignal::waveform_t out_buf`
- `MHASignal::waveform_t write_buf`
- `unsigned int n_zero`
- `unsigned int n_pad1`
- `unsigned int n_pad2`

### 5.387.1 Constructor & Destructor Documentation

#### 5.387.1.1 overlapadd\_t() overlapadd::overlapadd\_t::overlapadd\_t (

```
    mhaconfig_t spar_in,
    mhaconfig_t spar_out,
    float wexp,
    float wndpos,
    const MHAParser::window_t & window,
    const MHAParser::window_t & zerowindow,
    float & prescale_fac,
    float & postscale_fac )
```

**5.387.1.2 ~overlapadd\_t()** overlapadd::overlapadd\_t::~overlapadd\_t ( )

## 5.387.2 Member Function Documentation

**5.387.2.1 wave2spec()** mha\_spec\_t \* overlapadd::overlapadd\_t::wave2spec ( mha\_wave\_t \* s )

**5.387.2.2 spec2wave()** mha\_wave\_t \* overlapadd::overlapadd\_t::spec2wave ( mha\_spec\_t \* s )

**5.387.2.3 wave2spec\_hop\_forward()** void overlapadd::overlapadd\_t::wave2spec\_hop\_forward ( mha\_wave\_t \* s ) [private]

**5.387.2.4 wave2spec\_apply\_window()** void overlapadd::overlapadd\_t::wave2spec\_apply\_window ( void ) [private]

**5.387.2.5 wave2spec\_compute\_fft()** mha\_spec\_t \* overlapadd::overlapadd\_t::wave2spec\_compute\_fft ( void ) [private]

## 5.387.3 Member Data Documentation

**5.387.3.1 fft** mha\_fft\_t overlapadd::overlapadd\_t::fft [private]

**5.387.3.2 prewnd** `MHAWindow::base_t` `overlapadd::overlapadd_t::prewnd` [private]

**5.387.3.3 postwnd** `MHAWindow::base_t` `overlapadd::overlapadd_t::postwnd` [private]

**5.387.3.4 wave\_in1** `MHASignal::waveform_t` `overlapadd::overlapadd_t::wave_in1` [private]

**5.387.3.5 wave\_out1** `MHASignal::waveform_t` `overlapadd::overlapadd_t::wave_out1` [private]

**5.387.3.6 spec\_in** `MHASignal::spectrum_t` `overlapadd::overlapadd_t::spec_in` [private]

**5.387.3.7 calc\_out** `MHASignal::waveform_t` `overlapadd::overlapadd_t::calc_out` [private]

**5.387.3.8 out\_buf** `MHASignal::waveform_t` `overlapadd::overlapadd_t::out_buf` [private]

**5.387.3.9 write\_buf** `MHASignal::waveform_t` `overlapadd::overlapadd_t::write_buf` [private]

**5.387.3.10 n\_zero** `unsigned int` `overlapadd::overlapadd_t::n_zero` [private]

**5.387.3.11 n\_pad1** unsigned int overlapadd::overlapadd\_t::n\_pad1 [private]

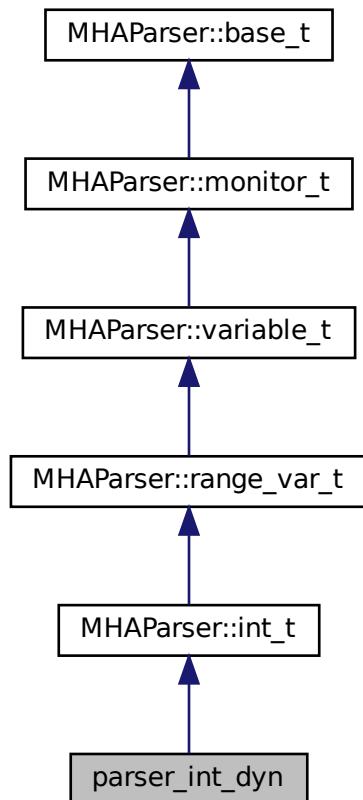
**5.387.3.12 n\_pad2** unsigned int overlapadd::overlapadd\_t::n\_pad2 [private]

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

- **overlapadd.hh**
- **overlapadd.cpp**

## 5.388 parser\_int\_dyn Class Reference

Inheritance diagram for parser\_int\_dyn:



## Public Member Functions

- **parser\_int\_dyn** (const std::string &help\_text, const std::string &initial\_value, const std::string & range)
- void **set\_max\_angle\_ind** (unsigned int max\_ind)

## Additional Inherited Members

### 5.388.1 Constructor & Destructor Documentation

```
5.388.1.1 parser_int_dyn() parser_int_dyn::parser_int_dyn (
    const std::string & help_text,
    const std::string & initial_value,
    const std::string & range ) [inline]
```

### 5.388.2 Member Function Documentation

```
5.388.2.1 set_max_angle_ind() void parser_int_dyn::set_max_angle_ind (
    unsigned int max_ind ) [inline]
```

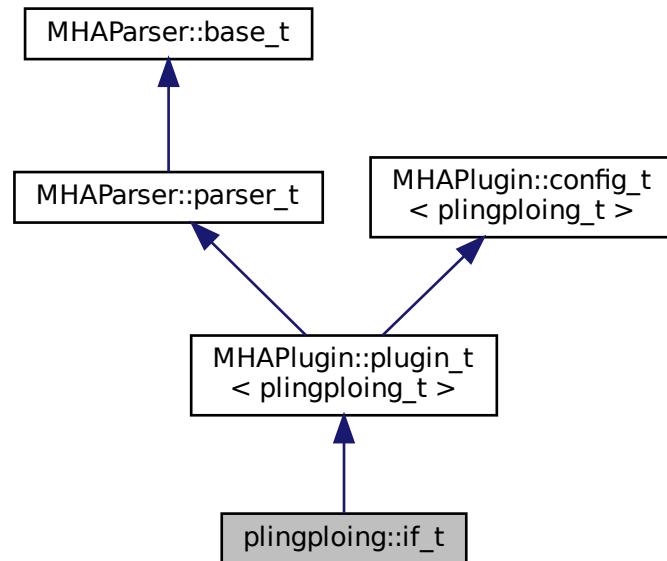
The documentation for this class was generated from the following file:

- **steerbf.h**

## 5.389 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 iac, const std::string &configured\_name)**
- **mha\_wave\_t \* process ( mha\_wave\_t \*)**
- **void prepare ( mhaconfig\_t &cf)**

### Private Member Functions

- **void update ()**

### Private Attributes

- **MHAEvents::patchbay\_t< if\_t > patchbay**
- **MHAParser::float\_t level**  
*Output level in dB SPL.*
- **MHAParser::float\_t pitch**  
*Bass pitch in Hz.*
- **MHAParser::float\_t fun1\_key**  
*Key1.*
- **MHAParser::float\_t fun1\_range**  
*Range1.*
- **MHAParser::float\_t fun2\_key**  
*Key 2.*
- **MHAParser::float\_t fun2\_range**  
*Range 2.*
- **MHAParser::float\_t bpm**  
*Speed in beats per minute (bpm)*
- **MHAParser::float\_t minlen**  
*Minimum note length in beats.*
- **MHAParser::float\_t maxlen**  
*Maximum note length in beats.*
- **MHAParser::float\_t bassmod**  
*Bass key modulation depth.*
- **MHAParser::float\_t bassperiod**  
*Bass key modulation period.*

### Additional Inherited Members

#### 5.389.1 Detailed Description

Plugin class of the plingploing music generator.

#### 5.389.2 Constructor & Destructor Documentation

##### 5.389.2.1 if\_t() plingploing::if\_t::if\_t (

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

### 5.389.3 Member Function Documentation

**5.389.3.1 `process()`** `mha_wave_t * plingploing::if_t::process ( mha_wave_t * s )`

**5.389.3.2 `prepare()`** `void plingploing::if_t::prepare ( mhaconfig_t & cf ) [virtual]`

Implements `MHAPlugin::plugin_t< plingploing_t >` (p. [1149](#)).

**5.389.3.3 `update()`** `void plingploing::if_t::update ( ) [private]`

### 5.389.4 Member Data Documentation

**5.389.4.1 `patchbay`** `MHAEEvents::patchbay_t< if_t > plingploing::if_t::patchbay [private]`

**5.389.4.2 `level`** `MHAParser::float_t plingploing::if_t::level [private]`

Output level in dB SPL.

**5.389.4.3 `pitch`** `MHAParser::float_t plingploing::if_t::pitch [private]`

Bass pitch in Hz.

**5.389.4.4 fun1\_key** `MHAParser::float_t` `plingploing::if_t::fun1_key` [private]

Key1.

**5.389.4.5 fun1\_range** `MHAParser::float_t` `plingploing::if_t::fun1_range` [private]

Range1.

**5.389.4.6 fun2\_key** `MHAParser::float_t` `plingploing::if_t::fun2_key` [private]

Key 2.

**5.389.4.7 fun2\_range** `MHAParser::float_t` `plingploing::if_t::fun2_range` [private]

Range 2.

**5.389.4.8 bpm** `MHAParser::float_t` `plingploing::if_t::bpm` [private]

Speed in beats per minute (bpm)

**5.389.4.9 minlen** `MHAParser::float_t` `plingploing::if_t::minlen` [private]

Minimum note length in beats.

**5.389.4.10 maxlen** `MHAParser::float_t` `plingploing::if_t::maxlen` [private]

Maximum note length in beats.

**5.389.4.11 bassmod** `MHAParser::float_t plingploing::if_t::bassmod` [private]

Bass key modulation depth.

**5.389.4.12 bassperiod** `MHAParser::float_t plingploing::if_t::bassperiod` [private]

Bass key modulation period.

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

- `plingploing.cpp`

**5.390 plingploing::plingploing\_t Class Reference**

Run-time configuration of the plingploing music generator.

**Public Member Functions**

- `plingploing_t ( mhaconfig_t, mha_real_t level, mha_real_t pitch, mha_real_t k1, mha_real_t k2, mha_real_t i1, mha_real_t i2, mha_real_t bpm, mha_real_t minlen, mha_real_t maxlen, mha_real_t bassmod, mha_real_t bassperiod)`
- `void process ( mha_wave_t *)`

**Private Attributes**

- `mhaconfig_t cf`
- `mha_real_t pitch_`
- `unsigned int bt`
- `unsigned int t`
- `unsigned int len`
- `mha_real_t dur_`
- `mha_real_t minlen_`
- `mha_real_t maxlen_`
- `mha_real_t bass`
- `mha_real_t freq`
- `mha_real_t fun1_key`
- `mha_real_t fun1_range`
- `mha_real_t fun1`
- `mha_real_t fun2`
- `mha_real_t fun2_key`

- `mha_real_t fun2_range`
- `mha_real_t dist`
- `mha_real_t dist1`
- `mha_real_t alph`
- `mha_real_t rms`
- `mha_real_t bassmod_`
- `mha_real_t bassperiod_`
- `MHAWindow::hanning_t hann1`
- `MHAWindow::hanning_t hann2`
- `mha_real_t level`

### 5.390.1 Detailed Description

Run-time configuration of the plingploing music generator.

### 5.390.2 Constructor & Destructor Documentation

#### 5.390.2.1 `plingploing_t()` `plingploing::plingploing_t::plingploing_t (`

```
    mhaconfig_t c,  
    mha_real_t level,  
    mha_real_t pitch,  
    mha_real_t k1,  
    mha_real_t k2,  
    mha_real_t i1,  
    mha_real_t i2,  
    mha_real_t bpm,  
    mha_real_t minlen,  
    mha_real_t maxlen,  
    mha_real_t bassmod,  
    mha_real_t bassperiod )
```

### 5.390.3 Member Function Documentation

#### 5.390.3.1 `process()` `void plingploing::plingploing_t::process (`

```
    mha_wave_t * s )
```

## 5.390.4 Member Data Documentation

**5.390.4.1 cf** `mhaconfig_t` `plingploing::plingploing_t::cf` [private]

**5.390.4.2 pitch\_** `mha_real_t` `plingploing::plingploing_t::pitch_` [private]

**5.390.4.3 bt** `unsigned int` `plingploing::plingploing_t::bt` [private]

**5.390.4.4 t** `unsigned int` `plingploing::plingploing_t::t` [private]

**5.390.4.5 len** `unsigned int` `plingploing::plingploing_t::len` [private]

**5.390.4.6 dur\_** `mha_real_t` `plingploing::plingploing_t::dur_` [private]

**5.390.4.7 minlen\_** `mha_real_t` `plingploing::plingploing_t::minlen_` [private]

**5.390.4.8 maxlen\_** `mha_real_t` `plingploing::plingploing_t::maxlen_` [private]

**5.390.4.9 bass** `mha_real_t` `plingploing::plingploing_t::bass` [private]

**5.390.4.10 freq** `mha_real_t` `plingploing::plingploing_t::freq` [private]

**5.390.4.11 fun1\_key** `mha_real_t` `plingploing::plingploing_t::fun1_key` [private]

**5.390.4.12 fun1\_range** `mha_real_t` `plingploing::plingploing_t::fun1_range` [private]

**5.390.4.13 fun1** `mha_real_t` `plingploing::plingploing_t::fun1` [private]

**5.390.4.14 fun2** `mha_real_t` `plingploing::plingploing_t::fun2` [private]

**5.390.4.15 fun2\_key** `mha_real_t` `plingploing::plingploing_t::fun2_key` [private]

**5.390.4.16 fun2\_range** `mha_real_t` `plingploing::plingploing_t::fun2_range` [private]

**5.390.4.17 dist** `mha_real_t` `plingploing::plingploing_t::dist` [private]

**5.390.4.18 dist1** `mha_real_t` `plingploing::plingploing_t::dist1` [private]

**5.390.4.19 alph** `mha_real_t` `plingploing::plingploing_t::alph` [private]

**5.390.4.20 rms** `mha_real_t` `plingploing::plingploing_t::rms` [private]

**5.390.4.21 bassmod\_** `mha_real_t` `plingploing::plingploing_t::bassmod_` [private]

**5.390.4.22 bassperiod\_** `mha_real_t` `plingploing::plingploing_t::bassperiod_` [private]

**5.390.4.23 hann1** `MHAWindow::hanning_t` `plingploing::plingploing_t::hann1` [private]

**5.390.4.24 hann2** `MHAWindow::hanning_t` `plingploing::plingploing_t::hann2` [private]

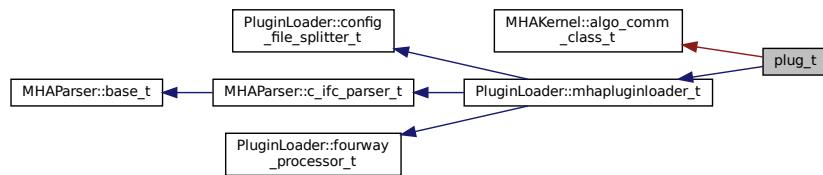
**5.390.4.25 level** `mha_real_t` `plingploing::plingploing_t::level` [private]

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

- `plingploing.cpp`

## 5.391 plug\_t Class Reference

Inheritance diagram for plug\_t:



### Public Member Functions

- `plug_t (const std::string & libname)`
- `~plug_t () throw ()`
- `MHAProc_wave2wave_t get_process_wave ()`
- `MHAProc_wave2spec_t get_process_spec ()`
- `void * get_handle ()`
- `algo_comm_t get_ac ()`

### Additional Inherited Members

#### 5.391.1 Constructor & Destructor Documentation

**5.391.1.1 plug\_t()** `plug_t::plug_t (const std::string & libname )`

**5.391.1.2 ~plug\_t()** `plug_t::~plug_t ( ) throw () [inline]`

#### 5.391.2 Member Function Documentation

**5.391.2.1 get\_process\_wave()** `MHAProc_wave2wave_t plug_t::get_process_wave ( )`

**5.391.2.2 get\_process\_spec()** `MHAProc_wave2spec_t plug_t::get_process_spec ( )`

**5.391.2.3 get\_handle()** `void * plug_t::get_handle ( )`

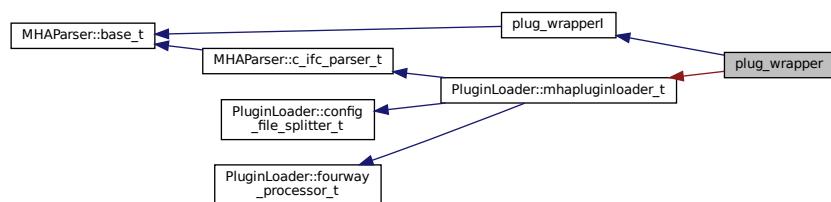
**5.391.2.4 get\_ac()** `algo_comm_t plug_t::get_ac ( ) [inline]`

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

- `analysispath.cpp`

## 5.392 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

### 5.392.1 Constructor & Destructor Documentation

#### 5.392.1.1 **plug\_wrapper()** plug\_wrapper::plug\_wrapper (

```
    algo_comm_t iac,  
    const std::string & libname ) [inline]
```

#### 5.392.1.2 ~**plug\_wrapper()** virtual plug\_wrapper::~plug\_wrapper ( ) [virtual], [default]

### 5.392.2 Member Function Documentation

#### 5.392.2.1 **get\_categories()** virtual std::vector<std::string> plug\_wrapper::get\_← categories ( ) [inline], [virtual]

Implements **plug\_wrapperI** (p. 1349).

#### 5.392.2.2 **parse()** virtual std::string plug\_wrapper::parse (

```
    const std::string & str ) [inline], [virtual]
```

Reimplemented from **PluginLoader::mhapluginloader\_t** (p. 1367).

#### 5.392.2.3 **has\_parser()** virtual bool plug\_wrapper::has\_parser ( ) [inline], [virtual]

Implements **plug\_wrapperI** (p. 1349).

**5.392.2.4 `get_documentation()`** `virtual std::string plug_wrapper::get_documentation()  
() [inline], [virtual]`

Implements **plug\_wrapperl** (p. 1349).

**5.392.2.5 `has_process()`** `virtual bool plug_wrapper::has_process(  
mha_domain_t in,  
mha_domain_t out) [inline], [virtual]`

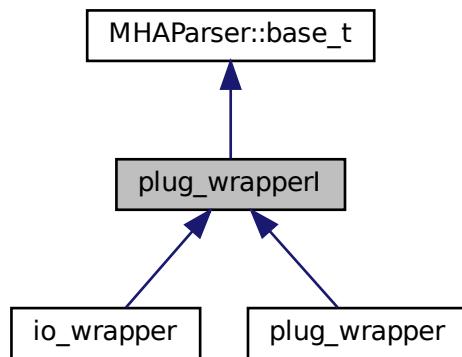
Implements **plug\_wrapperl** (p. 1350).

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

- `generatemhaplugindoc.cpp`

## 5.393 `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

### 5.393.1 Constructor & Destructor Documentation

**5.393.1.1 plug\_wrapperI()** `plug_wrapperI::plug_wrapperI ( ) [inline]`

**5.393.1.2 ~plug\_wrapperI()** `virtual plug_wrapperI::~plug_wrapperI ( ) [virtual], [default]`

### 5.393.2 Member Function Documentation

**5.393.2.1 get\_categories()** `virtual std::vector<std::string> plug_wrapperI::get_← categories ( ) [pure virtual]`

Implemented in **plug\_wrapper** (p. 1347), and **io\_wrapper** (p. 640).

**5.393.2.2 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. 1347), and **io\_wrapper** (p. 640).

**5.393.2.3 has\_parser()** `virtual bool plug_wrapperI::has_parser ( ) [pure virtual]`

Implemented in **plug\_wrapper** (p. 1347), and **io\_wrapper** (p. 640).

**5.393.2.4 get\_documentation()** `virtual std::string plug_wrapperI::get_documentation()  
() [pure virtual]`

Implemented in **plug\_wrapper** (p. 1347), and **io\_wrapper** (p. 640).

**5.393.2.5 has\_process()** `virtual bool plug_wrapperI::has_process(  
mha_domain_t ,  
mha_domain_t ) [pure virtual]`

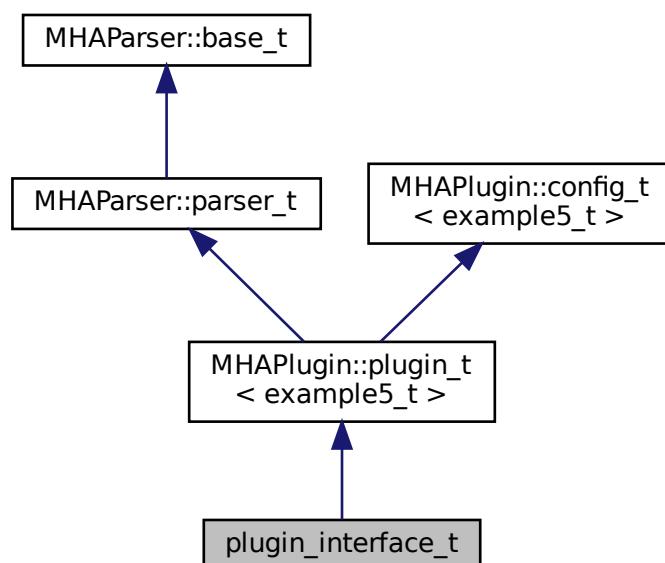
Implemented in **plug\_wrapper** (p. 1348), and **io\_wrapper** (p. 640).

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

- `generatemhaplugindoc.cpp`

## 5.394 plugin\_interface\_t Class Reference

Inheritance diagram for plugin\_interface\_t:



## Public Member Functions

- `plugin_interface_t ( algo_comm_t iac, const std::string &configured_name)`
- `mha_spec_t * process ( mha_spec_t *)`
- `void prepare ( mhaconfig_t &)`

## Private Member Functions

- `void update_cfg ()`

## Private Attributes

- `MHAParser::int_t scale_ch`
- `MHAParser::float_t factor`
- `MHAEvents::patchbay_t< plugin_interface_t > patchbay`

## Additional Inherited Members

### 5.394.1 Constructor & Destructor Documentation

```
5.394.1.1 plugin_interface_t() plugin_interface_t::plugin_interface_t (
    algo_comm_t iac,
    const std::string & configured_name )
```

### 5.394.2 Member Function Documentation

```
5.394.2.1 process() mha_spec_t * plugin_interface_t::process (
    mha_spec_t * spec )
```

---

**5.394.2.2 `prepare()`** `void plugin_interface_t::prepare ( mhaconfig_t & tfcfg ) [virtual]`

Implements **MHAPlugIn::plugin\_t< example5\_t >** (p. 1149).

**5.394.2.3 `update_cfg()`** `void plugin_interface_t::update_cfg ( ) [private]`

### 5.394.3 Member Data Documentation

**5.394.3.1 `scale_ch`** `MHAParser::int_t plugin_interface_t::scale_ch [private]`

**5.394.3.2 `factor`** `MHAParser::float_t plugin_interface_t::factor [private]`

**5.394.3.3 `patchbay`** `MHAEvents::patchbay_t< plugin_interface_t > plugin_interface_t::patchbay [private]`

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

- `example5.cpp`

## 5.395 pluginbrowser\_t Class Reference

### Public Member Functions

- `pluginbrowser_t ()`
- `void get_paths ()`
- `plugindescription_t scan_plugin (const std::string &name)`
- `void add_plugins ()`
- `void clear_plugins ()`
- `void scan_plugins ()`
- `void add_plugin (const std::string &name)`
- `std::list< plugindescription_t > get_plugins () const`

## Private Attributes

- std::string **plugin\_extension**
- std::list< std::string > **library\_paths**
- std::list< **plugindescription\_t** > **plugins**
- std::map< std::string, **pluginloader\_t** \* > **p**

## 5.395.1 Constructor & Destructor Documentation

**5.395.1.1 pluginbrowser\_t()** `pluginbrowser_t::pluginbrowser_t ( )`

## 5.395.2 Member Function Documentation

**5.395.2.1 get\_paths()** `void pluginbrowser_t::get_paths ( )`

**5.395.2.2 scan\_plugin()** `plugindescription_t pluginbrowser_t::scan_plugin (`  
`const std::string & name )`

**5.395.2.3 add\_plugins()** `void pluginbrowser_t::add_plugins ( )`

**5.395.2.4 clear\_plugins()** `void pluginbrowser_t::clear_plugins ( )`

**5.395.2.5 scan\_plugins()** `void pluginbrowser_t::scan_plugins ( )`

**5.395.2.6 add\_plugin()** void pluginbrowser\_t::add\_plugin ( const std::string & name )

**5.395.2.7 get\_plugins()** std::list< **plugindescription\_t**> pluginbrowser\_t::get\_plugins ( ) const [inline]

### 5.395.3 Member Data Documentation

**5.395.3.1 plugin\_extension** std::string pluginbrowser\_t::plugin\_extension [private]

**5.395.3.2 library\_paths** std::list<std::string> pluginbrowser\_t::library\_paths [private]

**5.395.3.3 plugins** std::list< **plugindescription\_t**> pluginbrowser\_t::plugins [private]

**5.395.3.4 p** std::map<std::string, **pluginloader\_t\***> pluginbrowser\_t::p [private]

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

- **pluginbrowser.h**
- **pluginbrowser.cpp**

## 5.396 **plugindescription\_t** Class Reference

### Public Attributes

- std::string **name**
- std::string **fullname**
- std::string **documentation**
- std::vector< std::string > **categories**
- bool **wave2wave**
- bool **wave2spec**
- bool **spec2wave**
- bool **spec2spec**
- std::vector< std::string > **query\_cmds**
- std::map< std::string, std::string > **queries**

### 5.396.1 Member Data Documentation

**5.396.1.1 name** std::string plugindescription\_t::name

**5.396.1.2 fullname** std::string plugindescription\_t::fullname

**5.396.1.3 documentation** std::string plugindescription\_t::documentation

**5.396.1.4 categories** std::vector<std::string> plugindescription\_t::categories

**5.396.1.5 wave2wave** bool plugindescription\_t::wave2wave

**5.396.1.6 `wave2spec`** bool plugindescription\_t::wave2spec

**5.396.1.7 `spec2wave`** bool plugindescription\_t::spec2wave

**5.396.1.8 `spec2spec`** bool plugindescription\_t::spec2spec

**5.396.1.9 `query_cmds`** std::vector<std::string> plugindescription\_t::query\_cmds

**5.396.1.10 `queries`** std::map<std::string, std::string> plugindescription\_t::queries

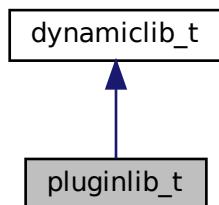
The documentation for this class was generated from the following file:

- `pluginbrowser.h`

## 5.397 `pluginlib_t` Class Reference

Specialisation of `dynamiclib_t` (p. 442) for mha plugin libraries.

Inheritance diagram for `pluginlib_t`:



## Public Member Functions

- **pluginlib\_t** (const std::string &name\_)  
*C'tor of the wrapper class.*
- virtual void \* **resolve** (const std::string &name\_) override  
*Resolves the plugin callback specified by name\_, e.g.*
- virtual ~**pluginlib\_t** ()  
*D'tor.*

## Additional Inherited Members

### 5.397.1 Detailed Description

Specialisation of **dynamiclib\_t** (p. 442) for mha plugin libraries.

### 5.397.2 Constructor & Destructor Documentation

#### 5.397.2.1 **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
--------------	---

#### 5.397.2.2 ~**pluginlib\_t()** pluginlib\_t::~pluginlib\_t ( ) [virtual]

D'tor.

### 5.397.3 Member Function Documentation

**5.397.3.1 `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

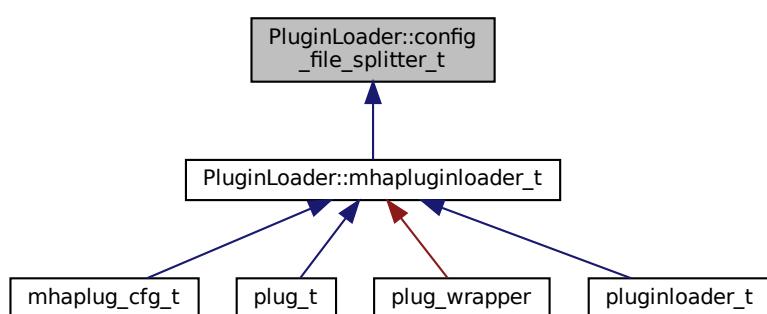
Reimplemented from **dynamiclib\_t** (p. 444).

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

- **mha\_os.h**
- **mha\_os.cpp**

### 5.398 PluginLoader::config\_file\_splitter\_t Class Reference

Inheritance diagram for `PluginLoader::config_file_splitter_t`:



## Public Member Functions

- `config_file_splitter_t (const std::string &name)`
- `const std::string & get_configname () const`
- `const std::string & get_libname () const`
- `const std::string & get_origname () const`
- `const std::string & get_configfile () const`

## Private Attributes

- `std::string libname`
- `std::string configname`
- `std::string origname`
- `std::string configfile`

### 5.398.1 Constructor & Destructor Documentation

**5.398.1.1 config\_file\_splitter\_t()** `PluginLoader::config_file_splitter_t::config_file_splitter_t (`  
`const std::string & name )`

### 5.398.2 Member Function Documentation

**5.398.2.1 get\_configname()** `const std::string& PluginLoader::config_file_splitter_t::get_configname ( ) const [inline]`

**5.398.2.2 get\_libname()** `const std::string& PluginLoader::config_file_splitter_t::get_libname ( ) const [inline]`

**5.398.2.3 `get_origname()`** const std::string& PluginLoader::config\_file\_splitter\_t::get\_origname ( ) const [inline]

**5.398.2.4 `get_configfile()`** const std::string& PluginLoader::config\_file\_splitter\_t::get\_configfile ( ) const [inline]

### 5.398.3 Member Data Documentation

**5.398.3.1 `libname`** std::string PluginLoader::config\_file\_splitter\_t::libname [private]

**5.398.3.2 `configname`** std::string PluginLoader::config\_file\_splitter\_t::configname [private]

**5.398.3.3 `origname`** std::string PluginLoader::config\_file\_splitter\_t::origname [private]

**5.398.3.4 `configfile`** std::string PluginLoader::config\_file\_splitter\_t::configfile [private]

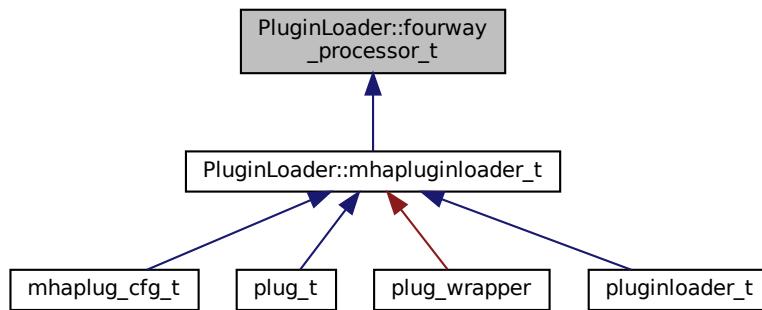
The documentation for this class was generated from the following files:

- **mhapluginloader.h**
- **mhapluginloader.cpp**

## 5.399 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. 1363) are released here in **fourway\_processor\_t::release** (p. 1364).*
- virtual std::string **parse** (const std::string &query)=0  
*Parser interface.*
- virtual ~**fourway\_processor\_t** ()  
*Classes with virtual methods need virtual destructor.*

### 5.399.1 Detailed Description

This abstract class defines the interface for classes that implement all types of signal domain processing supported by the MHA: wave2wave, spec2spec, wave2spec, and spec2wave.

For supporting different output domains for the same input domain, the processing methods are overloaded with respect to input domain and output domain.

## 5.399.2 Constructor & Destructor Documentation

**5.399.2.1 ~fourway\_processor\_t()** virtual PluginLoader::fourway\_processor\_t::~fourway\_processor\_t ( ) [inline], [virtual]

Classes with virtual methods need virtual destructor.

This destructor is empty.

## 5.399.3 Member Function Documentation

**5.399.3.1 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. 1368).

**5.399.3.2 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. 1368).

**5.399.3.3 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

<code>s_in</code>	input waveform signal
<code>s_out</code>	output spectrum signal

Implemented in [PluginLoader::mhapluginloader\\_t](#) (p. 1368).

**5.399.3.4 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

<code>s_in</code>	input spectrum signal
<code>s_out</code>	output waveform signal

Implemented in [PluginLoader::mhapluginloader\\_t](#) (p. 1368).

**5.399.3.5 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 <b>fourway_processor_t::prepare</b> (p. 1363), settings reflects domain and dimensions of the input signal. When <b>fourway_processor_t::prepare</b> (p. 1363) returns, settings reflects domain and dimensions of the output signal.
-----------------	---

Implemented in **PluginLoader::mhapluginloader\_t** (p. 1367).

**5.399.3.6 release()** `virtual void PluginLoader::fourway_processor_t::release () [pure virtual]`

Resources allocated for signal processing in **fourway\_processor\_t::prepare** (p. 1363) are released here in **fourway\_processor\_t::release** (p. 1364).

Implemented in **PluginLoader::mhapluginloader\_t** (p. 1368).

**5.399.3.7 parse()** `virtual std::string PluginLoader::fourway_processor_t::parse (const std::string & query) [pure virtual]`

Parser interface.

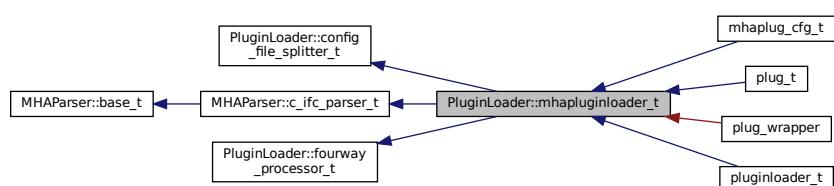
Implemented in **PluginLoader::mhapluginloader\_t** (p. 1367), and **plug\_wrapper** (p. 1347).

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

- **mhapluginloader.h**

## 5.400 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

### 5.400.1 Constructor & Destructor Documentation

#### 5.400.1.1 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.
----------------------	--

**5.400.1.2 ~mhapluginloader\_t()** `PluginLoader::mhapluginloader_t::~mhapluginloader_t ( ) throw ( )`

## 5.400.2 Member Function Documentation

**5.400.2.1 parse()** `std::string PluginLoader::mhapluginloader_t::parse ( const std::string & str ) [override], [virtual]`

Implements **PluginLoader::fourway\_processor\_t** (p. 1364).

Reimplemented in **plug\_wrapper** (p. 1347).

**5.400.2.2 has\_process()** `bool PluginLoader::mhapluginloader_t::has_process ( mha_domain_t in, mha_domain_t out ) const`

**5.400.2.3 has\_parser()** `bool PluginLoader::mhapluginloader_t::has_parser ( ) const`

**5.400.2.4 input\_domain()** `mha_domain_t PluginLoader::mhapluginloader_t::input_domain ( ) const`

**5.400.2.5 output\_domain()** `mha_domain_t PluginLoader::mhapluginloader_t::output_domain ( ) const`

**5.400.2.6 `prepare()`** void PluginLoader::mhaplugloader\_t::prepare (   
    mhaconfig\_t & tf ) [override], [virtual]

Implements **PluginLoader::fourway\_processor\_t** (p. 1363).

**5.400.2.7 `release()`** void PluginLoader::mhaplugloader\_t::release ( ) [override], [virtual]

Implements **PluginLoader::fourway\_processor\_t** (p. 1364).

**5.400.2.8 `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. 1362).

**5.400.2.9 `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. 1362).

**5.400.2.10 `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. 1363).

**5.400.2.11 `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. 1363).

**5.400.2.12 getfullname()** std::string PluginLoader::mhaplugloader\_t::getfullname ( ) const [inline]

**5.400.2.13 get\_documentation()** std::string PluginLoader::mhaplugloader\_t::get\_documentation ( ) const [inline]

**5.400.2.14 get\_categories()** std::vector<std::string> PluginLoader::mhaplugloader\_t::get\_categories ( ) const [inline]

**5.400.2.15 is\_prepared()** bool PluginLoader::mhaplugloader\_t::is\_prepared ( ) const [inline]

**5.400.2.16 test\_error()** void PluginLoader::mhaplugloader\_t::test\_error ( ) [protected]

**5.400.2.17 test\_version()** void PluginLoader::mhaplugloader\_t::test\_version ( ) [protected]

**5.400.2.18 mha\_test\_struct\_size()** void PluginLoader::mhaplugloader\_t::mha\_test\_struct\_size ( unsigned int s ) [protected]

**5.400.2.19 resolve\_and\_init()** void PluginLoader::mhaplugloader\_t::resolve\_and\_init ( ) [protected]

### 5.400.3 Member Data Documentation

**5.400.3.1 lib\_err** int PluginLoader::mhapluginloader\_t::lib\_err [protected]

**5.400.3.2 ac algo\_comm\_t** PluginLoader::mhapluginloader\_t::ac [protected]

**5.400.3.3 lib\_handle pluginlib\_t** PluginLoader::mhapluginloader\_t::lib\_handle [protected]

**5.400.3.4 lib\_data void\*** PluginLoader::mhapluginloader\_t::lib\_data [protected]

**5.400.3.5 MHAGetVersion\_cb MHAGetVersion\_t** PluginLoader::mhapluginloader\_t::MHAGetVersion\_cb [protected]

**5.400.3.6 MHAInit\_cb MHAInit\_t** PluginLoader::mhapluginloader\_t::MHAInit\_cb [protected]

**5.400.3.7 MHADestroy\_cb MHADestroy\_t** PluginLoader::mhapluginloader\_t::MHADestroy\_cb [protected]

**5.400.3.8 MHAPrepare\_cb MHAPrepare\_t** PluginLoader::mhapluginloader\_t::MHAPrepare\_cb [protected]

**5.400.3.9 MHARelease\_cb** `MHARelease_t` `PluginLoader::mhapluginloader_t::MHA::Release_cb` [protected]

**5.400.3.10 MHAProc\_wave2wave\_cb** `MHAProc_wave2wave_t` `PluginLoader::mhapluginloader_t::MHAProc_wave2wave_cb` [protected]

**5.400.3.11 MHAProc\_spec2spec\_cb** `MHAProc_spec2spec_t` `PluginLoader::mhapluginloader_t::MHAProc_spec2spec_cb` [protected]

**5.400.3.12 MHAProc\_wave2spec\_cb** `MHAProc_wave2spec_t` `PluginLoader::mhapluginloader_t::MHAProc_wave2spec_cb` [protected]

**5.400.3.13 MHAProc\_spec2wave\_cb** `MHAProc_spec2wave_t` `PluginLoader::mhapluginloader_t::MHAProc_spec2wave_cb` [protected]

**5.400.3.14 MHASet\_cb** `MHASet_t` `PluginLoader::mhapluginloader_t::MHASet_cb` [protected]

**5.400.3.15 MHASetcpp\_cb** `MHASetcpp_t` `PluginLoader::mhapluginloader_t::MHA::Setcpp_cb` [protected]

**5.400.3.16 MHAStrError\_cb** `MHAStrError_t` `PluginLoader::mhapluginloader_t::MHA::StrError_cb` [protected]

**5.400.3.17 cf\_input** `mhaconfig_t` `PluginLoader::mhaplugloader_t::cf_input` [protected]

**5.400.3.18 cf\_output** `mhaconfig_t` `PluginLoader::mhaplugloader_t::cf_output` [protected]

**5.400.3.19 plugin\_documentation** `std::string` `PluginLoader::mhaplugloader_t::plugin_documentation` [protected]

**5.400.3.20 plugin\_categories** `std::vector<std::string>` `PluginLoader::mhaplugloader_t::plugin_categories` [protected]

**5.400.3.21 b\_check\_version** `bool` `PluginLoader::mhaplugloader_t::b_check_version` [protected]

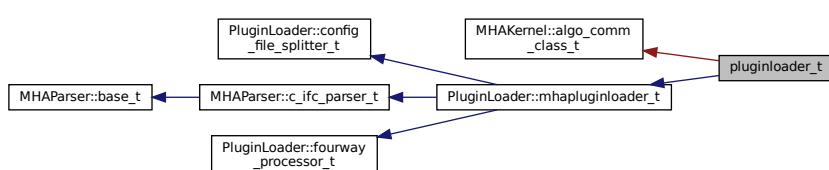
**5.400.3.22 b\_is\_prepared** `bool` `PluginLoader::mhaplugloader_t::b_is_prepared` [protected]

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

- `mhaplugloader.h`
- `mhaplugloader.cpp`

## 5.401 pluginloader\_t Class Reference

Inheritance diagram for pluginloader\_t:



## Public Member Functions

- **pluginloader\_t** (const std::string &name)
- **~pluginloader\_t** () throw ()

## Additional Inherited Members

### 5.401.1 Constructor & Destructor Documentation

**5.401.1.1 `pluginloader_t()`** pluginloader\_t::pluginloader\_t (const std::string & *name*)

**5.401.1.2 `~pluginloader_t()`** pluginloader\_t::~pluginloader\_t () throw ()

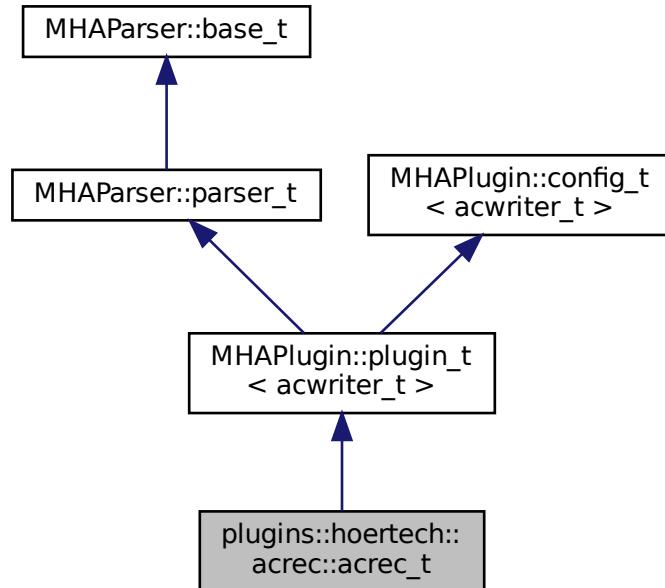
The documentation for this class was generated from the following files:

- **pluginbrowser.h**
- **pluginbrowser.cpp**

## 5.402 **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 ( algo_comm_t iac, const std::string &configured_name)`  
*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

### 5.402.1 Detailed Description

Plugin interface class of plugin acrec.

### 5.402.2 Constructor & Destructor Documentation

```
5.402.2.1 acrec_t() acrec_t::acrec_t (
    algo_comm_t iac,
    const std::string & configured_name )
```

Plugin interface constructor.

#### Parameters

<i>iac</i>	Algorithm communication variable space.
------------	---

### 5.402.3 Member Function Documentation

```
5.402.3.1 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**

<b>s</b>	input signal. The audio signal is not used or modified.
----------	---

**5.402.3.2 `prepare()`** `void acrec_t::prepare ( mhaconfig_t & cf ) [virtual]`

Prepare callback.

acrec does not modify the signal parameters.

**Parameters**

<b>cf</b>	The signal parameters.
-----------	------------------------

Implements **MHAPlugin::plugin\_t< acwriter\_t >** (p. 1149).

**5.402.3.3 `release()`** `void acrec_t::release ( ) [virtual]`

Ensure recorded data is flushed to disk.

Reimplemented from **MHAPlugin::plugin\_t< acwriter\_t >** (p. 1150).

**5.402.3.4 `start_new_session()`** `void acrec_t::start_new_session ( ) [private]`

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

#### 5.402.4 Member Data Documentation

**5.402.4.1 record** `MHAParser::bool_t` `plugins::hoertech::acrec::acrec_t::record` [private]

**5.402.4.2 fifolen** `MHAParser::int_t` `plugins::hoertech::acrec::acrec_t::fifolen` [private]

**5.402.4.3 minwrite** `MHAParser::int_t` `plugins::hoertech::acrec::acrec_t::minwrite` [private]

**5.402.4.4 prefix** `MHAParser::string_t` `plugins::hoertech::acrec::acrec_t::prefix` [private]

**5.402.4.5 varname** `MHAParser::string_t` `plugins::hoertech::acrec::acrec_t::varname` [private]

**5.402.4.6 use\_date** `MHAParser::bool_t` `plugins::hoertech::acrec::acrec_t::use_date` [private]

**5.402.4.7 patchbay** `MHAEVENTS::patchbay_t< acrec_t>` `plugins::hoertech::acrec<::acrec_t::patchbay` [private]

#### 5.402.4.8 **cv** `comm_var_t` `plugins::hoertech::acrec::acrec_t::cv` [private]

#### 5.402.4.9 **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`

### 5.403 `plugins::hoertech::acrec::acwriter_t` Class Reference

`acwriter_t` (p. 1378) 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**  
*Ouput file.*
- unsigned **num\_channels** = 0U  
*Number of channels of AC variable using stride.*
- bool **is\_num\_channels\_known** = false  
*The number of channels is determined during the first process callback.*
- bool **is\_complex** = false  
*If the AC variable is of complex valued type or not.*
- const std::string **varname**  
*The name of the ac variable to publish.*

### 5.403.1 Detailed Description

**acwriter\_t** (p. 1378) decouples signal processing from writing to disk.

Data arriving in numeric AC variables is converted to data type double, placed into a fifo pipeline to transport the data from the signal processing thread to the disk writing thread, and finally written to disk in chunks of at least minwrite numbers. All numbers are written to disk as binary doubles (8 bytes) in host byte order.

### 5.403.2 Member Typedef Documentation

#### 5.403.2.1 **output\_type** `typedef double plugins::hoertech::acrec::acwriter_t::output_type`

The numeric data type used for outputting the data to disk.

### 5.403.3 Constructor & Destructor Documentation

```
5.403.3.1 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

<i>active</i>	Only write data to disk when this is true.
<i>fifosize</i>	Capacity of both the fifo pipeline and of the disk buffer.
<i>minwrite</i>	Wait for a fifo fill count of at least minwrite doubles before flushing the contents of the fifo to disk. Fifo is also flushed before this object is destroyed.
<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.
<i>varname</i>	Name of AC variable to save into file. Can be accessed through getter method <b>get_varname()</b> (p. 1381). Stored here to avoid races between processing thread and configuration thread.

**5.403.3.2 ~acwriter\_t()** plugins::hoertech::acrec::acwriter\_t::~acwriter\_t ( ) [default]

Deallocates memory but does not terminate the write\_thread.

write\_thread must be terminated before the destructor executes by calling exit\_request.

### 5.403.4 Member Function Documentation

**5.403.4.1 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.

**5.403.4.2 exit\_request()** void acwriter\_t::exit\_request ( )

Terminate output thread.

**5.403.4.3 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

**5.403.4.4 write\_thread()** void acwriter\_t::write\_thread ( ) [private]

Main method of the disk writer thread.

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

**5.403.4.5 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.

## 5.403.5 Member Data Documentation

**5.403.5.1 `close_session`** std::atomic<bool> plugins::hoertech::acrec::acwriter\_t::close\_session [private]

cross-thread-synchronization.

`write_thread()` (p. 1381) terminates after this is set to true by `exit_request()` (p. 1381).

**5.403.5.2 `active`** const bool plugins::hoertech::acrec::acwriter\_t::active [private]

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

**5.403.5.3 `fifo`** std::unique\_ptr< mha\_fifo\_lf\_t< output\_type> > plugins::hoertech::acrec::acwriter\_t::fifo [private]

Fifo for decoupling signal processing thread from disk writer thread.

**5.403.5.4 `disk_write_threshold_min_num_samples`** const unsigned int plugins::hoertech::acrec::acwriter\_t::disk\_write\_threshold\_min\_num\_samples [private]

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

**5.403.5.5 writethread** std::thread plugins::hoertech::acrec::acwriter\_t::writethread [private]

The thread that writes to disk.

**5.403.5.6 diskbuffer** std::unique\_ptr< **output\_type** []> plugins::hoertech::acrec::acwriter\_t::diskbuffer [private]

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

**5.403.5.7 outfile** std::fstream plugins::hoertech::acrec::acwriter\_t::outfile [private]

Output file.

**5.403.5.8 num\_channels** unsigned plugins::hoertech::acrec::acwriter\_t::num\_channels = 0U [private]

Number of channels of AC variable using stride.

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

**5.403.5.9 is\_num\_channels\_known** bool plugins::hoertech::acrec::acwriter\_t::is\_num\_channels\_known = false [private]

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

is\_num\_channels\_known is set to true after the first process callback.

**5.403.5.10 is\_complex** bool plugins::hoertech::acrec::acwriter\_t::is\_complex = false [private]

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

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

**5.403.5.11 varname** const std::string plugins::hoertech::acrec::acwriter\_t::varname  
[private]

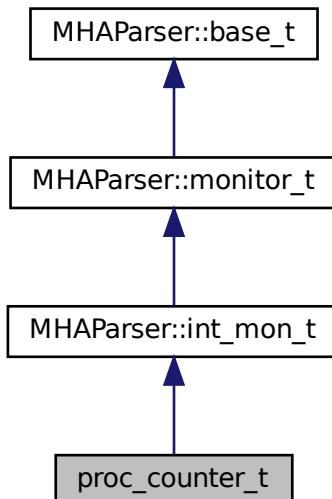
The name of the ac variable to publish.

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

- **acrec.hh**
- **acrec.cpp**

## 5.404 proc\_counter\_t Class Reference

Inheritance diagram for proc\_counter\_t:



### Public Member Functions

- **proc\_counter\_t ( algo\_comm\_t iac, const std::string &configured\_name)**
- **~proc\_counter\_t ()**
- **mha\_wave\_t \* process ( mha\_wave\_t \*s)**
- **mha\_spec\_t \* process ( mha\_spec\_t \*s)**
- **void prepare\_ ( mhaconfig\_t &)**
- **void release\_ ()**

### Private Attributes

- algo\_comm\_t ac

### Additional Inherited Members

#### 5.404.1 Constructor & Destructor Documentation

**5.404.1.1 proc\_counter\_t()** proc\_counter\_t::proc\_counter\_t (algo\_comm\_t iac, const std::string & configured\_name )

**5.404.1.2 ~proc\_counter\_t()** proc\_counter\_t::~proc\_counter\_t ( )

#### 5.404.2 Member Function Documentation

**5.404.2.1 process() [1/2]** mha\_wave\_t \* proc\_counter\_t::process (mha\_wave\_t \* s )

**5.404.2.2 process() [2/2]** mha\_spec\_t \* proc\_counter\_t::process (mha\_spec\_t \* s )

**5.404.2.3 prepare\_()** void proc\_counter\_t::prepare\_ (mhaconfig\_t & ) [inline]

**5.404.2.4 `release_()`** `void proc_counter_t::release_() [inline]`

### 5.404.3 Member Data Documentation

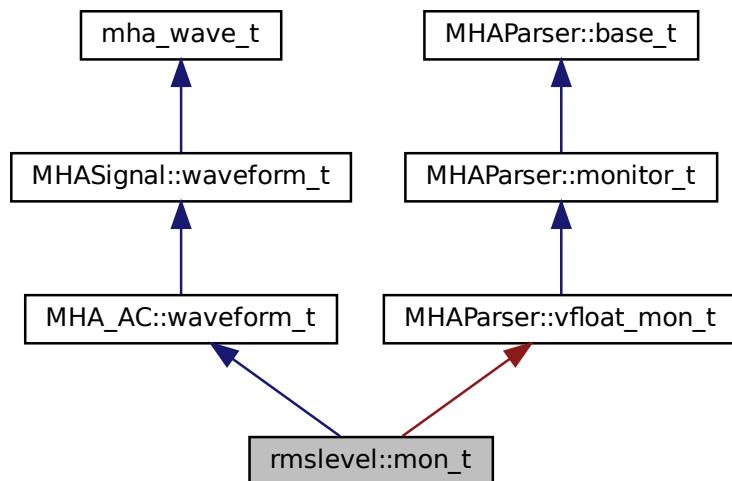
**5.404.3.1 `ac algo_comm_t`** `proc_counter_t::ac [private]`

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

- `proc_counter.cpp`

### 5.405 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, MHAParser::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

### 5.405.1 Constructor & Destructor Documentation

#### 5.405.1.1 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 )
```

#### 5.405.1.2 mon\_t() [2/3] rmslevel::mon\_t::mon\_t (

```
    const mon_t & ) [delete]
```

#### 5.405.1.3 mon\_t() [3/3] rmslevel::mon\_t::mon\_t (

```
    mon_t && ) [delete]
```

### 5.405.2 Member Function Documentation

#### 5.405.2.1 operator=() [1/2] mon\_t& rmslevel::mon\_t::operator= (

```
    const mon_t & ) [delete]
```

#### 5.405.2.2 operator=() [2/2] mon\_t& rmslevel::mon\_t::operator= (

```
    mon_t && ) [delete]
```

### 5.405.2.3 `store()` `void rmslevel::mon_t::store( )`

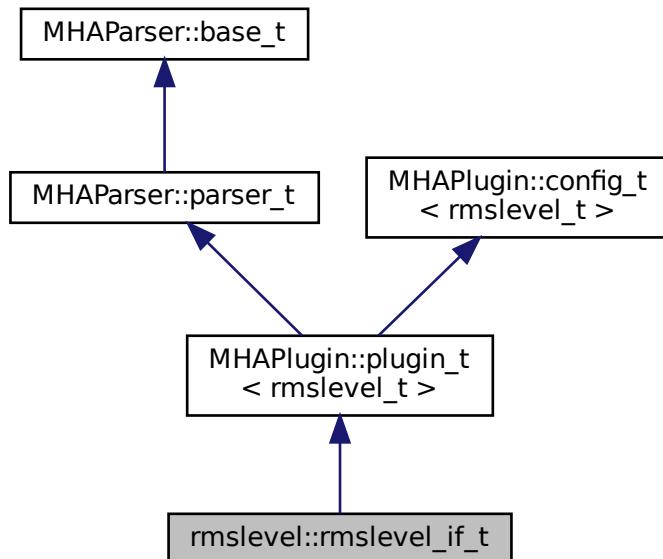
The documentation for this class was generated from the following file:

- `rmslevel.cpp`

## 5.406 `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( algo_comm_t iac, const std::string &configured_name)`
- `mha_spec_t * process( mha_spec_t *)`
- `mha_wave_t * process( mha_wave_t *)`
- `void prepare( mhaconfig_t &)`

### Private Member Functions

- `void update()`

## Private Attributes

- **MHAEvents::patchbay\_t**< rmslevel\_if\_t > patchbay
- std::string name
- **MHAParser::kw\_t** unit

## Additional Inherited Members

### 5.406.1 Detailed Description

Interface class of the rmslevel plugin.

### 5.406.2 Constructor & Destructor Documentation

```
5.406.2.1 rmslevel_if_t() rmslevel::rmslevel_if_t::rmslevel_if_t (
    algo_comm_t iac,
    const std::string & configured_name )
```

### 5.406.3 Member Function Documentation

```
5.406.3.1 process() [1/2] mha_spec_t * rmslevel::rmslevel_if_t::process (
    mha_spec_t * s )
```

```
5.406.3.2 process() [2/2] mha_wave_t * rmslevel::rmslevel_if_t::process (
    mha_wave_t * s )
```

```
5.406.3.3 prepare() void rmslevel::rmslevel_if_t::prepare (
    mhaconfig_t & tf ) [virtual]
```

Implements **MHAPlugin::plugin\_t**< rmslevel\_t > (p. 1149).

**5.406.3.4 update()** void rmslevel::rmslevel\_if\_t::update ( ) [private]

#### 5.406.4 Member Data Documentation

**5.406.4.1 patchbay** MHAEvents::patchbay\_t< rmslevel\_if\_t> rmslevel::rmslevel\_if\_t::patchbay [private]

**5.406.4.2 name** std::string rmslevel::rmslevel\_if\_t::name [private]

**5.406.4.3 unit** MHAParser::kw\_t rmslevel::rmslevel\_if\_t::unit [private]

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

- **rmslevel.cpp**

### 5.407 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.*

### 5.407.1 Detailed Description

Run-time configuration class of the rmslevel plugin.

### 5.407.2 Constructor & Destructor Documentation

```
5.407.2.1 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_</i> $\leftarrow$ <i>channels_</i>	Number of monitor channels if channel summation shall be used, zero otherwise

### 5.407.2.2 ~rmslevel\_t() rmslevel::rmslevel\_t::~rmslevel\_t ( ) [inline]

### 5.407.3 Member Function Documentation

**5.407.3.1 process()** [1/2] `mha_spec_t * rmslevel::rmslevel_t::process (`  
`mha_spec_t * s )`

**5.407.3.2 process()** [2/2] `mha_wave_t * rmslevel::rmslevel_t::process (`  
`mha_wave_t * s )`

**5.407.3.3 fill()** [1/2] `void rmslevel::rmslevel_t::fill (`  
`mha_spec_t * )`

**5.407.3.4 sum\_and\_fill()** [1/2] `void rmslevel::rmslevel_t::sum_and_fill (`  
`mha_spec_t * )`

**5.407.3.5 fill()** [2/2] `void rmslevel::rmslevel_t::fill (`  
`mha_wave_t * )`

**5.407.3.6 sum\_and\_fill()** [2/2] `void rmslevel::rmslevel_t::sum_and_fill (`  
`mha_wave_t * )`

**5.407.3.7 insert()** `void rmslevel::rmslevel_t::insert ( )`

#### 5.407.4 Member Data Documentation

**5.407.4.1 monitors** std::unordered\_map<std::string, std::unique\_ptr< **mon\_t**> > rmslevel::rmslevel\_t::monitors [private]

**5.407.4.2 fftlen** unsigned int rmslevel::rmslevel\_t::fftlens [private]

**5.407.4.3 unit** **UNIT** rmslevel::rmslevel\_t::unit [private]

**5.407.4.4 domain** **mha\_domain\_t** rmslevel::rmslevel\_t::domain [private]

**5.407.4.5 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**

## 5.408 rohBeam::configOptions Struct Reference

### Public Attributes

- bool **enable\_adaptive\_beam**
- int **binaural\_type\_index**
- float **alpha\_postfilter**
- float **alpha\_blocking\_XkXi**
- float **alpha\_blocking\_XkY**

## 5.408.1 Member Data Documentation

**5.408.1.1 enable\_adaptive\_beam** bool rohBeam::configOptions::enable\_adaptive\_beam

**5.408.1.2 binaural\_type\_index** int rohBeam::configOptions::binaural\_type\_index

**5.408.1.3 alpha\_postfilter** float rohBeam::configOptions::alpha\_postfilter

**5.408.1.4 alpha\_blocking\_XkXi** float rohBeam::configOptions::alpha\_blocking\_XkXi

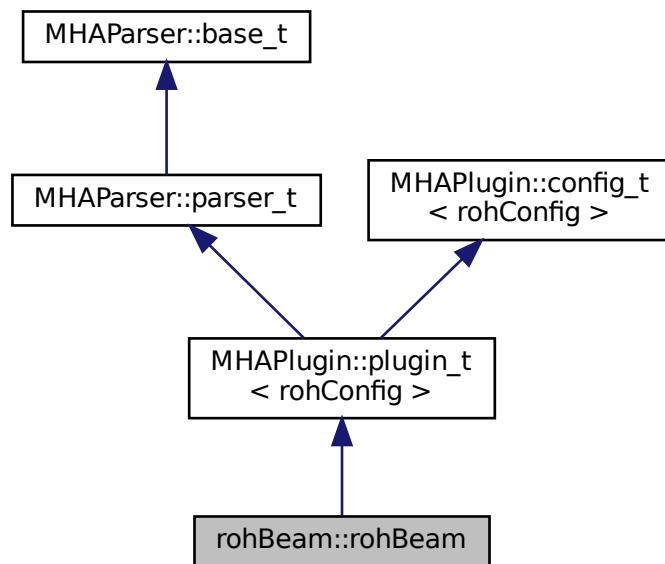
**5.408.1.5 alpha\_blocking\_XkY** float rohBeam::configOptions::alpha\_blocking\_XkY

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

- **rohBeam.hh**

## 5.409 rohBeam::rohBeam Class Reference

Inheritance diagram for rohBeam::rohBeam:



### Public Member Functions

- `rohBeam ( algo_comm_t iac, const std::string &configured_name)`
- `~rohBeam ()`
- `mha_spec_t * process ( mha_spec_t *)`
- `void prepare ( mhaconfig_t &)`
- `void release (void)`

### Private 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

### 5.409.1 Constructor & Destructor Documentation

**5.409.1.1 rohBeam()** rohBeam::rohBeam::rohBeam (

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

**5.409.1.2 ~rohBeam()** rohBeam::rohBeam::~rohBeam ( )

## 5.409.2 Member Function Documentation

**5.409.2.1 process()** mha\_spec\_t\* rohBeam::rohBeam::process ( mha\_spec\_t \* )

**5.409.2.2 prepare()** void rohBeam::rohBeam::prepare ( mhaconfig\_t & ) [virtual]

Implements **MHAPlugIn::plugin\_t< rohConfig >** (p. 1149).

**5.409.2.3 release()** void rohBeam::rohBeam::release ( void ) [inline], [virtual]

Reimplemented from **MHAPlugIn::plugin\_t< rohConfig >** (p. 1150).

**5.409.2.4 update\_cfg()** void rohBeam::rohBeam::update\_cfg ( ) [private]

**5.409.2.5 compute\_head\_model\_T()** float rohBeam::rohBeam::compute\_head\_model\_T ( float ) [private]

**5.409.2.6 compute\_head\_model\_alpha()** float rohBeam::rohBeam::compute\_head\_model\_alpha ( float ) [private]

**5.409.2.7 compute\_head\_model\_mat()** Eigen::MatrixXcf\* rohBeam::rohBeam::compute←  
\_head\_model\_mat ( float src\_az\_degrees ) [private]

**5.409.2.8 compute\_delaycomp\_vec()** MHASignal::matrix\_t\* rohBeam::rohBeam::compute←  
\_delaycomp\_vec ( Eigen::MatrixXcf \* headModel ) [private]

**5.409.2.9 noise\_integrate\_hrtf()** std::vector<Eigen::MatrixXcf>\* rohBeam::rohBeam←  
::noise\_integrate\_hrtf ( ) [private]

**5.409.2.10 solve\_MVDR()** Eigen::VectorXcf rohBeam::rohBeam::solve\_MVDR ( Eigen::VectorXcf propVec,  
Eigen::MatrixXcf noiseM ) [private]

**5.409.2.11 compute\_uncorr()** const Eigen::MatrixXf rohBeam::rohBeam::compute←  
uncorr ( float w ) [private]

**5.409.2.12 compute\_diff2D()** const Eigen::MatrixXf rohBeam::rohBeam::compute\_diff2D  
( float ) [private]

**5.409.2.13 compute\_diff3D()** const Eigen::MatrixXf rohBeam::rohBeam::compute\_diff3D  
( float ) [private]

```
5.409.2.14 compute_beamW()  MHASignal::matrix_t* rohBeam::rohBeam::compute_beamW
(
    Eigen::MatrixXcf * ) [private]
```

```
5.409.2.15 compute_wng()  float rohBeam::rohBeam::compute_wng (
    Eigen::VectorXcf freqRes,
    Eigen::VectorXcf propVec ) [private]
```

```
5.409.2.16 export_beam_design()  void rohBeam::rohBeam::export_beam_design (
    const MHASignal::matrix_t & beamW,
    const Eigen::MatrixXcf & headModel ) [private]
```

```
5.409.2.17 get_noise_model_func()  noiseFuncPtr rohBeam::rohBeam::get_noise_←
model_func (
    void ) [private]
```

```
5.409.2.18 on_model_param_valuechanged()  void rohBeam::rohBeam::on_model_←
param_valuechanged ( ) [private]
```

### 5.409.3 Member Data Documentation

```
5.409.3.1 noiseFuncPtr  const typedef Eigen::MatrixXf(rohBeam::* rohBeam::rohBeam←
::noiseFuncPtr) (float) [private]
```

```
5.409.3.2 prop_type  MHAParser::kw_t rohBeam::rohBeam::prop_type [private]
```

**5.409.3.3 sampled\_hrir\_path** `MHAParser::string_t` `rohBeam::rohBeam::sampled_hrir_` ↵  
path [private]

**5.409.3.4 source\_azimuth\_degrees** `MHAParser::float_t` `rohBeam::rohBeam::source_` ↵  
azimuth\_degrees [private]

**5.409.3.5 mic\_azimuth\_degrees\_vec** `MHAParser::vfloat_t` `rohBeam::rohBeam::mic_` ↵  
azimuth\_degrees\_vec [private]

**5.409.3.6 head\_model\_sphere\_radius\_cm** `MHAParser::float_t` `rohBeam::rohBeam::head_model_sphere_radius_cm` [private]

**5.409.3.7 intermic\_distance\_cm** `MHAParser::mfloat_t` `rohBeam::rohBeam::intermic_` ↵  
distance\_cm [private]

**5.409.3.8 noise\_field\_model** `MHAParser::kw_t` `rohBeam::rohBeam::noise_field_model`  
[private]

**5.409.3.9 enable\_adaptive\_beam** `MHAParser::bool_t` `rohBeam::rohBeam::enable_` ↵  
adaptive\_beam [private]

**5.409.3.10 binaural\_type** `MHAParser::kw_t` `rohBeam::rohBeam::binaural_type` [private]

**5.409.3.11 diag\_loading\_mu** `MHAParser::float_t` `rohBeam::rohBeam::diag_loading_mu`  
[private]

**5.409.3.12 enable\_export** `MHAParser::bool_t` `rohBeam::rohBeam::enable_export` [private]

**5.409.3.13 enable\_wng\_optimization** `MHAParser::bool_t` `rohBeam::rohBeam::enable_←`  
`wng_optimization` [private]

**5.409.3.14 tau\_postfilter\_ms** `MHAParser::float_t` `rohBeam::rohBeam::tau_postfilter←`  
`_ms` [private]

**5.409.3.15 tau\_blocking\_XkXi\_ms** `MHAParser::float_t` `rohBeam::rohBeam::tau_←`  
`blocking_XkXi_ms` [private]

**5.409.3.16 tau\_blocking\_XkY\_ms** `MHAParser::float_t` `rohBeam::rohBeam::tau_blocking←`  
`_XkY_ms` [private]

**5.409.3.17 patchbay** `MHAEvents::patchbay_t< rohBeam>` `rohBeam::rohBeam::patchbay`  
[private]

**5.409.3.18 prepared** `bool` `rohBeam::rohBeam::prepared` [private]

**5.409.3.19 beamExport** `MHA_AC::spectrum_t*` `rohBeam::rohBeam::beamExport` [private]

**5.409.3.20 noiseModelExport** `MHA_AC::waveform_t*` `rohBeam::rohBeam::noiseModelExport` [private]

**5.409.3.21 propExport** `MHA_AC::spectrum_t*` `rohBeam::rohBeam::propExport` [private]

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

- **rohBeam.hh**

## 5.410 rohBeam::rohConfig Class Reference

### Public Member Functions

- **rohConfig** (const `mhaconfig_t in_cfg`, const `mhaconfig_t out_cfg`, std::unique\_ptr< Eigen::MatrixXcf > headModel\_, std::unique\_ptr< `MHASignal::matrix_t` > beamW\_, std::unique\_ptr< `MHASignal::matrix_t` > delayComp\_, const `configOptions` &options)
- **rohConfig** ( `rohConfig` \*lastConfig, const `mhaconfig_t`, const `mhaconfig_t out_cfg`, std::unique\_ptr< Eigen::MatrixXcf > headModel\_, std::unique\_ptr< `MHASignal::matrix_t` > beamW\_, std::unique\_ptr< `MHASignal::matrix_t` > delayComp\_, const `configOptions` &options)
- **~rohConfig** ()
- **rohConfig** (const `rohConfig` &)=delete
- **rohConfig** & **operator=** (const `rohConfig` &)=delete
- **mha\_spec\_t \* process** ( `mha_spec_t` \*)
- **void init\_dynamic** ()

### Private Member Functions

- **void phasereconstruction** ( `MHASignal::spectrum_t` \*)
- **void postfilter** ( `mha_spec_t` \*, `MHASignal::spectrum_t` \*)
- **void copyfixedbfoutput** ( `MHASignal::spectrum_t` \*)

## Private Attributes

- int **nfreq**
- int **nchan\_block**
- **mhaconfig\_t in\_cfg**
- **mhaconfig\_t out\_cfg**
- bool **enable\_adaptive\_beam**
- int **binaural\_type\_index**
- std::unique\_ptr< Eigen::MatrixXcf > **headModel**
- std::unique\_ptr< **MHASignal::matrix\_t** > **beamW**
- std::unique\_ptr< **MHASignal::matrix\_t** > **delayComp**
- **MHASignal::spectrum\_t \* beam1**
- **MHASignal::spectrum\_t \* beamA**
- **MHASignal::spectrum\_t \* blockSpec**
- **MHASignal::spectrum\_t \* outSpec**
- float **alpha\_postfilter**
- float **alpha\_blocking\_XkXi**
- float **alpha\_blocking\_XkY**
- std::vector< Eigen::MatrixXcf > **corrXpXp**
- std::vector< Eigen::VectorXcf > **corrXpYf**
- Eigen::VectorXf **corrZZ**
- Eigen::VectorXf **corrLL**
- Eigen::VectorXf **corrRR**
- Eigen::HouseholderQR< Eigen::MatrixXcf > **hhCorrXpXp**
- Eigen::VectorXcf **nextXpYf**
- Eigen::VectorXcf **blockXp**
- Eigen::VectorXcf **freqResp**
- Eigen::ArrayXf **magResp**
- float **minLim**
- float **maxLim**

### 5.410.1 Constructor & Destructor Documentation

```
5.410.1.1 rohConfig() [1/3] rohBeam::rohConfig::rohConfig (
    const mhaconfig_t in_cfg,
    const mhaconfig_t out_cfg,
    std::unique_ptr< Eigen::MatrixXcf > headModel_,
    std::unique_ptr< MHASignal::matrix_t > beamW_,
    std::unique_ptr< MHASignal::matrix_t > delayComp_,
    const configOptions & options )
```

---

**5.410.1.2 `rohConfig()` [2/3]** `rohBeam::rohConfig::rohConfig (`

```
    rohConfig * lastConfig,
    const mhaconfig_t,
    const mhaconfig_t out_cfg,
    std::unique_ptr< Eigen::MatrixXcf > headModel_,
    std::unique_ptr< MHASignal::matrix_t > beamW_,
    std::unique_ptr< MHASignal::matrix_t > delayComp_,
    const configOptions & options )
```

**5.410.1.3 `~rohConfig()`** `rohBeam::rohConfig::~rohConfig ( )`

**5.410.1.4 `rohConfig()` [3/3]** `rohBeam::rohConfig::rohConfig (`

```
    const rohConfig & ) [delete]
```

## 5.410.2 Member Function Documentation

**5.410.2.1 `operator=()`** `rohConfig& rohBeam::rohConfig::operator= (`

```
    const rohConfig & ) [delete]
```

**5.410.2.2 `process()`** `mha_spec_t * rohBeam::rohConfig::process (`

```
    mha_spec_t * inSpec )
```

**5.410.2.3 `init_dynamic()`** `void rohBeam::rohConfig::init_dynamic ( )`

**5.410.2.4 `phasereconstruction()`** `void rohBeam::rohConfig::phasereconstruction (`

```
    MHASignal::spectrum_t * prevSpecPost ) [private]
```

**5.410.2.5 postfilter()** void rohBeam::rohConfig::postfilter (

```
mha_spec_t * inSpec,
MHASignal::spectrum_t * prevSpecPost ) [private]
```

**5.410.2.6 copyfixedbfoutput()** void rohBeam::rohConfig::copyfixedbfoutput (

```
MHASignal::spectrum_t * prevSpecPost ) [private]
```

### 5.410.3 Member Data Documentation

**5.410.3.1 nfreq** int rohBeam::rohConfig::nfreq [private]

**5.410.3.2 nchan\_block** int rohBeam::rohConfig::nchan\_block [private]

**5.410.3.3 in\_cfg** mhaconfig\_t rohBeam::rohConfig::in\_cfg [private]

**5.410.3.4 out\_cfg** mhaconfig\_t rohBeam::rohConfig::out\_cfg [private]

**5.410.3.5 enable\_adaptive\_beam** bool rohBeam::rohConfig::enable\_adaptive\_beam [private]

**5.410.3.6 binaural\_type\_index** int rohBeam::rohConfig::binaural\_type\_index [private]

**5.410.3.7 headModel** std::unique\_ptr<Eigen::MatrixXcf> rohBeam::rohConfig::headModel [private]

**5.410.3.8 beamW** std::unique\_ptr< MHASignal::matrix\_t > rohBeam::rohConfig::beamW [private]

**5.410.3.9 delayComp** std::unique\_ptr< MHASignal::matrix\_t > rohBeam::rohConfig::delayComp [private]

**5.410.3.10 beam1** MHASignal::spectrum\_t\* rohBeam::rohConfig::beam1 [private]

**5.410.3.11 beamA** MHASignal::spectrum\_t\* rohBeam::rohConfig::beamA [private]

**5.410.3.12 blockSpec** MHASignal::spectrum\_t\* rohBeam::rohConfig::blockSpec [private]

**5.410.3.13 outSpec** MHASignal::spectrum\_t\* rohBeam::rohConfig::outSpec [private]

**5.410.3.14 alpha\_postfilter** float rohBeam::rohConfig::alpha\_postfilter [private]

**5.410.3.15 alpha\_blocking\_XkXi** float rohBeam::rohConfig::alpha\_blocking\_XkXi [private]

**5.410.3.16 alpha\_blocking\_XkY** float rohBeam::rohConfig::alpha\_blocking\_XkY [private]

**5.410.3.17 corrXpXp** std::vector<Eigen::MatrixXcf> rohBeam::rohConfig::corrXpXp [private]

**5.410.3.18 corrXpYf** std::vector<Eigen::VectorXcf> rohBeam::rohConfig::corrXpYf [private]

**5.410.3.19 corrZZ** Eigen::VectorXf rohBeam::rohConfig::corrZZ [private]

**5.410.3.20 corrLL** Eigen::VectorXf rohBeam::rohConfig::corrLL [private]

**5.410.3.21 corrRR** Eigen::VectorXf rohBeam::rohConfig::corrRR [private]

**5.410.3.22 hhCorrXpXp** Eigen::HouseholderQR<Eigen::MatrixXcf> rohBeam::rohConfig::hhCorrXpXp [private]

**5.410.3.23 nextXpYf** Eigen::VectorXcf rohBeam::rohConfig::nextXpYf [private]

**5.410.3.24 blockXp** Eigen::VectorXcf rohBeam::rohConfig::blockXp [private]

**5.410.3.25 freqResp** Eigen::VectorXcf rohBeam::rohConfig::freqResp [private]

**5.410.3.26 magResp** Eigen::ArrayXf rohBeam::rohConfig::magResp [private]

**5.410.3.27 minLim** float rohBeam::rohConfig::minLim [private]

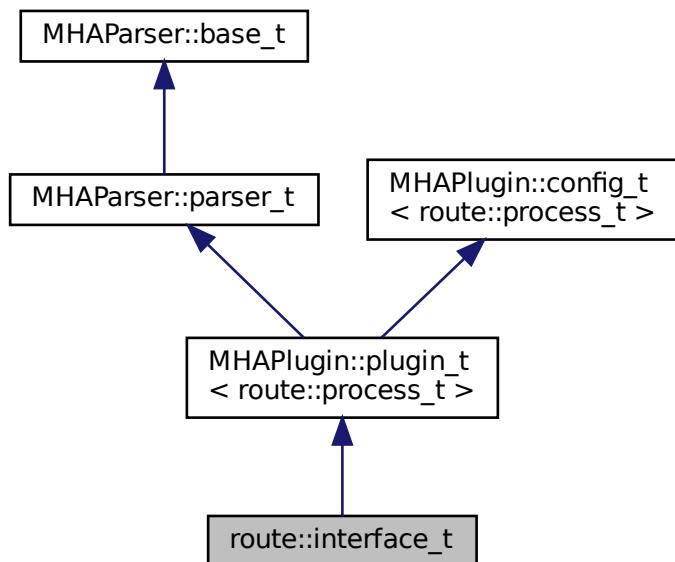
**5.410.3.28 maxLim** float rohBeam::rohConfig::maxLim [private]

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

- **rohBeam.hh**
- **rohBeam.cpp**

## 5.411 route::interface\_t Class Reference

Inheritance diagram for route::interface\_t:



## Public Member Functions

- `interface_t ( algo_comm_t iac, const std::string &configured_name)`
- `void prepare ( mhaconfig_t &)`
- `void release ()`
- `mha_wave_t * process ( mha_wave_t *)`
- `mha_spec_t * process ( mha_spec_t *)`

## Private Member Functions

- `void update ()`

## Private Attributes

- `MHAEvents::patchbay_t< route::interface_t > patchbay`
- `MHAParser::vstring_t route_out`
- `MHAParser::vstring_t route_ac`
- `mhaconfig_t cfin`
- `mhaconfig_t cfout`
- `mhaconfig_t cfac`
- `bool prepared`
- `bool stopped`
- `std::string algo`

## Additional Inherited Members

### 5.411.1 Constructor & Destructor Documentation

**5.411.1.1 interface\_t()** `route::interface_t::interface_t (`  
`algo_comm_t iac,`  
`const std::string & configured_name )`

### 5.411.2 Member Function Documentation

**5.411.2.1 `prepare()`** `void route::interface_t::prepare (`  
`mhaconfig_t & cf ) [virtual]`

Implements `MHAPlugIn::plugin_t< route::process_t >` (p. 1149).

**5.411.2.2 `release()`** `void route::interface_t::release ( ) [virtual]`

Reimplemented from `MHAPlugIn::plugin_t< route::process_t >` (p. 1150).

**5.411.2.3 `process()` [1/2]** `mha_wave_t * route::interface_t::process (`  
`mha_wave_t * s )`

**5.411.2.4 `process()` [2/2]** `mha_spec_t * route::interface_t::process (`  
`mha_spec_t * s )`

**5.411.2.5 `update()`** `void route::interface_t::update ( ) [private]`

### 5.411.3 Member Data Documentation

**5.411.3.1 `patchbay`** `MHAEvents::patchbay_t< route::interface_t > route::interface->-t::patchbay [private]`

**5.411.3.2 `route_out`** `MHAParser::vstring_t route::interface_t::route_out [private]`

**5.411.3.3 route\_ac** `MHAParser::vstring_t route::interface_t::route_ac [private]`

**5.411.3.4 cfin** `mhaconfig_t route::interface_t::cfin [private]`

**5.411.3.5 cfout** `mhaconfig_t route::interface_t::cfout [private]`

**5.411.3.6 cfac** `mhaconfig_t route::interface_t::cfac [private]`

**5.411.3.7 prepared** `bool route::interface_t::prepared [private]`

**5.411.3.8 stopped** `bool route::interface_t::stopped [private]`

**5.411.3.9 algo** `std::string route::interface_t::algo [private]`

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

- `route.cpp`

## 5.412 route::process\_t Class Reference

### Public Member Functions

- `process_t ( algo_comm_t iac, const std::string acname, const std::vector< std::string > &r_out, const std::vector< std::string > &r_ac, const mhaconfig_t &cf_in, const mhaconfig_t &cf_out, const mhaconfig_t &cf_ac, bool sync)`
- `mha_wave_t * process ( mha_wave_t *)`
- `mha_spec_t * process ( mha_spec_t *)`

## Private Attributes

- **MHAMultiSrc::waveform\_t wout**
- **MHAMultiSrc::spectrum\_t sout**
- **MHAMultiSrc::waveform\_t wout\_ac**
- **MHAMultiSrc::spectrum\_t sout\_ac**

### 5.412.1 Constructor & Destructor Documentation

**5.412.1.1 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 )
```

### 5.412.2 Member Function Documentation

**5.412.2.1 process() [1/2]** mha\_wave\_t \* route::process\_t::process (

```
mha_wave_t * s )
```

**5.412.2.2 process() [2/2]** mha\_spec\_t \* route::process\_t::process (

```
mha_spec_t * s )
```

### 5.412.3 Member Data Documentation

**5.412.3.1 wout** `MHAMultiSrc::waveform_t` `route::process_t::wout` [private]

**5.412.3.2 sout** `MHAMultiSrc::spectrum_t` `route::process_t::sout` [private]

**5.412.3.3 wout\_ac** `MHAMultiSrc::waveform_t` `route::process_t::wout_ac` [private]

**5.412.3.4 sout\_ac** `MHAMultiSrc::spectrum_t` `route::process_t::sout_ac` [private]

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

- `route.cpp`

## 5.413 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**  
*Input signal cache.*
- **MHASignal::waveform\_t Uflt**  
*Input signal cache (second filter)*
- **MHASignal::waveform\_t Pu**  
*Power of input signal delayline.*
- **MHASignal::waveform\_t fu**  
*Filtered input signal.*
- **MHASignal::waveform\_t fuflt**  
*Filtered input signal.*
- **MHASignal::waveform\_t fu\_previous**
- **MHASignal::waveform\_t y\_previous**
- **MHASignal::waveform\_t P\_Sum**
- std::string **name\_u\_**
- std::string **name\_d\_**
- std::string **name\_e\_**
- int **n\_no\_update\_**
- int **no\_iter**
- **mha\_wave\_t s\_E**

### 5.413.1 Constructor & Destructor Documentation

```
5.413.1.1 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 )
```

**5.413.1.2 ~rt\_nlms\_t()** rt\_nlms\_t::~rt\_nlms\_t ( ) [inline]

## 5.413.2 Member Function Documentation

**5.413.2.1 process()** mha\_wave\_t \* rt\_nlms\_t::process (

```
mha_wave_t * sUD,
mha_real_t rho,
mha_real_t c,
unsigned int norm_type,
unsigned int estim_type,
mha_real_t lambda_smooth )
```

**5.413.2.2 insert()** void rt\_nlms\_t::insert ( )

## 5.413.3 Member Data Documentation

**5.413.3.1 ac** algo\_comm\_t rt\_nlms\_t::ac [private]

**5.413.3.2 ntaps** unsigned int rt\_nlms\_t::ntaps [private]

**5.413.3.3 frames** unsigned int rt\_nlms\_t::frames [private]

**5.413.3.4 channels** unsigned int rt\_nlms\_t::channels [private]

**5.413.3.5 F** `MHA_AC::waveform_t rt_nlms_t::F` [private]

**5.413.3.6 U** `MHASignal::waveform_t rt_nlms_t::U` [private]

Input signal cache.

**5.413.3.7 Uflt** `MHASignal::waveform_t rt_nlms_t::Uflt` [private]

Input signal cache (second filter)

**5.413.3.8 Pu** `MHASignal::waveform_t rt_nlms_t::Pu` [private]

Power of input signal delayline.

**5.413.3.9 fu** `MHASignal::waveform_t rt_nlms_t::fu` [private]

Filtered input signal.

**5.413.3.10 fuflt** `MHASignal::waveform_t rt_nlms_t::fuflt` [private]

Filtered input signal.

**5.413.3.11 fu\_previous** `MHASignal::waveform_t rt_nlms_t::fu_previous` [private]

**5.413.3.12 y\_previous** `MHASignal::waveform_t rt_nlms_t::y_previous [private]`

**5.413.3.13 P\_Sum** `MHASignal::waveform_t rt_nlms_t::P_Sum [private]`

**5.413.3.14 name\_u\_** `std::string rt_nlms_t::name_u_ [private]`

**5.413.3.15 name\_d\_** `std::string rt_nlms_t::name_d_ [private]`

**5.413.3.16 name\_e\_** `std::string rt_nlms_t::name_e_ [private]`

**5.413.3.17 n\_no\_update\_** `int rt_nlms_t::n_no_update_ [private]`

**5.413.3.18 no\_iter** `int rt_nlms_t::no_iter [private]`

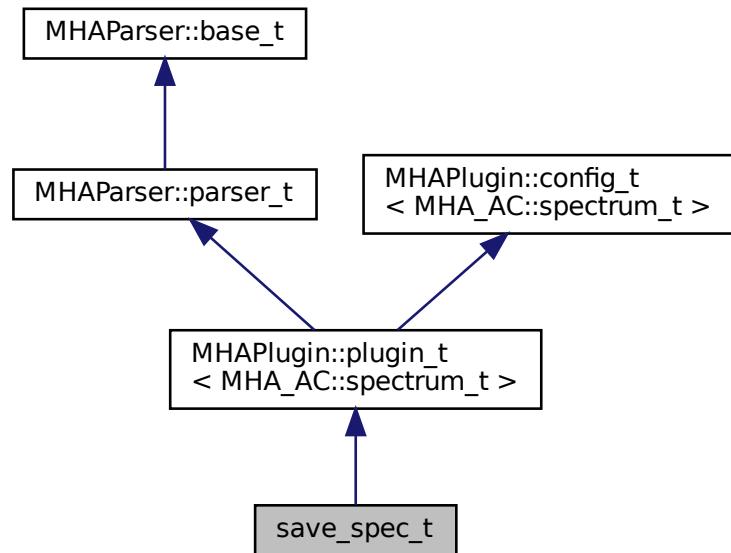
**5.413.3.19 s\_E** `mha_wave_t rt_nlms_t::s_E [private]`

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

- **nlms\_wave.cpp**

## 5.414 save\_spec\_t Class Reference

## Inheritance diagram for save\_spec\_t:



## Public Member Functions

- **save\_spec\_t** ( **algo\_comm\_t** iac, const std::string &configured\_name)
  - **mha\_spec\_t \* process** ( **mha\_spec\_t** \*s)
  - void **prepare** ( **mhaconfig\_t** &tf)

## Private Attributes

- std::string **basename**

## **Additional Inherited Members**

#### **5.414.1 Constructor & Destructor Documentation**

```
5.414.1.1 save_spec_t() save_spec_t::save_spec_t (
    algo_comm_t iac,
    const std::string & configured_name ) [inline]
```

## 5.414.2 Member Function Documentation

```
5.414.2.1 process() mha_spec_t* save_spec_t::process (
    mha_spec_t * s ) [inline]
```

```
5.414.2.2 prepare() void save_spec_t::prepare (
    mhaconfig_t & tf ) [inline], [virtual]
```

Implements **MHAPlugin::plugin\_t< MHA\_AC::spectrum\_t >** (p. 1149).

## 5.414.3 Member Data Documentation

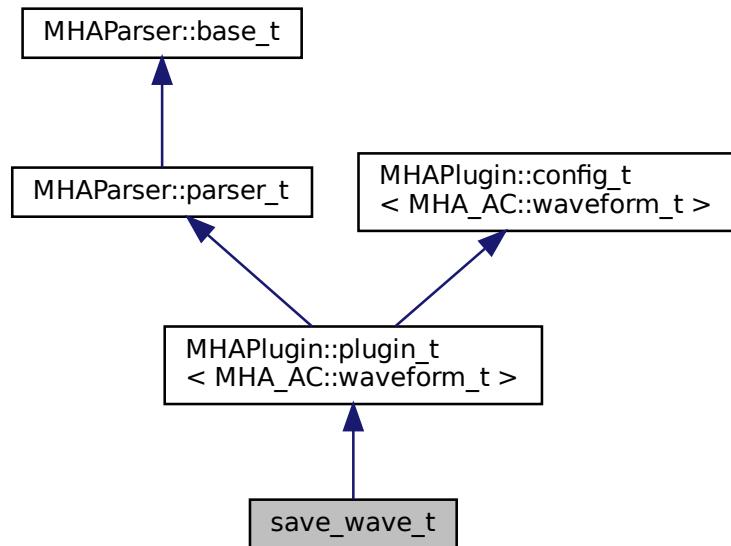
```
5.414.3.1 basename std::string save_spec_t::basename [private]
```

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

- **save\_spec.cpp**

## 5.415 save\_wave\_t Class Reference

Inheritance diagram for save\_wave\_t:



### Public Member Functions

- `save_wave_t ( algo_comm_t iac, const std::string &configured_name)`
- `mha_wave_t * process ( mha_wave_t *s)`
- `void prepare ( mhaconfig_t &tf)`

### Private Attributes

- `std::string basename`

### Additional Inherited Members

#### 5.415.1 Constructor & Destructor Documentation

```
5.415.1.1 save_wave_t() save_wave_t::save_wave_t (
    algo_comm_t iac,
    const std::string & configured_name ) [inline]
```

## 5.415.2 Member Function Documentation

```
5.415.2.1 process() mha_wave_t* save_wave_t::process (
    mha_wave_t * s ) [inline]
```

```
5.415.2.2 prepare() void save_wave_t::prepare (
    mhaconfig_t & tf ) [inline], [virtual]
```

Implements **MHAPlugin::plugin\_t< MHA\_AC::waveform\_t >** (p. [1149](#)).

## 5.415.3 Member Data Documentation

```
5.415.3.1 basename std::string save_wave_t::basename [private]
```

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

- **save\_wave.cpp**

## 5.416 shadowfilter\_begin::cfg\_t Class Reference

### Public Member Functions

- **cfg\_t** (int nfft, int inch, int outch, **algo\_comm\_t** ac, std::string name)
- **mha\_spec\_t \* process ( mha\_spec\_t \*)**

## Private Attributes

- **MHA\_AC::spectrum\_t in\_spec\_copy**
- **MHASignal::spectrum\_t out\_spec**
- **MHA\_AC::int\_t nch**
- **MHA\_AC::int\_t ntracks**

### 5.416.1 Constructor & Destructor Documentation

**5.416.1.1 cfg\_t()** `cfg_t::cfg_t (`  
    `int nfft,`  
    `int inch,`  
    `int outch,`  
    `algo_comm_t ac,`  
    `std::string name )`

### 5.416.2 Member Function Documentation

**5.416.2.1 process()** `mha_spec_t * cfg_t::process (`  
    `mha_spec_t * s )`

### 5.416.3 Member Data Documentation

**5.416.3.1 in\_spec\_copy** `MHA_AC::spectrum_t shadowfilter_begin::cfg_t::in_spec_`  
`copy [private]`

**5.416.3.2 out\_spec** `MHASignal::spectrum_t shadowfilter_begin::cfg_t::out_spec`  
`[private]`

## **5.417 shadowfilter\_begin::shadowfilter\_begin\_t Class Reference**

**5.416.3.3 nch** `MHA_AC::int_t shadowfilter_begin::cfg_t::nch [private]`

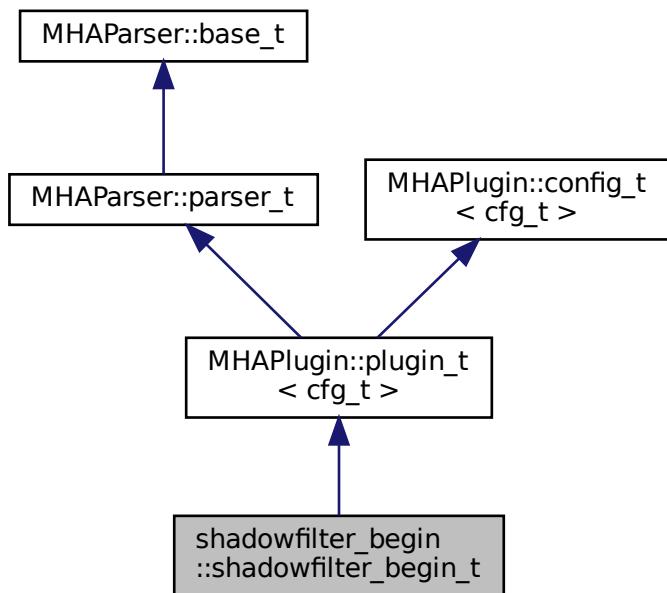
**5.416.3.4 ntracks** `MHA_AC::int_t shadowfilter_begin::cfg_t::ntracks [private]`

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

- `shadowfilter_begin.cpp`

## **5.417 shadowfilter\_begin::shadowfilter\_begin\_t Class Reference**

Inheritance diagram for `shadowfilter_begin::shadowfilter_begin_t`:



### **Public Member Functions**

- `shadowfilter_begin_t ( algo_comm_t iac, const std::string &configured_name)`
- `mha_spec_t * process ( mha_spec_t *)`
- `void prepare ( mhaconfig_t &)`

## Private Attributes

- std::string **basename**
- MHAParser::int\_t **nch**
- MHAParser::int\_t **ntracks**

## Additional Inherited Members

### 5.417.1 Constructor & Destructor Documentation

**5.417.1.1 shadowfilter\_begin\_t()** shadowfilter\_begin::shadowfilter\_begin\_t::shadowfilter\_begin\_t (

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

### 5.417.2 Member Function Documentation

**5.417.2.1 process()** mha\_spec\_t \* shadowfilter\_begin::shadowfilter\_begin\_t::process (

```
mha_spec_t * s )
```

**5.417.2.2 prepare()** void shadowfilter\_begin::shadowfilter\_begin\_t::prepare (

```
mhaconfig_t & tf ) [virtual]
```

Implements **MHAParser::parser\_t<cfg\_t>** (p. 1149).

### 5.417.3 Member Data Documentation

**5.417.3.1 basename** std::string shadowfilter\_begin::shadowfilter\_begin\_t::basename  
[private]

**5.417.3.2 nch** MHAParser::int\_t shadowfilter\_begin::shadowfilter\_begin\_t::nch [private]

**5.417.3.3 ntracks** MHAParser::int\_t shadowfilter\_begin::shadowfilter\_begin\_t::ntracks [private]

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

- [shadowfilter\\_begin.cpp](#)

## 5.418 shadowfilter\_end::cfg\_t Class Reference

### Public Member Functions

- [cfg\\_t \(int nfft\\_, algo\\_comm\\_t ac\\_, std::string name\\_\)](#)
- [mha\\_spec\\_t \\* process \( mha\\_spec\\_t \\*\)](#)

### Private Attributes

- [algo\\_comm\\_t ac](#)
- [std::string name](#)
- [int nfft](#)
- [int ntracks](#)
- [int nch\\_out](#)
- [mha\\_spec\\_t in\\_spec](#)
- [MHASignal::spectrum\\_t out\\_spec](#)
- [MHA\\_AC::spectrum\\_t gains](#)

### 5.418.1 Constructor & Destructor Documentation

**5.418.1.1 cfg\_t()** `cfg_t::cfg_t (`  
    `int nfft_,`  
    `algo_comm_t ac_,`  
    `std::string name_ )`

## 5.418.2 Member Function Documentation

**5.418.2.1 process()** `mha_spec_t * cfg_t::process (`  
    `mha_spec_t * s )`

## 5.418.3 Member Data Documentation

**5.418.3.1 ac** `algo_comm_t shadowfilter_end::cfg_t::ac [private]`

**5.418.3.2 name** `std::string shadowfilter_end::cfg_t::name [private]`

**5.418.3.3 nfft** `int shadowfilter_end::cfg_t::nfft [private]`

**5.418.3.4 ntracks** `int shadowfilter_end::cfg_t::ntracks [private]`

**5.418.3.5 nch\_out** `int shadowfilter_end::cfg_t::nch_out [private]`

**5.418.3.6 in\_spec mha\_spec\_t shadowfilter\_end::cfg\_t::in\_spec [private]**

**5.418.3.7 out\_spec MHASignal::spectrum\_t shadowfilter\_end::cfg\_t::out\_spec [private]**

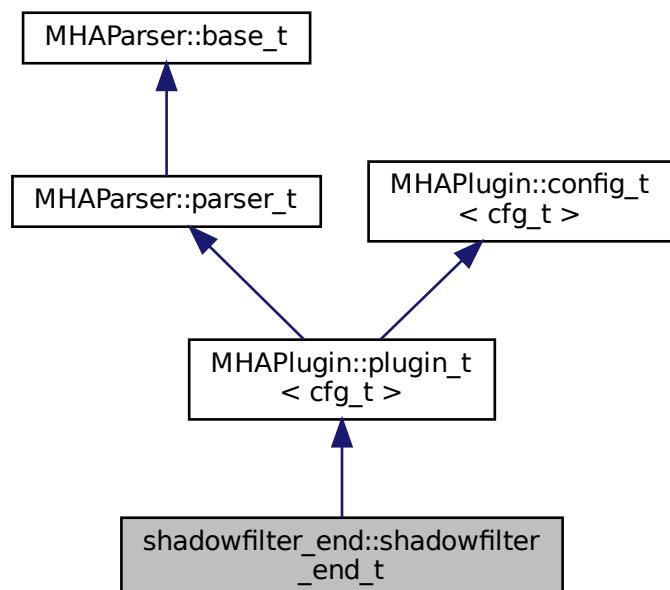
**5.418.3.8 gains MHA\_AC::spectrum\_t shadowfilter\_end::cfg\_t::gains [private]**

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

- **shadowfilter\_end.cpp**

## **5.419 shadowfilter\_end::shadowfilter\_end\_t Class Reference**

Inheritance diagram for shadowfilter\_end::shadowfilter\_end\_t:



## Public Member Functions

- `shadowfilter_end_t ( algo_comm_t iac, const std::string &configured_name)`
- `mha_spec_t * process ( mha_spec_t *)`
- `void prepare ( mhaconfig_t &)`

## Private Attributes

- `MHAParser::string_t basename`

## Additional Inherited Members

### 5.419.1 Constructor & Destructor Documentation

**5.419.1.1 shadowfilter\_end()** `shadowfilter_end::shadowfilter_end_t::shadowfilter_end_t ( algo_comm_t iac, const std::string & configured_name )`

### 5.419.2 Member Function Documentation

**5.419.2.1 process()** `mha_spec_t * shadowfilter_end::shadowfilter_end_t::process ( mha_spec_t * s )`

**5.419.2.2 prepare()** `void shadowfilter_end::shadowfilter_end_t::prepare ( mhaconfig_t & tf ) [virtual]`

Implements `MHAParser::plugin_t< cfg_t >` (p. 1149).

### 5.419.3 Member Data Documentation

**5.419.3.1 basename** `MHAParser::string_t shadowfilter_end::shadowfilter_end_t::basename` [private]

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

- `shadowfilter_end.cpp`

## 5.420 sine\_cfg\_t Struct Reference

Runtime configuration of the sine plugin.

### Public Member Functions

- `sine_cfg_t` (double sampling\_rate, `mha_real_t` frequency, `mha_real_t` newlev, int \_mix, const std::vector< int > &\_channels)

*Constructor computes data members from input parameters.*

### Public Attributes

- double `phase_increment_div_2pi`  
*Phase increment per sample, divided by 2 pi for easier phase wrapping.*
- double `amplitude`  
*Amplitude of the sinusoid in Pascal.*
- int `mix`  
*0 for mode replace, 1 for mode mix. Used as factor on input signal.*
- const std::vector< int > `channels`  
*Indices of affected audio channels.*

### 5.420.1 Detailed Description

Runtime configuration of the sine plugin.

### 5.420.2 Constructor & Destructor Documentation

```
5.420.2.1 sine_cfg_t() sine_cfg_t::sine_cfg_t (
    double sampling_rate,
    mha_real_t frequency,
    mha_real_t newlev,
    int _mix,
    const std::vector< int > & _channels ) [inline]
```

Constructor computes data members from input parameters.

### 5.420.3 Member Data Documentation

5.420.3.1 **phase\_increment\_div\_2pi** double sine\_cfg\_t::phase\_increment\_div\_2pi

Phase increment per sample, divided by 2 pi for easier phase wrapping.

5.420.3.2 **amplitude** double sine\_cfg\_t::amplitude

Amplitude of the sinusoid in Pascal.

5.420.3.3 **mix** int sine\_cfg\_t::mix

0 for mode replace, 1 for mode mix. Used as factor on input signal.

5.420.3.4 **channels** const std::vector<int> sine\_cfg\_t::channels

Indices of affected audio channels.

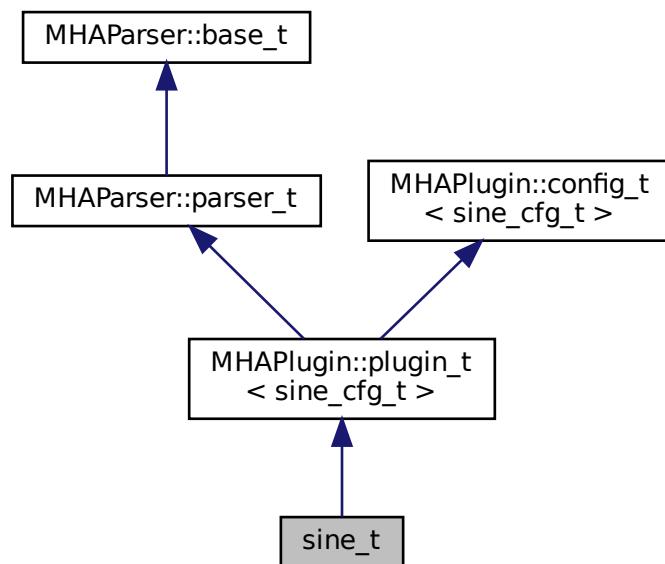
The documentation for this struct was generated from the following file:

- **sine.cpp**

## 5.421 sine\_t Class Reference

Interface class of plugin `sine`, a sinusoid generator plugin.

Inheritance diagram for `sine_t`:



### Public Member Functions

- **`sine_t ( algo_comm_t iac, const std::string &configured_name )`**  
*Constructor initializes and connects configuration variables.*
- **`mha_wave_t * process ( mha_wave_t * )`**  
*Computes sinusoid and mixes/replaces input signal.*
- **`void prepare ( mhaconfig_t & )`**  
*Adapts range of channel variable and prepares.*

### Private Member Functions

- **`void update_cfg ()`**  
*Computes new runtime configuration.*

## Private Attributes

- `MHAParser::float_t lev`
- `MHAParser::float_t frequency`
- `MHAParser::kw_t mode`
- `MHAParser::vint_t channels`
- `double phase_div_2pi`
- `MHAEvents::patchbay_t< sine_t > patchbay`

## Additional Inherited Members

### 5.421.1 Detailed Description

Interface class of plugin `sine`, a sinusoid generator plugin.

### 5.421.2 Constructor & Destructor Documentation

```
5.421.2.1 sine_t() sine_t::sine_t (
    algo_comm_t iac,
    const std::string & configured_name )
```

Constructor initializes and connects configuration variables.

### 5.421.3 Member Function Documentation

```
5.421.3.1 process() mha_wave_t * sine_t::process (
    mha_wave_t * s )
```

Computes sinusoid and mixes/replaces input signal.

If the amplitude has changed since the last process callback, spread out the amplitude change linearly across all samples of the buffer to avoid clicks.

**5.421.3.2 `prepare()`** `void sine_t::prepare (`  
`mhaconfig_t & tf ) [virtual]`

Adapts range of channel variable and prepares.

Implements `MHAPlugIn::plugin_t< sine_cfg_t >` (p. 1149).

**5.421.3.3 `update_cfg()`** `void sine_t::update_cfg ( ) [private]`

Computes new runtime configuration.

#### 5.421.4 Member Data Documentation

**5.421.4.1 `lev`** `MHAParser::float_t sine_t::lev [private]`

**5.421.4.2 `frequency`** `MHAParser::float_t sine_t::frequency [private]`

**5.421.4.3 `mode`** `MHAParser::kw_t sine_t::mode [private]`

**5.421.4.4 `channels`** `MHAParser::vint_t sine_t::channels [private]`

**5.421.4.5 `phase_div_2pi`** `double sine_t::phase_div_2pi [private]`

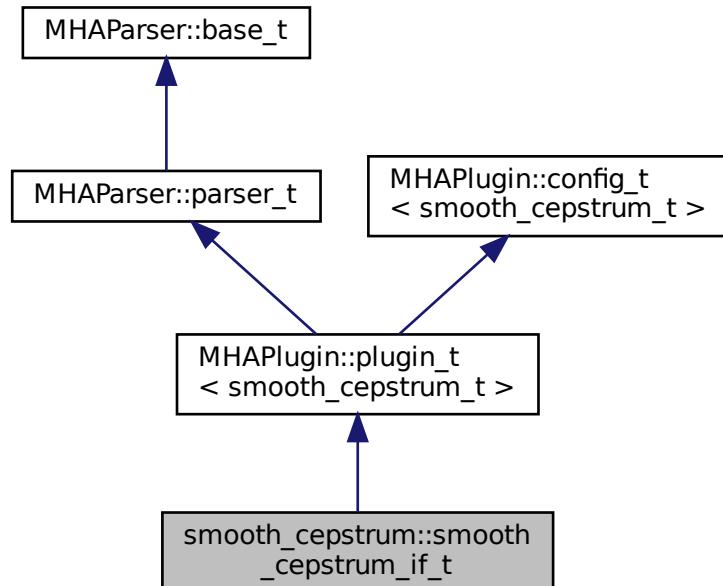
#### 5.421.4.6 patchbay `MHAEEvents::patchbay_t< sine_t> sine_t::patchbay [private]`

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

- `sine.cpp`

#### 5.422 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 iac, const std::string &configured_name)`**  
*Constructs the beamforming plugin.*
- **`mha_spec_t * process ( mha_spec_t *)`**  
*This plugin implements noise reduction using spectral subtraction: by nonnegative subtraction from the output magnitude of the estimated noise magnitude spectrum.*
- **`void prepare ( mhaconfig_t &)`**  
*Plugin preparation.*
- **`void release (void)`**

### Private Member Functions

- void `update_cfg ()`
- void `on_model_param_valuechanged ()`

### Private Attributes

- `MHAParser::float_t xi_min_db`
- `MHAParser::float_t f0_low`
- `MHAParser::float_t f0_high`
- `MHAParser::float_t delta_pitch`
- `MHAParser::float_t lambda_thresh`
- `MHAParser::float_t alpha_pitch`
- `MHAParser::float_t beta_const`
- `MHAParser::float_t kappa_const`
- `MHAParser::float_t gain_min_db`
- `MHAParser::vfloat_t win_f0`
- `MHAParser::vfloat_t alpha_const_vals`
- `MHAParser::vfloat_t alpha_const_limits_hz`
- `MHAParser::string_t noisePow_name`
- `MHAParser::parser_t spp`
- `MHAParser::float_t prior_q`
- `MHAParser::float_t xi_opt_db`
- `MHAEvents::patchbay_t < smooth_cepstrum_if_t > patchbay`
- bool `prepared`

### Additional Inherited Members

#### 5.422.1 Constructor & Destructor Documentation

```
5.422.1.1 smooth_cepstrum_if_t() smooth_cepstrum::smooth_cepstrum_if_t::smooth_cepstrum_if_t (
    algo_comm_t iac,
    const std::string & configured_name )
```

Constructs the beamforming plugin.

#### 5.422.2 Member Function Documentation

```
5.422.2.1 process() mha_spec_t * smooth_cepstrum::smooth_cepstrum_if_t::process (
    mha_spec_t * signal )
```

This plugin implements noise reduction using spectral subtraction: by nonnegative subtraction from the output magnitude of the estimated noise magnitude spectrum.

## Parameters

<i>signal</i>	Pointer to the input signal structure.
---------------	--

## Returns

Returns a pointer to the input signal structure, with a the signal modified by this plugin.

**5.422.2.2 `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. [1149](#)).

**5.422.2.3 `release()`** `void smooth_cepstrum::smooth_cepstrum_if_t::release ( void ) [inline], [virtual]`

Reimplemented from `MHAPlugin::plugin_t< smooth_cepstrum_t >` (p. [1150](#)).

**5.422.2.4 `update_cfg()`** `void smooth_cepstrum::smooth_cepstrum_if_t::update_cfg ( ) [private]`

**5.422.2.5 `on_model_param_valuechanged()`** `void smooth_cepstrum::smooth_cepstrum_if_t::on_model_param_valuechanged ( ) [private]`

### 5.422.3 Member Data Documentation

**5.422.3.1 xi\_min\_db** `MHAParser::float_t` `smooth_cepstrum::smooth_cepstrum_if_t::xi_min_db` [private]

**5.422.3.2 f0\_low** `MHAParser::float_t` `smooth_cepstrum::smooth_cepstrum_if_t::f0_low` [private]

**5.422.3.3 f0\_high** `MHAParser::float_t` `smooth_cepstrum::smooth_cepstrum_if_t::f0_high` [private]

**5.422.3.4 delta\_pitch** `MHAParser::float_t` `smooth_cepstrum::smooth_cepstrum_if_t::delta_pitch` [private]

**5.422.3.5 lambda\_thresh** `MHAParser::float_t` `smooth_cepstrum::smooth_cepstrum_if_t::lambda_thresh` [private]

**5.422.3.6 alpha\_pitch** `MHAParser::float_t` `smooth_cepstrum::smooth_cepstrum_if_t::alpha_pitch` [private]

**5.422.3.7 beta\_const** `MHAParser::float_t` `smooth_cepstrum::smooth_cepstrum_if_t::beta_const` [private]

**5.422.3.8 kappa\_const** `MHAParser::float_t` `smooth_cepstrum::smooth_cepstrum_if_t::kappa_const` [private]

**5.422.3.9 gain\_min\_db** `MHAParser::float_t` `smooth_cepstrum::smooth_cepstrum_if_t::gain_min_db` [private]

**5.422.3.10 win\_f0** `MHAParser::vfloat_t` `smooth_cepstrum::smooth_cepstrum_if_t::win_f0` [private]

**5.422.3.11 alpha\_const\_vals** `MHAParser::vfloat_t` `smooth_cepstrum::smooth_cepstrum_if_t::alpha_const_vals` [private]

**5.422.3.12 alpha\_const\_limits\_hz** `MHAParser::vfloat_t` `smooth_cepstrum::smooth_cepstrum_if_t::alpha_const_limits_hz` [private]

**5.422.3.13 noisePow\_name** `MHAParser::string_t` `smooth_cepstrum::smooth_cepstrum_if_t::noisePow_name` [private]

**5.422.3.14 spp** `MHAParser::parser_t` `smooth_cepstrum::smooth_cepstrum_if_t::spp` [private]

**5.422.3.15 prior\_q** `MHAParser::float_t` `smooth_cepstrum::smooth_cepstrum_if_t::prior_q` [private]

## **5.423 smooth\_cepstrum::smooth\_cepstrum\_t Class Reference 1439**

**5.422.3.16 xi\_opt\_db** `MHAParser::float_t smooth_cepstrum::smooth_cepstrum_if_t::xi_opt_db` [private]

**5.422.3.17 patchbay** `MHAEvents::patchbay_t< smooth_cepstrum_if_t> smooth_cepstrum::smooth_cepstrum_if_t::patchbay` [private]

**5.422.3.18 prepared** `bool smooth_cepstrum::smooth_cepstrum_if_t::prepared` [private]

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

- **smooth\_cepstrum.hh**
- **smooth\_cepstrum.cpp**

## **5.423 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**

### 5.423.1 Constructor & Destructor Documentation

**5.423.1.1 smooth\_cepstrum\_t() [1/2]** `smooth_cepstrum::smooth_cepstrum_t::smooth_cepstrum_t (`

```
algo_comm_t & ac,
smooth_params & params )
```

**5.423.1.2 smooth\_cepstrum\_t() [2/2]** `smooth_cepstrum::smooth_cepstrum_t::smooth_cepstrum_t (`

```
const smooth_cepstrum_t & ) [delete]
```

## **5.423 smooth\_cepstrum::smooth\_cepstrum\_t Class Reference 1441**

**5.423.1.3 ~smooth\_cepstrum\_t()** `smooth_cepstrum::smooth_cepstrum_t::~smooth_cepstrum_t ( )`

### **5.423.2 Member Function Documentation**

**5.423.2.1 operator=()** `smooth_cepstrum_t& smooth_cepstrum::smooth_cepstrum_t::operator= ( const smooth_cepstrum_t & ) [delete]`

**5.423.2.2 process()** `mha_spec_t * smooth_cepstrum::smooth_cepstrum_t::process ( mha_spec_t * noisyFrame )`

### **5.423.3 Member Data Documentation**

**5.423.3.1 ac** `algo_comm_t smooth_cepstrum::smooth_cepstrum_t::ac [private]`

**5.423.3.2 params** `smooth_params smooth_cepstrum::smooth_cepstrum_t::params [private]`

**5.423.3.3 fftlen** `unsigned int smooth_cepstrum::smooth_cepstrum_t::ffflen [private]`

**5.423.3.4 mha\_fft** `mha_fft_t smooth_cepstrum::smooth_cepstrum_t::mha_fft [private]`

**5.423.3.5 nfreq** unsigned int smooth\_cepstrum::smooth\_cepstrum\_t::nfreq [private]

**5.423.3.6 nchan** unsigned int smooth\_cepstrum::smooth\_cepstrum\_t::nchan [private]

**5.423.3.7 ola\_powspec\_scale** float smooth\_cepstrum::smooth\_cepstrum\_t::ola\_powspec←  
\_scale [private]

**5.423.3.8 q\_low** float smooth\_cepstrum::smooth\_cepstrum\_t::q\_low [private]

**5.423.3.9 q\_high** float smooth\_cepstrum::smooth\_cepstrum\_t::q\_high [private]

**5.423.3.10 winF0** MHASignal::waveform\_t smooth\_cepstrum::smooth\_cepstrum\_t::winF0  
[private]

**5.423.3.11 xi\_min** float smooth\_cepstrum::smooth\_cepstrum\_t::xi\_min [private]

**5.423.3.12 gain\_min** float smooth\_cepstrum::smooth\_cepstrum\_t::gain\_min [private]

**5.423.3.13 alpha\_const** MHASignal::waveform\_t smooth\_cepstrum::smooth\_cepstrum\_t←  
::alpha\_const [private]

## **5.423 smooth\_cepstrum::smooth\_cepstrum\_t Class Reference 1443**

**5.423.3.14 alpha\_prev** `MHASignal::waveform_t` `smooth_cepstrum::smooth_cepstrum_t::alpha_prev` [private]

**5.423.3.15 noisePow** `MHASignal::waveform_t` `smooth_cepstrum::smooth_cepstrum_t::noisePow` [private]

**5.423.3.16 powSpec** `MHASignal::waveform_t` `smooth_cepstrum::smooth_cepstrum_t::powSpec` [private]

**5.423.3.17 gamma\_post** `MHASignal::waveform_t` `smooth_cepstrum::smooth_cepstrum_t::gamma_post` [private]

**5.423.3.18 xi\_ml** `MHASignal::waveform_t` `smooth_cepstrum::smooth_cepstrum_t::xi_ml` [private]

**5.423.3.19 lambda\_ml\_full** `MHASignal::spectrum_t` `smooth_cepstrum::smooth_cepstrum_t::lambda_ml_full` [private]

**5.423.3.20 lambda\_ml\_ceps** `MHASignal::spectrum_t` `smooth_cepstrum::smooth_cepstrum_t::lambda_ml_ceps` [private]

**5.423.3.21 lambda\_ml\_smooth** `MHASignal::waveform_t` `smooth_cepstrum::smooth_cepstrum_t::lambda_ml_smooth` [private]

**5.423.3.22 alpha\_hat** `MHASignal::waveform_t` `smooth_cepstrum::smooth_cepstrum_t`  
  `::alpha_hat` [private]

**5.423.3.23 alpha\_frame** `MHASignal::waveform_t` `smooth_cepstrum::smooth_cepstrum_t`  
  `::alpha_frame` [private]

**5.423.3.24 lambda\_ceps** `MHASignal::spectrum_t` `smooth_cepstrum::smooth_cepstrum_t`  
  `::lambda_ceps` [private]

**5.423.3.25 lambda\_ceps\_prev** `MHASignal::waveform_t` `smooth_cepstrum::smooth_cepstrum_t`  
  `::lambda_ceps_prev` [private]

**5.423.3.26 log\_lambda\_spec** `MHASignal::spectrum_t` `smooth_cepstrum::smooth_cepstrum_t`  
  `::log_lambda_spec` [private]

**5.423.3.27 lambda\_spec** `MHASignal::waveform_t` `smooth_cepstrum::smooth_cepstrum_t`  
  `::lambda_spec` [private]

**5.423.3.28 xi\_est** `MHASignal::waveform_t` `smooth_cepstrum::smooth_cepstrum_t`  
  `::xi_est` [private]

**5.423.3.29 gain\_wiener** `MHASignal::waveform_t` `smooth_cepstrum::smooth_cepstrum_t`  
  `::gain_wiener` [private]

## **5.423 smooth\_cepstrum::smooth\_cepstrum\_t Class Reference 1445**

**5.423.3.30 spec\_out** `MHASignal::spectrum_t` `smooth_cepstrum::smooth_cepstrum_t::spec_out` [private]

**5.423.3.31 max\_val** `double*` `smooth_cepstrum::smooth_cepstrum_t::max_val` [private]

**5.423.3.32 max\_q** `int*` `smooth_cepstrum::smooth_cepstrum_t::max_q` [private]

**5.423.3.33 pitch\_set\_first** `int*` `smooth_cepstrum::smooth_cepstrum_t::pitch_set_first` [private]

**5.423.3.34 pitch\_set\_last** `int*` `smooth_cepstrum::smooth_cepstrum_t::pitch_set_last` [private]

**5.423.3.35 priorFact** `float` `smooth_cepstrum::smooth_cepstrum_t::priorFact` [private]

**5.423.3.36 xiOpt** `float` `smooth_cepstrum::smooth_cepstrum_t::xiOpt` [private]

**5.423.3.37 logGLRFact** `float` `smooth_cepstrum::smooth_cepstrum_t::logGLRFact` [private]

**5.423.3.38 GLRexp** `float` `smooth_cepstrum::smooth_cepstrum_t::GLRexp` [private]

**5.423.3.39 GLR** `MHASignal::waveform_t smooth_cepstrum::smooth_cepstrum_t::GLR`  
[private]

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

- `smooth_cepstrum.hh`
- `smooth_cepstrum.cpp`

## 5.424 `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`

### 5.424.1 Constructor & Destructor Documentation

```
5.424.1.1 smooth_params() smooth_cepstrum::smooth_params::smooth_params (
    const mhaconfig_t & _in_cfg,
    float _xi_min_db,
    float _f0_low,
    float _f0_high,
    float _delta_pitch,
    float _lambda_thresh,
    float _alpha_pitch,
    float _beta_const,
    float _kappa_const,
    float _prior_q,
    float _xi_opt_db,
    float _gain_min_db,
    std::vector< float > & _winF0,
    std::vector< float > & _alpha_const_vals,
    std::vector< float > & _alpha_const_limits_hz,
    std::string & _noisePow_name ) [inline]
```

## 5.424.2 Member Data Documentation

**5.424.2.1 in\_cfg** const mhaconfig\_t smooth\_cepstrum::smooth\_params::in\_cfg

**5.424.2.2 xi\_min\_db** float smooth\_cepstrum::smooth\_params::xi\_min\_db

**5.424.2.3 f0\_low** float smooth\_cepstrum::smooth\_params::f0\_low

**5.424.2.4 f0\_high** float smooth\_cepstrum::smooth\_params::f0\_high

**5.424.2.5 delta\_pitch** float smooth\_cepstrum::smooth\_params::delta\_pitch

**5.424.2.6 lambda\_thresh** float smooth\_cepstrum::smooth\_params::lambda\_thresh

**5.424.2.7 alpha\_pitch** float smooth\_cepstrum::smooth\_params::alpha\_pitch

**5.424.2.8 beta\_const** float smooth\_cepstrum::smooth\_params::beta\_const

**5.424.2.9 kappa\_const** float smooth\_cepstrum::smooth\_params::kappa\_const

**5.424.2.10 prior\_q** float smooth\_cepstrum::smooth\_params::prior\_q

**5.424.2.11 xi\_opt\_db** float smooth\_cepstrum::smooth\_params::xi\_opt\_db

**5.424.2.12 gain\_min\_db** float smooth\_cepstrum::smooth\_params::gain\_min\_db

**5.424.2.13 winF0** std::vector<float> smooth\_cepstrum::smooth\_params::winF0

**5.424.2.14 alpha\_const\_vals** std::vector<float> smooth\_cepstrum::smooth\_params::alpha\_const\_vals

**5.424.2.15 alpha\_const\_limits\_hz** std::vector<float> smooth\_cepstrum::smooth\_<→  
params::alpha\_const\_limits\_hz

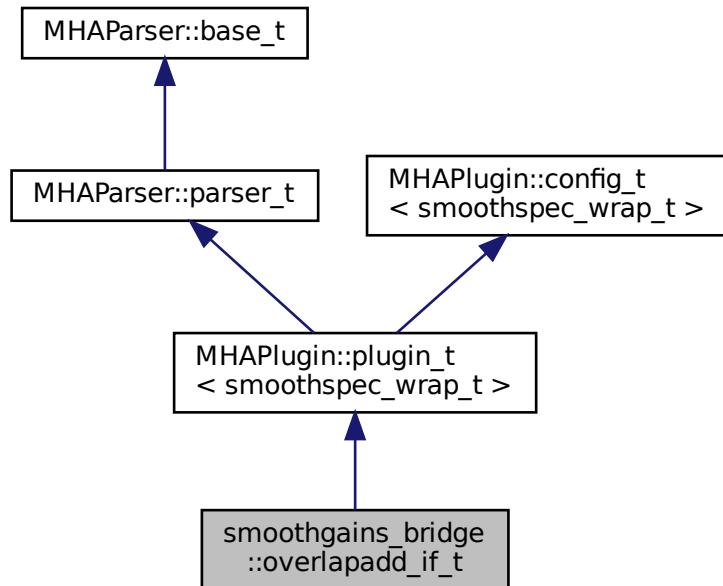
**5.424.2.16 noisePow\_name** std::string smooth\_cepstrum::smooth\_params::noisePow\_<→  
name

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

- **smooth\_cepstrum.hh**

## 5.425 smoothgains\_bridge::overlapadd\_if\_t Class Reference

Inheritance diagram for smoothgains\_bridge::overlapadd\_if\_t:



### Public Member Functions

- **overlapadd\_if\_t ( algo\_comm\_t iac, const std::string &configured\_name)**
- **~overlapadd\_if\_t ()**
- **void prepare ( mhaconfig\_t &)**
- **void release ()**
- **mha\_spec\_t \* process ( mha\_spec\_t \*)**

## Private Member Functions

- void **update ()**

## Private Attributes

- **MHAEvents::patchbay\_t< overlapadd\_if\_t > patchbay**
- **MHAParser::kw\_t mode**
- **MHAParser::window\_t irswnd**
- **MHAParser::float\_t epsilon**
- **MHAParser::mhapluginloader\_t plugloader**
- std::string **algo**
- **mhaconfig\_t cf\_in**
- **mhaconfig\_t cf\_out**

## Additional Inherited Members

### 5.425.1 Constructor & Destructor Documentation

**5.425.1.1 `overlapadd_if_t()`** smoothgains\_bridge::overlapadd\_if\_t::overlapadd\_if\_t (   
`algo_comm_t iac,`  
`const std::string & configured_name )`

**5.425.1.2 `~overlapadd_if_t()`** smoothgains\_bridge::overlapadd\_if\_t::~overlapadd\_if\_t ( )

### 5.425.2 Member Function Documentation

**5.425.2.1 `prepare()`** void smoothgains\_bridge::overlapadd\_if\_t::prepare (   
`mhaconfig_t & t ) [virtual]`

Implements **MHAPlugin::plugin\_t< smoothspec\_wrap\_t >** (p. 1149).

**5.425.2.2 release()** void smoothgains\_bridge::overlapadd\_if\_t::release () [virtual]

Reimplemented from **MHAPlugIn::plugin\_t<smoothspec\_wrap\_t>** (p. 1150).

**5.425.2.3 process()** mha\_spec\_t \* smoothgains\_bridge::overlapadd\_if\_t::process (mha\_spec\_t \* spec )

**5.425.2.4 update()** void smoothgains\_bridge::overlapadd\_if\_t::update () [private]

### **5.425.3 Member Data Documentation**

**5.425.3.1 patchbay** MHAEvents::patchbay\_t< overlapadd\_if\_t> smoothgains\_bridge::overlapadd\_if\_t::patchbay [private]

**5.425.3.2 mode** MHAParser::kw\_t smoothgains\_bridge::overlapadd\_if\_t::mode [private]

**5.425.3.3 irswnd** MHAParser::window\_t smoothgains\_bridge::overlapadd\_if\_t::irswnd [private]

**5.425.3.4 epsilon** MHAParser::float\_t smoothgains\_bridge::overlapadd\_if\_t::epsilon [private]

**5.425.3.5 plugloader** `MHAParser::mhapluginloader_t` `smoothgains_bridge::overlapadd_if_t::plugloader` [private]

**5.425.3.6 algo** `std::string` `smoothgains_bridge::overlapadd_if_t::algo` [private]

**5.425.3.7 cf\_in** `mhaconfig_t` `smoothgains_bridge::overlapadd_if_t::cf_in` [private]

**5.425.3.8 cf\_out** `mhaconfig_t` `smoothgains_bridge::overlapadd_if_t::cf_out` [private]

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

- `smoothgains_bridge.cpp`

## 5.426 smoothgains\_bridge::smoothspec\_wrap\_t Class Reference

### Public Member Functions

- `smoothspec_wrap_t` ( `mhaconfig_t` `spar_in`, `mhaconfig_t` `spar_out`, const `MHAParser::kw_t` &`mode`, const `MHAParser::window_t` &`irswnd`, const `MHAParser::float_t` &`epsilon`)
- `mha_spec_t * proc_1` ( `mha_spec_t *`)
- `mha_spec_t * proc_2` ( `mha_spec_t *`)

### Private Attributes

- **MHASignal::spectrum\_t spec\_in\_copy**  
*Copy of input spectrum for smoothspec.*
- **MHAFilter::smoothspec\_t smoothspec**  
*Smoothspec calculator.*
- `bool use_smoothspec`
- `float smoothspec_epsilon`

## 5.426.1 Constructor & Destructor Documentation

**5.426.1.1 smoothspec\_wrap\_t()** smoothgains\_bridge::smoothspec\_wrap\_t::smoothspec\_wrap\_t (

```
    mhaconfig_t spar_in,
    mhaconfig_t spar_out,
    const MHAParser::kw_t & mode,
    const MHAParser::window_t & irswnd,
    const MHAParser::float_t & epsilon )
```

## 5.426.2 Member Function Documentation

**5.426.2.1 proc\_1()** mha\_spec\_t \* smoothgains\_bridge::smoothspec\_wrap\_t::proc\_1 (

```
    mha_spec_t * s )
```

**5.426.2.2 proc\_2()** mha\_spec\_t \* smoothgains\_bridge::smoothspec\_wrap\_t::proc\_2 (

```
    mha_spec_t * s )
```

## 5.426.3 Member Data Documentation

**5.426.3.1 spec\_in\_copy** MHASignal::spectrum\_t smoothgains\_bridge::smoothspec\_wrap\_t::spec\_in\_copy [private]

Copy of input spectrum for smoothspec.

**5.426.3.2 smoothspec** MHAFilter::smoothspec\_t smoothgains\_bridge::smoothspec\_wrap\_t::smoothspec [private]

Smoothspec calculator.

**5.426.3.3 use\_smoothspec** bool smoothgains\_bridge::smoothspec\_wrap\_t::use\_smoothspec  
[private]

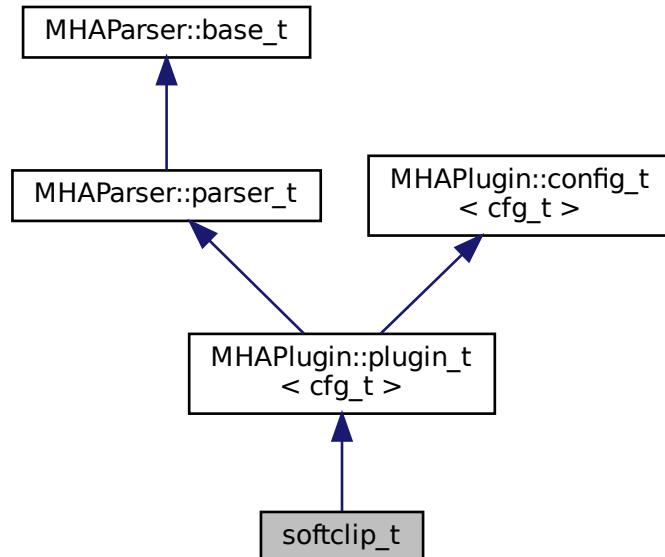
**5.426.3.4 smoothspec\_epsilon** float smoothgains\_bridge::smoothspec\_wrap\_t::smoothspec←  
\_epsilon [private]

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

- **smoothgains\_bridge.cpp**

## 5.427 softclip\_t Class Reference

Inheritance diagram for softclip\_t:



### Public Member Functions

- **softclip\_t ( algo\_comm\_t iac, const std::string &configured\_name)**
- **mha\_wave\_t \* process ( mha\_wave\_t \*)**
- **void prepare ( mhaconfig\_t &)**
- **void update ()**

## Private Attributes

- `mhaconfig_t tftype`
- `MHAParser::float_t attack`
- `MHAParser::float_t decay`
- `MHAParser::float_t start_limit`
- `MHAParser::float_t slope_db`
- `MHAEvents::patchbay_t< softclip_t > patchbay`

## Additional Inherited Members

### 5.427.1 Constructor & Destructor Documentation

```
5.427.1.1 softclip_t() softclip_t::softclip_t (
    algo_comm_t iac,
    const std::string & configured_name )
```

### 5.427.2 Member Function Documentation

```
5.427.2.1 process() mha_wave_t * softclip_t::process (
    mha_wave_t * s )
```

```
5.427.2.2 prepare() void softclip_t::prepare (
    mhaconfig_t & tf ) [virtual]
```

Implements `MHAPlugIn::plugin_t< cfg_t >` (p. 1149).

```
5.427.2.3 update() void softclip_t::update ( )
```

### 5.427.3 Member Data Documentation

**5.427.3.1 `tftype`** `mhaconfig_t` `softclip_t::tftype` [private]

**5.427.3.2 `attack`** `MHAParser::float_t` `softclip_t::attack` [private]

**5.427.3.3 `decay`** `MHAParser::float_t` `softclip_t::decay` [private]

**5.427.3.4 `start_limit`** `MHAParser::float_t` `softclip_t::start_limit` [private]

**5.427.3.5 `slope_db`** `MHAParser::float_t` `softclip_t::slope_db` [private]

**5.427.3.6 `patchbay`** `MHAEEvents::patchbay_t< softclip_t>` `softclip_t::patchbay` [private]

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

- `softclip.cpp`

### 5.428 `softclipper_t` Class Reference

#### Public Member Functions

- `softclipper_t (const softclipper_variables_t &v, const mhaconfig_t &)`
- `mha_real_t process (mha_wave_t *)`

## Private Attributes

- `MHAFilter::o1flt_lowpass_t attack`
- `MHAFilter::o1flt_maxtrack_t decay`
- `MHAFilter::o1flt_lowpass_t clipmeter`
- `mha_real_t threshold`
- `mha_real_t hardlimit`
- `mha_real_t slope`
- `bool linear`

### 5.428.1 Constructor & Destructor Documentation

```
5.428.1.1 softclipper_t() softclipper_t::softclipper_t (
    const softclipper_variables_t & v,
    const mhaconfig_t & tf )
```

### 5.428.2 Member Function Documentation

```
5.428.2.1 process() mha_real_t softclipper_t::process (
    mha_wave_t * s )
```

### 5.428.3 Member Data Documentation

```
5.428.3.1 attack MHAFilter::o1flt_lowpass_t softclipper_t::attack [private]
```

```
5.428.3.2 decay MHAFilter::o1flt_maxtrack_t softclipper_t::decay [private]
```

**5.428.3.3 clipmeter** `MHAFilter::olflt_lowpass_t` `softclipper_t::clipmeter` [private]

**5.428.3.4 threshold** `mha_real_t` `softclipper_t::threshold` [private]

**5.428.3.5 hardlimit** `mha_real_t` `softclipper_t::hardlimit` [private]

**5.428.3.6 slope** `mha_real_t` `softclipper_t::slope` [private]

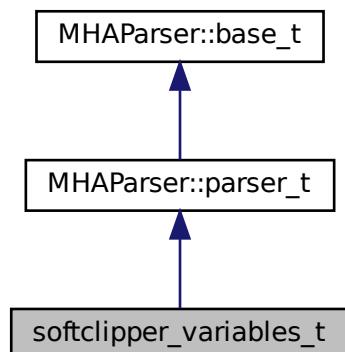
**5.428.3.7 linear** `bool` `softclipper_t::linear` [private]

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

- `transducers.cpp`

## 5.429 softclipper\_variables\_t Class Reference

Inheritance diagram for `softclipper_variables_t`:



## Public Member Functions

- `softclipper_variables_t ()`

## Public Attributes

- `MHAParser::float_t tau_attack`
- `MHAParser::float_t tau_decay`
- `MHAParser::float_t tau_clip`
- `MHAParser::float_t threshold`
- `MHAParser::float_t hardlimit`
- `MHAParser::float_t slope`
- `MHAParser::bool_t linear`
- `MHAParser::float_mon_t clipped`
- `MHAParser::float_t max_clipped`

## Additional Inherited Members

### 5.429.1 Constructor & Destructor Documentation

**5.429.1.1 `softclipper_variables_t()`** `softclipper_variables_t::softclipper_variables_t ( )`

### 5.429.2 Member Data Documentation

**5.429.2.1 `tau_attack`** `MHAParser::float_t softclipper_variables_t::tau_attack`

**5.429.2.2 `tau_decay`** `MHAParser::float_t softclipper_variables_t::tau_decay`

**5.429.2.3 tau\_clip** `MHAParser::float_t` `softclipper_variables_t::tau_clip`

**5.429.2.4 threshold** `MHAParser::float_t` `softclipper_variables_t::threshold`

**5.429.2.5 hardlimit** `MHAParser::float_t` `softclipper_variables_t::hardlimit`

**5.429.2.6 slope** `MHAParser::float_t` `softclipper_variables_t::slope`

**5.429.2.7 linear** `MHAParser::bool_t` `softclipper_variables_t::linear`

**5.429.2.8 clipped** `MHAParser::float_mon_t` `softclipper_variables_t::clipped`

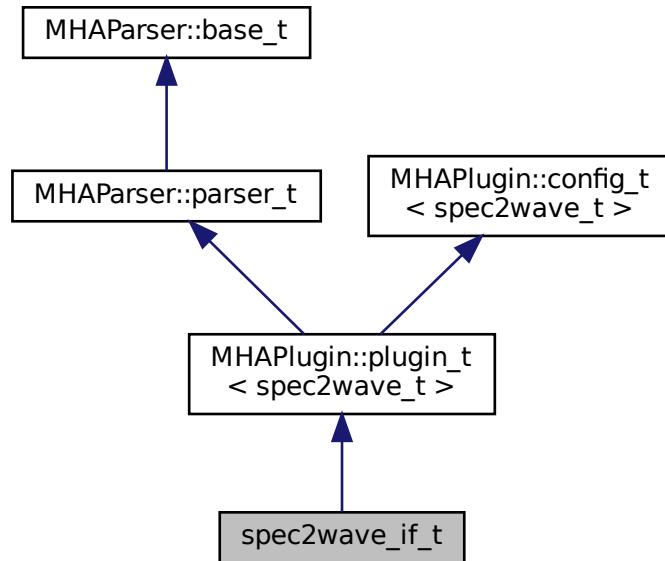
**5.429.2.9 max\_clipped** `MHAParser::float_t` `softclipper_variables_t::max_clipped`

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

- `transducers.cpp`

## 5.430 spec2wave\_if\_t Class Reference

Inheritance diagram for spec2wave\_if\_t:



### Public Member Functions

- `spec2wave_if_t ( algo_comm_t iac, const std::string &configured_name)`
- `void prepare ( mhaconfig_t &)`
- `void release ()`
- `mha_wave_t * process ( mha_spec_t *)`

### Private Member Functions

- `void update ()`
- `void setlock (bool b)`

### Private Attributes

- `MHAParser::float_t ramplen`
- `windowselector_t window_config`

## Additional Inherited Members

### 5.430.1 Constructor & Destructor Documentation

```
5.430.1.1 spec2wave_if_t() spec2wave_if_t::spec2wave_if_t (
    algo_comm_t iac,
    const std::string & configured_name )
```

### 5.430.2 Member Function Documentation

```
5.430.2.1 prepare() void spec2wave_if_t::prepare (
    mhaconfig_t & t ) [virtual]
```

Implements **MHAPlugin::plugin\_t< spec2wave\_t >** (p. [1149](#)).

```
5.430.2.2 release() void spec2wave_if_t::release () [virtual]
```

Reimplemented from **MHAPlugin::plugin\_t< spec2wave\_t >** (p. [1150](#)).

```
5.430.2.3 process() mha_wave_t * spec2wave_if_t::process (
    mha_spec_t * spec_in )
```

```
5.430.2.4 update() void spec2wave_if_t::update () [private]
```

```
5.430.2.5 setlock() void spec2wave_if_t::setlock (
    bool b ) [private]
```

### 5.430.3 Member Data Documentation

**5.430.3.1 ramplen** `MHAParser::float_t spec2wave_if_t::ramplen [private]`

**5.430.3.2 window\_config** `windowselector_t spec2wave_if_t::window_config [private]`

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

- `spec2wave.cpp`

## 5.431 spec2wave\_t Class Reference

### Public Member Functions

- `spec2wave_t` (unsigned int nfft\_, unsigned int nwnd\_, unsigned int nwndshift\_, unsigned int nch, `mha_real_t` ramplen, const `MHAWindow::base_t` &postwin)
- `~spec2wave_t ()`
- `mha_wave_t * process ( mha_spec_t *)`

### Private Attributes

- `mha_fft_t ft`  
*FFT class.*
- unsigned int `npad1`  
*length of zero padding before window*
- unsigned int `npad2`  
*length of zero padding after window*
- `hanning_ramps_t ramps`
- `MHASignal::waveform_t calc_out`
- `MHASignal::waveform_t out_buf`
- `MHASignal::waveform_t write_buf`
- `mha_real_t sc`
- unsigned int `nfft`
- unsigned int `nwndshift`
- `MHAWindow::base_t postwindow`

### 5.431.1 Constructor & Destructor Documentation

**5.431.1.1 `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 )`

**5.431.1.2 `~spec2wave_t()`** `spec2wave_t::~spec2wave_t ( )`

### 5.431.2 Member Function Documentation

**5.431.2.1 `process()`** `mha_wave_t * spec2wave_t::process (`  
    `mha_spec_t * spec_in )`

### 5.431.3 Member Data Documentation

**5.431.3.1 `ft`** `mha_fft_t spec2wave_t::ft [private]`

FFT class.

**5.431.3.2 `npad1`** `unsigned int spec2wave_t::npad1 [private]`

length of zero padding before window

**5.431.3.3 npad2** unsigned int spec2wave\_t::npad2 [private]

length of zero padding after window

**5.431.3.4 ramps** hanning\_ramps\_t spec2wave\_t::ramps [private]

**5.431.3.5 calc\_out** MHASignal::waveform\_t spec2wave\_t::calc\_out [private]

**5.431.3.6 out\_buf** MHASignal::waveform\_t spec2wave\_t::out\_buf [private]

**5.431.3.7 write\_buf** MHASignal::waveform\_t spec2wave\_t::write\_buf [private]

**5.431.3.8 sc** mha\_real\_t spec2wave\_t::sc [private]

**5.431.3.9 nfft** unsigned int spec2wave\_t::nfft [private]

**5.431.3.10 nwndshift** unsigned int spec2wave\_t::nwndshift [private]

**5.431.3.11 postwindow** MHAWindow::base\_t spec2wave\_t::postwindow [private]

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

- **spec2wave.cpp**

## 5.432 spec\_fader\_t Class Reference

### Public Member Functions

- `spec_fader_t` (unsigned int `ch`, `mha_real_t` `fr`, `MHAParser::vfloat_t` &`ng`, `MHAParser::float_t` &`t`)
- `~spec_fader_t` ()

### Public Attributes

- unsigned int `nch`
- `mha_real_t` \* `gains`
- unsigned int `fr`

#### 5.432.1 Constructor & Destructor Documentation

**5.432.1.1 `spec_fader_t()`** `spec_fader_t::spec_fader_t` (

```
    unsigned int ch,
    mha_real_t fr,
    MHAParser::vfloat_t & ng,
    MHAParser::float_t & t )
```

**5.432.1.2 `~spec_fader_t()`** `spec_fader_t::~spec_fader_t` ( ) [inline]

#### 5.432.2 Member Data Documentation

**5.432.2.1 `nch`** unsigned int `spec_fader_t::nch`

**5.432.2.2 `gains`** `mha_real_t*` `spec_fader_t::gains`

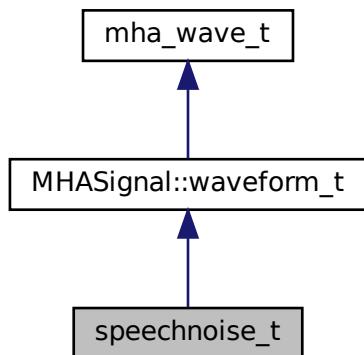
### 5.432.2.3 fr unsigned int spec\_fader\_t::fr

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

- [fader\\_spec.cpp](#)

## 5.433 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 srte, unsigned int **channels**, **speechnoise\_t**::  
    **noise\_type\_t** noise\_type= **speechnoise\_t**::mha)
- **speechnoise\_t** (unsigned int length\_samples, float srte, 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

### 5.433.1 Member Enumeration Documentation

#### 5.433.1.1 `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	

### 5.433.2 Constructor & Destructor Documentation

#### 5.433.2.1 `speechnoise_t()` [1/2] `speechnoise_t::speechnoise_t (` `float duration,`

```
float srate,  
unsigned int channels,  
speechnoise_t::noise_type_t noise_type = speechnoise_t::mha )
```

**5.433.2.2 speechnoise\_t() [2/2]** `speechnoise_t::speechnoise_t (`  
`unsigned int length_samples,`  
`float srate,`  
`unsigned int channels,`  
`speechnoise_t::noise_type_t noise_type = speechnoise_t::mha )`

### 5.433.3 Member Function Documentation

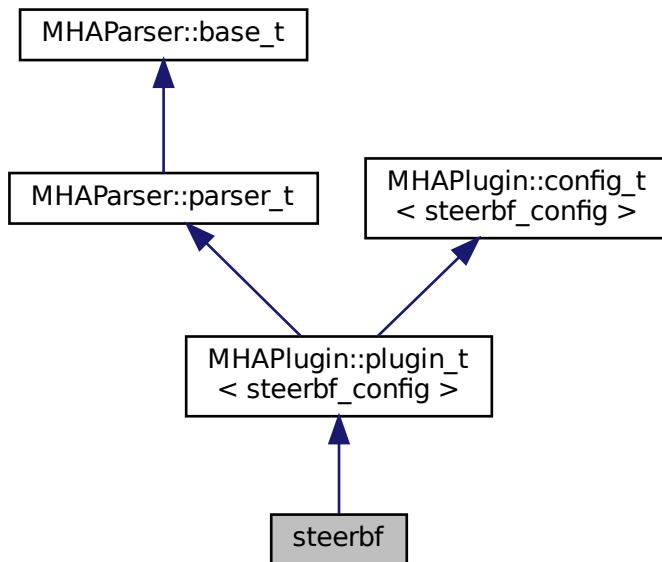
**5.433.3.1 creator()** `void speechnoise_t::creator (`  
`speechnoise_t::noise_type_t noise_type,`  
`float srate ) [private]`

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

- **speechnoise.h**
- **speechnoise.cpp**

## 5.434 steerbf Class Reference

Inheritance diagram for steerbf:



### Public Member Functions

- **steerbf ( algo\_comm\_t iac, const std::string &configured\_name)**  
*Constructs our plugin.*
- **~steerbf ()**
- **mha\_spec\_t \* process ( mha\_spec\_t \*)**  
*Defers to configuration class.*
- **void prepare ( mhaconfig\_t &)**  
*Plugin preparation.*
- **void release (void)**

### Public Attributes

- **MHAParser::string\_t bf\_src**
- **parser\_int\_dyn angle\_ind**
- **MHAParser::string\_t angle\_src**

### Private Member Functions

- void **update\_cfg ()**

### Private Attributes

- **MHAEvents::patchbay\_t< steerbf > patchbay**

### Additional Inherited Members

#### 5.434.1 Constructor & Destructor Documentation

```
5.434.1.1 steerbf() steerbf::steerbf (
    algo_comm_t iac,
    const std::string & configured_name )
```

Constructs our plugin.

```
5.434.1.2 ~steerbf() steerbf::~steerbf ( )
```

#### 5.434.2 Member Function Documentation

```
5.434.2.1 process() mha_spec_t * steerbf::process (
    mha_spec_t * signal )
```

Defers to configuration class.

```
5.434.2.2 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. 1149).

**5.434.2.3 `release()`** `void steerbf::release ( void ) [inline], [virtual]`

Reimplemented from `MHAPlugIn::plugin_t< steerbf_config >` (p. 1150).

**5.434.2.4 `update_cfg()`** `void steerbf::update_cfg ( ) [private]`

### 5.434.3 Member Data Documentation

**5.434.3.1 `bf_src`** `MHAParser::string_t steerbf::bf_src`

**5.434.3.2 `angle_ind`** `parser_int_dyn steerbf::angle_ind`

**5.434.3.3 `angle_src`** `MHAParser::string_t steerbf::angle_src`

**5.434.3.4 `patchbay`** `MHAEvents::patchbay_t< steerbf> steerbf::patchbay [private]`

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

- `steerbf.h`
- `steerbf.cpp`

## 5.435 steerbf\_config Class Reference

### Public Member Functions

- **steerbf\_config ( algo\_comm\_t & ac, const mhaconfig\_t in\_cfg, steerbf \* steerbf)**
- **~steerbf\_config ()**
- **mha\_spec\_t \* process ( mha\_spec\_t \*)**

### Private Attributes

- unsigned int **nchan**
- unsigned int **nfreq**
- **MHASignal::spectrum\_t outSpec**
- **mha\_spec\_t bf\_vec**
- unsigned int **nangle**
- **steerbf \* \_steerbf**
- **algo\_comm\_t & ac**
- std::string **bf\_src\_copy**

#### 5.435.1 Constructor & Destructor Documentation

**5.435.1.1 steerbf\_config()** steerbf\_config::steerbf\_config (

```
algo_comm_t & ac,
const mhaconfig_t in_cfg,
steerbf * steerbf )
```

**5.435.1.2 ~steerbf\_config()** steerbf\_config::~steerbf\_config ( )

#### 5.435.2 Member Function Documentation

**5.435.2.1 process()** mha\_spec\_t \* steerbf\_config::process (

```
mha_spec_t * inSpec )
```

### 5.435.3 Member Data Documentation

**5.435.3.1 nchan** unsigned int steerbf\_config::nchan [private]

**5.435.3.2 nfreq** unsigned int steerbf\_config::nfreq [private]

**5.435.3.3 outSpec** MHASignal::spectrum\_t steerbf\_config::outSpec [private]

**5.435.3.4 bf\_vec** mha\_spec\_t steerbf\_config::bf\_vec [private]

**5.435.3.5 nangle** unsigned int steerbf\_config::nangle [private]

**5.435.3.6 \_steerbf** steerbf\* steerbf\_config::\_steerbf [private]

**5.435.3.7 ac** algo\_comm\_t& steerbf\_config::ac [private]

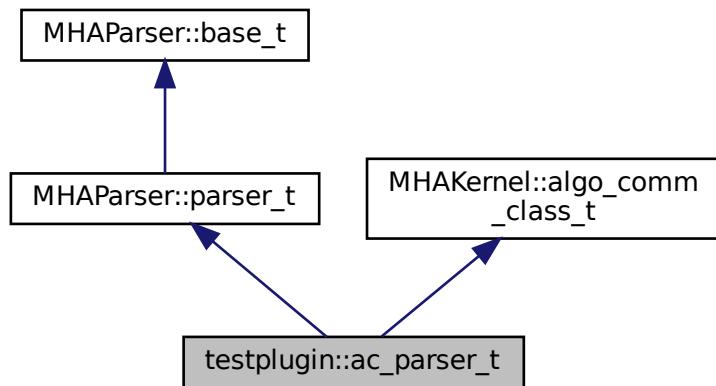
**5.435.3.8 bf\_src\_copy** std::string steerbf\_config::bf\_src\_copy [private]

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

- **steerbf.h**
- **steerbf.cpp**

## 5.436 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

### 5.436.1 Member Enumeration Documentation

#### 5.436.1.1 `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>	

### 5.436.2 Constructor & Destructor Documentation

#### 5.436.2.1 `ac_parser_t()` `testplugin::ac_parser_t::ac_parser_t ( ) [inline]`

### 5.436.3 Member Function Documentation

#### 5.436.3.1 `do_insert_var()` `void testplugin::ac_parser_t::do_insert_var ( ) [inline]`

Insert variable into AC space.

This leaks memory by design, as the plugin is for testing only

#### 5.436.3.2 `do_get_var()` `void testplugin::ac_parser_t::do_get_var ( ) [inline]`

## 5.436.4 Member Data Documentation

**5.436.4.1 insert\_var** `MHAParser::string_t` `testplugin::ac_parser_t::insert_var`

**5.436.4.2 get\_var** `MHAParser::string_t` `testplugin::ac_parser_t::get_var`

**5.436.4.3 data\_type** `MHAParser::kw_t` `testplugin::ac_parser_t::data_type`

**5.436.4.4 num\_entries** `MHAParser::int_t` `testplugin::ac_parser_t::num_entries`

**5.436.4.5 stride** `MHAParser::int_t` `testplugin::ac_parser_t::stride`

**5.436.4.6 char\_data** `MHAParser::string_t` `testplugin::ac_parser_t::char_data`

**5.436.4.7 int\_data** `MHAParser::vint_t` `testplugin::ac_parser_t::int_data`

**5.436.4.8 float\_data** `MHAParser::vfloat_t` `testplugin::ac_parser_t::float_data`

**5.436.4.9 complex\_data** `MHAParser::vcomplex_t testplugin::ac_parser_t::complex_` ↵  
data

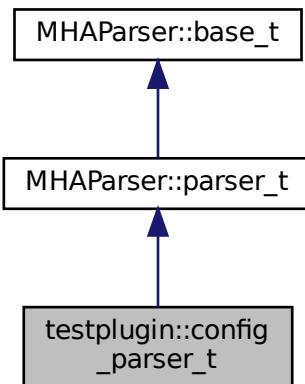
**5.436.4.10 patchbay** `MHAEvents::patchbay_t< ac_parser_t> testplugin::ac_parser_t::patchbay`

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

- `testplugin.cpp`

## 5.437 testplugin::config\_parser\_t Class Reference

Inheritance diagram for testplugin::config\_parser\_t:



### Public Member Functions

- `void setlock (const bool &b)`
- `config_parser_t ()`
- `mhaconfig_t get () const`
- `void set ( mhaconfig_t c)`

## Public Attributes

- `MHAParser::int_t channels`
- `MHAParser::kw_t domain`
- `MHAParser::int_t fragsize`
- `MHAParser::int_t wndlen`
- `MHAParser::int_t fftlen`
- `MHAParser::float_t srate`

## Additional Inherited Members

### 5.437.1 Constructor & Destructor Documentation

**5.437.1.1 config\_parser\_t()** `testplugin::config_parser_t::config_parser_t ( ) [inline]`

### 5.437.2 Member Function Documentation

**5.437.2.1 setlock()** `void testplugin::config_parser_t::setlock ( const bool & b ) [inline]`

**5.437.2.2 get()** `mhaconfig_t testplugin::config_parser_t::get ( ) const [inline]`

**5.437.2.3 set()** `void testplugin::config_parser_t::set ( mhaconfig_t c ) [inline]`

### 5.437.3 Member Data Documentation

**5.437.3.1 channels** `MHAParser::int_t testplugin::config_parser_t::channels`

**5.437.3.2 domain** `MHAParser::kw_t testplugin::config_parser_t::domain`

**5.437.3.3 fragsize** `MHAParser::int_t testplugin::config_parser_t::fragsize`

**5.437.3.4 wndlen** `MHAParser::int_t testplugin::config_parser_t::wndlen`

**5.437.3.5 fftlen** `MHAParser::int_t testplugin::config_parser_t::fftlens`

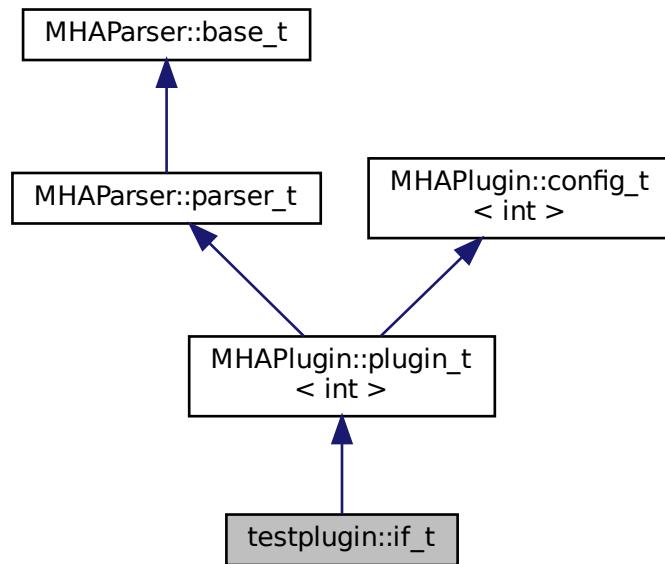
**5.437.3.6 srate** `MHAParser::float_t testplugin::config_parser_t::srate`

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

- `testplugin.cpp`

## 5.438 testplugin::if\_t Class Reference

Inheritance diagram for testplugin::if\_t:



### Public Member Functions

- `if_t ( algo_comm_t iac, const std::string &configured_name)`
- `mha_spec_t * process ( mha_spec_t *s_in)`
- `mha_wave_t * process ( mha_wave_t *s_in)`
- `void prepare ( mhaconfig_t &)`

### Private Member Functions

- `void test_prepare ()`
- `void test_process ()`

### Private Attributes

- `config_parser_t config_in`
- `config_parser_t config_out`
- `ac_parser_t ac`
- `signal_parser_t signal`
- `MHPParser::bool_t _prepare`
- `MHAEvents::patchbay_t< if_t > patchbay`
- `MHPParser::mhapluginloader_t plug`

## Additional Inherited Members

### 5.438.1 Constructor & Destructor Documentation

```
5.438.1.1 if_t() testplugin::if_t::if_t (
    algo_comm_t iac,
    const std::string & configured_name )
```

### 5.438.2 Member Function Documentation

```
5.438.2.1 process() [1/2] mha_spec_t* testplugin::if_t::process (
    mha_spec_t * s_in ) [inline]
```

```
5.438.2.2 process() [2/2] mha_wave_t* testplugin::if_t::process (
    mha_wave_t * s_in ) [inline]
```

```
5.438.2.3 prepare() void testplugin::if_t::prepare (
    mhaconfig_t & ) [inline], [virtual]
```

Implements **MHAPlugin::plugin\_t< int >** (p. 1149).

```
5.438.2.4 test_prepare() void testplugin::if_t::test_prepare ( ) [private]
```

```
5.438.2.5 test_process() void testplugin::if_t::test_process ( ) [private]
```

### 5.438.3 Member Data Documentation

**5.438.3.1 config\_in** `config_parser_t` testplugin::if\_t::config\_in [private]

**5.438.3.2 config\_out** `config_parser_t` testplugin::if\_t::config\_out [private]

**5.438.3.3 ac** `ac_parser_t` testplugin::if\_t::ac [private]

**5.438.3.4 signal** `signal_parser_t` testplugin::if\_t::signal [private]

**5.438.3.5 \_prepare** `MHAParser::bool_t` testplugin::if\_t::\_prepare [private]

**5.438.3.6 patchbay** `MHAEEvents::patchbay_t< if_t>` testplugin::if\_t::patchbay [private]

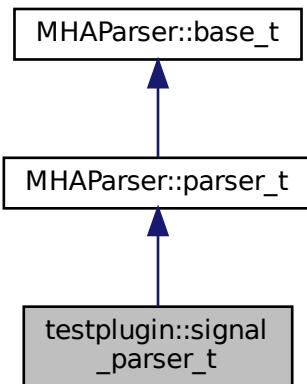
**5.438.3.7 plug** `MHAParser::mhapluginloader_t` testplugin::if\_t::plug [private]

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

- **testplugin.cpp**

## 5.439 testplugin::signal\_parser\_t Class Reference

Inheritance diagram for testplugin::signal\_parser\_t:



### Public Member Functions

- **signal\_parser\_t ()**

### Public Attributes

- **MHAParser::mfloat\_t input\_wave**
- **MHAParser::mcomplex\_t input\_spec**
- **MHAParser::mfloat\_mon\_t output\_wave**
- **MHAParser::mcomplex\_mon\_t output\_spec**

### Additional Inherited Members

#### 5.439.1 Constructor & Destructor Documentation

##### 5.439.1.1 **signal\_parser\_t()** testplugin::signal\_parser\_t::signal\_parser\_t ( ) [inline]

### 5.439.2 Member Data Documentation

**5.439.2.1 input\_wave** `MHAParser::mfloat_t testplugin::signal_parser_t::input_wave`

**5.439.2.2 input\_spec** `MHAParser::mcomplex_t testplugin::signal_parser_t::input_spec`

**5.439.2.3 output\_wave** `MHAParser::mfloat_mon_t testplugin::signal_parser_t::output_wave`

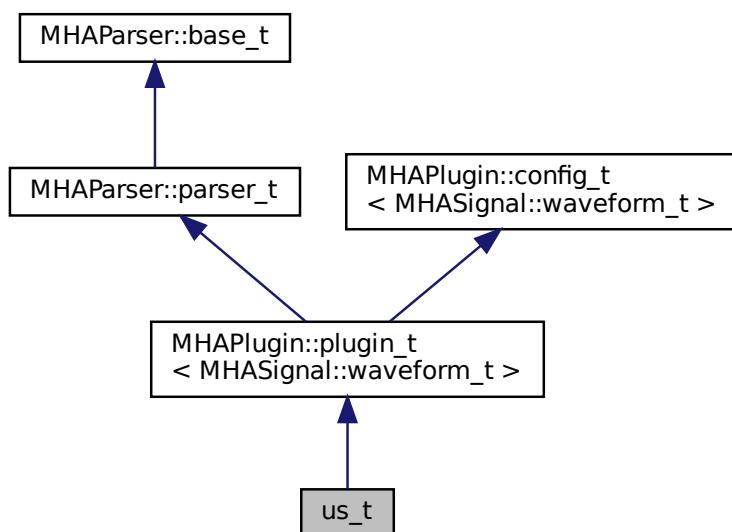
**5.439.2.4 output\_spec** `MHAParser::mcomplex_mon_t testplugin::signal_parser_t::output_spec`

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

- `testplugin.cpp`

## 5.440 us\_t Class Reference

Inheritance diagram for us\_t:



## Public Member Functions

- `us_t ( algo_comm_t iac, const std::string &configured_name)`
- `mha_wave_t * process ( mha_wave_t *)`
- `void prepare ( mhaconfig_t &)`
- `void release ()`

## Private Attributes

- `MHAParser::int_t ratio`
- `MHAFilter::iir_filter_t antialias`

## Additional Inherited Members

### 5.440.1 Constructor & Destructor Documentation

```
5.440.1.1 us_t() us_t::us_t (
    algo_comm_t iac,
    const std::string & configured_name )
```

### 5.440.2 Member Function Documentation

```
5.440.2.1 process() mha_wave_t * us_t::process (
    mha_wave_t * s )
```

```
5.440.2.2 prepare() void us_t::prepare (
    mhaconfig_t & cf ) [virtual]
```

Implements `MHAPlugin::plugin_t< MHASignal::waveform_t >` (p. [1149](#)).

**5.440.2.3 release()** void us\_t::release ( ) [virtual]

Reimplemented from **MHAParser::parser\_t**< **MHASignal::waveform\_t** > (p. 1150).

**5.440.3 Member Data Documentation****5.440.3.1 ratio** MHAParser::int\_t us\_t::ratio [private]**5.440.3.2 antialias** MHAFilter::iir\_filter\_t us\_t::antialias [private]

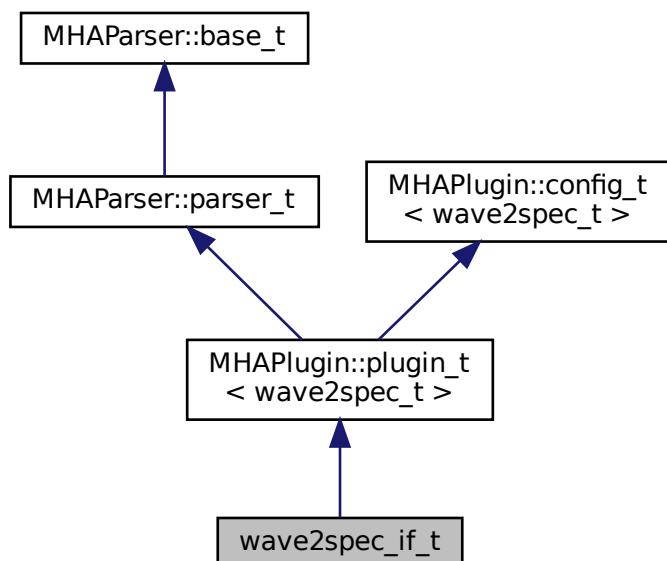
The documentation for this class was generated from the following file:

- **upsample.cpp**

**5.441 wave2spec\_if\_t Class Reference**

Plugin wave2spec interface class, uses **wave2spec\_t** (p. 1492) as runtime configuration.

Inheritance diagram for wave2spec\_if\_t:



## Public Member Functions

- **wave2spec\_if\_t** (**algo\_comm\_t** iac, const std::string &configured\_name)  
*Constructor of wave2spec plugin, sets up configuration variables and callbacks.*
- **void prepare** (**mhaconfig\_t** &t)  
*prepare for signal processing*
- **void release** ()  
*Unprepare signal processing.*
- **void process** (**mha\_wave\_t** \*wave\_in, **mha\_spec\_t** \*\*sout)  
*processing callback used for domain transformation*
- **void process** (**mha\_wave\_t** \*wave\_in, **mha\_wave\_t** \*\*sout)  
*processing callback used if output of original waveform is requested.*

## Private Member Functions

- **void update** ()  
*Create a new runtime configuration from configuration parameters when the plugin is prepared, or when the window position or other window parameters change.*
- **void setlock** (bool b)  
*Lock/Unlock all configuration variables.*

## Private Attributes

- **MHAParser::int\_t nfft**  
*FFT length selector.*
- **MHAParser::int\_t nwnd**  
*Window length selector.*
- **MHAParser::float\_t wndpos**  
*Window position selector.*
- **windowselector\_t window\_config**
- **MHAParser::bool\_t strict\_window\_ratio**  
*Switch to disallow window sizes that are not a multiple of the fragsize a by power of two.*
- **MHAParser::bool\_t return\_wave**  
*Switch to select return domain.*
- **std::string algo**  
*configured name this plugin, used to name the AC variables*
- **MHAParser::vfloat\_mon\_t zeropadding**

## Additional Inherited Members

### 5.441.1 Detailed Description

Plugin wave2spec interface class, uses **wave2spec\_t** (p. 1492) as runtime configuration.

## 5.441.2 Constructor & Destructor Documentation

**5.441.2.1 `wave2spec_if_t()`** `wave2spec_if_t::wave2spec_if_t ( algo_comm_t iac, const std::string & configured_name )`

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

## 5.441.3 Member Function Documentation

**5.441.3.1 `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. [1149](#)).

**5.441.3.2 `release()`** `void wave2spec_if_t::release ( ) [virtual]`

Unprepare signal processing.

Reimplemented from **MHAPlugIn::plugin\_t< wave2spec\_t >** (p. [1150](#)).

### 5.441.3.3 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

### 5.441.3.4 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)

### 5.441.3.5 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

<b>MHA_Error</b> (p. <a href="#">760</a> )	if the configuration change is not compatible with the current input and FFT length constraints.
--	--

**5.441.3.6 setlock()** void wave2spec\_if\_t::setlock ( bool b ) [private]

Lock/Unlock all configuration variables.

**Parameters**

<b>b</b>	Desired lock state
----------	--------------------

#### 5.441.4 Member Data Documentation

**5.441.4.1 nfft** `MHAParser::int_t` wave2spec\_if\_t::nfft [private]

FFT length selector.

**5.441.4.2 nwnd** `MHAParser::int_t` wave2spec\_if\_t::nwnd [private]

Window length selector.

**5.441.4.3 wndpos** `MHAParser::float_t` wave2spec\_if\_t::wndpos [private]

Window position selector.

**5.441.4.4 window\_config** `windowselector_t` wave2spec\_if\_t::window\_config [private]

**5.441.4.5 strict\_window\_ratio** `MHAParser::bool_t` wave2spec\_if\_t::strict\_window\_ratio [private]

Switch to disallow window sizes that are not a multiple of the fragsize a by power of two.

#### **5.441.4.6 return\_wave    `MHAParser::bool_t wave2spec_if_t::return_wave` [private]**

Switch to select return domain.

#### **5.441.4.7 algo    `std::string wave2spec_if_t::algo` [private]**

configured name this plugin, used to name the AC variables

#### **5.441.4.8 zeropadding    `MHAParser::vfloat_mon_t wave2spec_if_t::zeropadding` [private]**

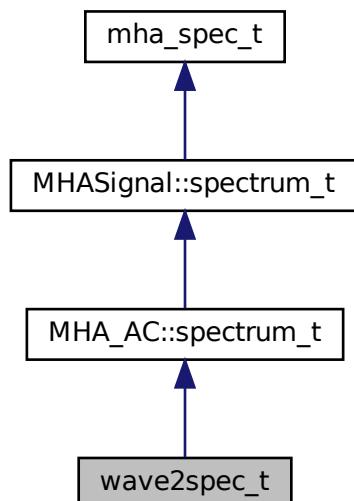
The documentation for this class was generated from the following files:

- `wave2spec.hh`
- `wave2spec.cpp`

### **5.442 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

### 5.442.1 Detailed Description

Runtime configuration class for plugin wave2spec.

Manages window shift, windowing, zero-padding, and FFT. Inserts current window shape and current STFT spectrum into AC space.

### 5.442.2 Constructor & Destructor Documentation

```
5.442.2.1 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

<code>nfft</code>	FFT length
<code>nwnd_</code>	window length in samples
<code>nwndshift_</code>	window shift (hop size) in samples
<code>nch</code>	number of audio channels
<code>wndpos</code>	for cases <code>nfft &gt; nwnd_</code> , 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
<code>window</code>	Analysis window shape
<code>ac</code>	algorithm communication storage accessor
<code>algo</code>	configured name of this plugin, used to name the AC variables published by wave2spec

**5.442.2.2 ~wave2spec\_t()** `wave2spec_t::~wave2spec_t ( )`

Destructor removes AC variables from AC space and deallocates memory.

**5.442.3 Member Function Documentation****5.442.3.1 publish\_ac\_variables()** `void wave2spec_t::publish_ac_variables ( )`

Insert two AC variables into AC space:

- <configured\_name>: Contains the current STFT spectrum.
- <configured\_name>\_wnd: Contains the window shape as individual weights.

**5.442.3.2 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.

**5.442.3.3 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.
--------------	--

```
5.442.3.4 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.

**5.442.4 Member Data Documentation**

```
5.442.4.1 nwnd unsigned int wave2spec_t::nwnd [private]
```

window length

```
5.442.4.2 nwndshift unsigned int wave2spec_t::nwndshift [private]
```

window shift or hop size

**5.442.4.3 ft** **mha\_fft\_t** wave2spec\_t::ft [private]

FFT instance used for transformation.

**5.442.4.4 npad1** unsigned int wave2spec\_t::npad1 [private]

length of zero padding before window

**5.442.4.5 npad2** unsigned int wave2spec\_t::npad2 [private]

length of zero padding after window

**5.442.4.6 window** **MHAWindow::base\_t** wave2spec\_t::window [private]

Analysis window.

**5.442.4.7 calc\_in** **MHASignal::waveform\_t** wave2spec\_t::calc\_in [private]

waveform buffer with FFT length samples per channel

**5.442.4.8 in\_buf** **MHASignal::waveform\_t** wave2spec\_t::in\_buf [private]

waveform buffer with window length samples per channel

**5.442.4.9 spec\_in** **MHASignal::spectrum\_t** wave2spec\_t::spec\_in [private]

spectrum buffer containing only the positive frequency bins

#### **5.442.4.10 ac\_wndshape\_name std::string wave2spec\_t::ac\_wndshape\_name [private]**

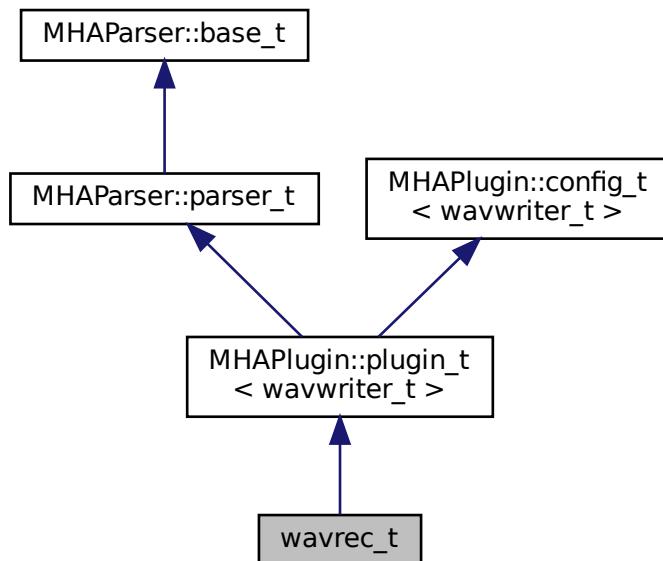
name of window shape AC variable

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

- **wave2spec.hh**
- **wave2spec.cpp**

### **5.443 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 ( algo\_comm\_t iac, const std::string &configured\_name)**

#### **Private Member Functions**

- **void start\_new\_session ()**

## Private Attributes

- `MHAParser::bool_t record`
- `MHAParser::int_t fifolen`
- `MHAParser::int_t minwrite`
- `MHAParser::string_t prefix`
- `MHAParser::bool_t use_date`
- `MHAParser::kw_t output_sample_format`
- `MHAEvents::patchbay_t< wavrec_t > patchbay`

## Additional Inherited Members

### 5.443.1 Constructor & Destructor Documentation

```
5.443.1.1 wavrec_t() wavrec_t::wavrec_t (
    algo_comm_t iac,
    const std::string & configured_name )
```

### 5.443.2 Member Function Documentation

```
5.443.2.1 process() mha_wave_t * wavrec_t::process (
    mha_wave_t * s )
```

```
5.443.2.2 prepare() void wavrec_t::prepare (
    mhaconfig_t & cf ) [virtual]
```

Implements `MHAParser::parser_t< wavwriter_t >` (p. 1149).

```
5.443.2.3 release() void wavrec_t::release ( ) [virtual]
```

Reimplemented from `MHAParser::parser_t< wavwriter_t >` (p. 1150).

**5.443.2.4 start\_new\_session()** void wavrec\_t::start\_new\_session ( ) [private]

### 5.443.3 Member Data Documentation

**5.443.3.1 record** MHAParser::bool\_t wavrec\_t::record [private]

**5.443.3.2 fifolen** MHAParser::int\_t wavrec\_t::fifolen [private]

**5.443.3.3 minwrite** MHAParser::int\_t wavrec\_t::minwrite [private]

**5.443.3.4 prefix** MHAParser::string\_t wavrec\_t::prefix [private]

**5.443.3.5 use\_date** MHAParser::bool\_t wavrec\_t::use\_date [private]

**5.443.3.6 output\_sample\_format** MHAParser::kw\_t wavrec\_t::output\_sample\_format [private]

**5.443.3.7 patchbay** MHAEEvents::patchbay\_t< wavrec\_t> wavrec\_t::patchbay [private]

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

- **wavrec.cpp**

## 5.444 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`

### 5.444.1 Constructor & Destructor Documentation

---

**5.444.1.1 `wavwriter_t()`** `wavwriter_t::wavwriter_t (`  
`bool active,`  
`const mhaconfig_t & cf,`  
`unsigned int fifosize,`  
`unsigned int minwrite,`  
`const std::string & prefix,`  
`bool use_date,`  
`const std::string & format_name_ )`

**5.444.1.2 `~wavwriter_t()`** `wavwriter_t::~wavwriter_t ( )`

## 5.444.2 Member Function Documentation

**5.444.2.1 `process()`** `void wavwriter_t::process (`  
`mha_wave_t * s )`

**5.444.2.2 `exit_request()`** `void wavwriter_t::exit_request ( )`

**5.444.2.3 `write_thread()` [1/2]** `static void* wavwriter_t::write_thread (`  
`void * this_ ) [inline], [static], [private]`

**5.444.2.4 `write_thread()` [2/2]** `void wavwriter_t::write_thread ( ) [private]`

**5.444.2.5 `create_soundfile()`** `void wavwriter_t::create_soundfile (`  
`const std::string & prefix,`  
`bool use_date ) [private]`

**5.444.2.6 `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**

<i>sf_info</i>	Destination sf_info struct for the format
----------------	---

**Exceptions**

<i>MHA_Error</i> (p. 760)	If no sample format of name format_name is offered by libsndfile
---------------------------	--

**5.444.3 Member Data Documentation****5.444.3.1 close\_session** std::atomic<bool> wavwriter\_t::close\_session [private]**5.444.3.2 act\_** bool wavwriter\_t::act\_ [private]**5.444.3.3 cf\_** mhaconfig\_t wavwriter\_t::cf\_ [private]**5.444.3.4 sf** SNDFILE\* wavwriter\_t::sf [private]**5.444.3.5 fifo** mha\_fifo\_if\_t< mha\_real\_t> wavwriter\_t::fifo [private]**5.444.3.6 minw\_** unsigned int wavwriter\_t::minw\_ [private]

**5.444.3.7 writethread** pthread\_t wavwriter\_t::writethread [private]

**5.444.3.8 data** float\* wavwriter\_t::data [private]

**5.444.3.9 format\_name** std::string wavwriter\_t::format\_name [private]

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

- **wavrec.cpp**

## 5.445 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.*

### 5.445.1 Detailed Description

Runtime config class for windnoise plugin.

Computes power spectra of incoming STFT spectra, smoothes the power spectrum over time by low-pass filtering the intensities of each bin over time, then detects wind noise presence by comparing intensity at low frequency bins to broadband intensity.

### 5.445.2 Constructor & Destructor Documentation

```
5.445.2.1 cfg_t() cfg_t::cfg_t (
    const mhaconfig_t & signal_info,
    bool UseChannel_LF_attenuation,
    float tau_Lowpass,
    float LowPassCutOffFrequency,
    float LowPassFraction_dB,
    float LowPassWindGain_dB )
```

constructor translates configuration variables to runtime config

### 5.445.3 Member Function Documentation

```
5.445.3.1 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

<b>MHA_Error</b> (p. <a href="#">760</a> )	if windnoise_indicators.size() != signal.num_channels.
--	--

```
5.445.3.2 update_PSD_Lowpass() void cfg_t::update_PSD_Lowpass (
    const mha_spec_t * signal )
```

Low-pass filters the power spectrum.

```
5.445.3.3 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.

**5.445.3.4 remapping()** `int cfg_t::remapping (`  
`const std::vector< float > & lowpass_quotient )`

**5.445.3.5 compensation()** `void cfg_t::compensation (`  
`mha_spec_t * signal,`  
`int best_signal_channel_index )`

### 5.445.4 Member Data Documentation

**5.445.4.1 UseChannel\_LF\_attenuation** `bool windnoise::cfg_t::UseChannel_LF_attenuation`  
`= false`

FIXME: documentation for UseChannel\_LF\_attenuation.

**5.445.4.2 alpha\_Lowpass** `float windnoise::cfg_t::alpha_Lowpass = 0`

Filter coefficient for low-pass filtering each bin in the power spectrum with a first-order recursive low-pass filter.

**5.445.4.3 FrequencyBinLowPass** `unsigned windnoise::cfg_t::FrequencyBinLowPass = 0`

Only smoothed power spectrum bins < FrequencyBinLowPass are added to the low-pass intensity.

**5.445.4.4 LowPassFraction** `float windnoise::cfg_t::LowPassFraction = 1`

The wind noise detection threshold: We have wind noise if `lowFreqIntensity / broadBandIntensity > LowPassFraction`.

**5.445.4.5 LowPassWindGain** `float windnoise::cfg_t::LowPassWindGain = 1`

FIXME: documentation for LowPassWindGain.

**5.445.4.6 PSD\_Lowpass** `MHASignal::waveform_t windnoise::cfg_t::PSD_Lowpass`

The smoothed-over-time power spectrum.

**5.445.4.7 powspec** `MHASignal::waveform_t windnoise::cfg_t::powspec`

Temporary storage for the power spectrum of the current input spectrum.

Only needed to hold the newest squared magnitudes until they are filtered into PSD\_Lowpass.

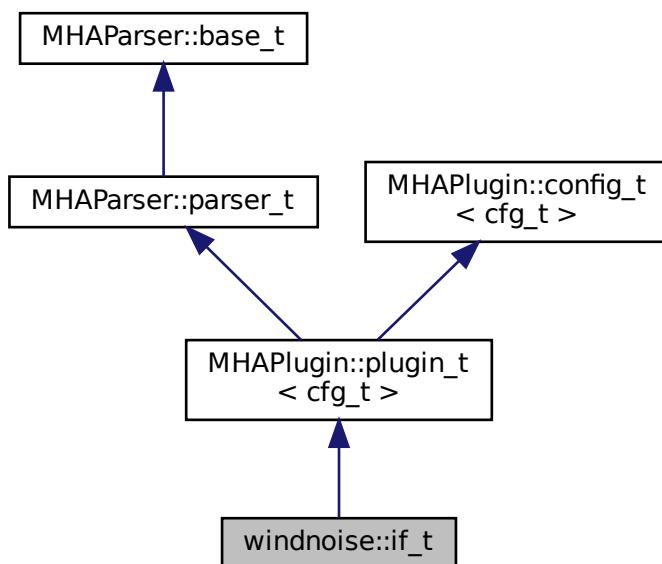
The documentation for this class was generated from the following files:

- `windnoise.hh`
- `windnoise.cpp`

## 5.446 windnoise::if\_t Class Reference

interface class for windnoise plugin

Inheritance diagram for windnoise::if\_t:



### Public Member Functions

- **`if_t ( algo_comm_t iac, const std::string &configured_name)`**  
*Constructor instantiates one windnoise plugin.*
- **`void prepare ( mhaconfig_t &signal_info) override`**  
*Prepare windnoise plugin for signal processing.*
- **`void release (void) override`**  
*Nothing needs to be deallocated on release.*
- **`mha_spec_t * process ( mha_spec_t *signal)`**  
*signal processing, delegates to `cfg_t::process` (p. 1506)*
- **`void update (void)`**  
*update runtime config when configuration parameters have changed*
- **`void insert ()`**  
*inserts the windnoise detection vector into AC space*

## Public Attributes

- **MHAParser::bool\_t UseChannel\_LF\_attenuation**
- **MHAParser::float\_t tau\_Lowpass**
- **MHAParser::float\_t LowPassCutOffFrequency**
- **MHAParser::float\_t LowPassFraction**
- **MHAParser::float\_t LowPassWindGain**
- **MHAParser::kw\_t WindNoiseDetector**
- **MHAParser::vint\_mon\_t detected**
- **MHAParser::vfloat\_mon\_t lowpass\_quotient**
- const std::string **detected\_acname**  
*Name of AC variable mirroring the configuration monitor variable "detector".*
- const std::string **lowpass\_quotient\_acname**  
*Name of AC variable mirroring the configuration monitor variable "lowpass\_quotient".*

## Private Attributes

- **MHAEvents::patchbay\_t< if\_t > patchbay**  
*The Event connector.*

## Additional Inherited Members

### 5.446.1 Detailed Description

interface class for windnoise plugin

### 5.446.2 Constructor & Destructor Documentation

```
5.446.2.1 if_t() windnoise::if_t::if_t (
    algo_comm_t iac,
    const std::string & configured_name )
```

Constructor instantiates one windnoise plugin.

### 5.446.3 Member Function Documentation

```
5.446.3.1 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. 1149).

**5.446.3.2 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. 1150).

**5.446.3.3 process()** `mha_spec_t * windnoise::if_t::process ( mha_spec_t * signal )`

signal processing, delegates to **cfg\_t::process** (p. 1506)

**5.446.3.4 update()** `void windnoise::if_t::update ( void )`

update runtime config when configuration parameters have changed

**5.446.3.5 insert()** `void windnoise::if_t::insert ( ) [inline]`

inserts the windnoise detection vector into AC space

## 5.446.4 Member Data Documentation

**5.446.4.1 patchbay** `MHAEVENTS::patchbay_t< if_t> windnoise::if_t::patchbay [private]`

The Event connector.

**5.446.4.2 UseChannel\_LF\_attenuation** `MHAPARSER::bool_t windnoise::if_t::Use←  
Channel_LF_attenuation`**5.446.4.3 tau\_Lowpass** `MHAPARSER::float_t windnoise::if_t::tau_Lowpass`**5.446.4.4 LowPassCutOffFrequency** `MHAPARSER::float_t windnoise::if_t::LowPass←  
CutOffFrequency`**5.446.4.5 LowPassFraction** `MHAPARSER::float_t windnoise::if_t::LowPassFraction`**5.446.4.6 LowPassWindGain** `MHAPARSER::float_t windnoise::if_t::LowPassWindGain`**5.446.4.7 WindNoiseDetector** `MHAPARSER::kw_t windnoise::if_t::WindNoiseDetector`**5.446.4.8 detected** `MHAPARSER::vint_mon_t windnoise::if_t::detected`**5.446.4.9 lowpass\_quotient** `MHAPARSER::vfloat_mon_t windnoise::if_t::lowpass_←  
quotient`

**5.446.4.10 detected\_acname** const std::string windnoise::if\_t::detected\_acname

Name of AC variable mirroring the configuration monitor variable "detector".

Usually "windnoise\_detected".

**5.446.4.11 lowpass\_quotient\_acname** const std::string windnoise::if\_t::lowpass\_<quotient\_acname>

Name of AC variable mirroring the configuration monitor variable "lowpass\_quotient".

Usually "windnoise\_lowpass\_quotient".

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

- **windnoise.hh**
- **windnoise.cpp**

## 5.447 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. [1515](#))*
- **MHAParser::kw\_t wndtype**  
*parser variable for window type*
- **MHAParser::float\_t wndexp**  
*parser variable for window exponent*
- **MHAParser::vfloat\_t userwnd**  
*parser variable for user window samples to use*
- **MHAEvents::patchbay\_t< windowselector\_t > patchbay**  
*patchbay to watch for changes for the parser variables*

### 5.447.1 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.

### 5.447.2 Constructor & Destructor Documentation

**5.447.2.1 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"
---------------------------	--

**5.447.2.2 ~windowselector\_t()** `windowselector_t::~windowselector_t ( )`

destructor frees window data that were allocated

**5.447.3 Member Function Documentation****5.447.3.1 get\_window\_data()** `const MHAWindow::base_t & windowselector_t::get_<→`  
`window_data (`  
`unsigned length )`

re-computes the window if required.

**Parameters**

<code>length</code>	the desired window length in samples return the window's samples as a constref to <b>MHAWindow::base_t</b> (p. 1287) instance. The referenced instance lives until the window parameters are changed, or this <b>windowselector_t</b> (p. 1513) instance is destroyed.
---------------------	--

**5.447.3.2 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**

<code>p</code>	The configuration parser where to insert the window parameters. E.g. the plugin wave2spec's interface class.
----------------	--

**5.447.3.3 setlock()** `void windowselector_t::setlock (`  
`bool b_ )`

Lock/Unlock variables.

#### Parameters

$b \leftarrow$	Desired lock state
$\_ \leftarrow$	

**5.447.3.4 invalidate\_window\_data()** `void windowselector_t::invalidate_window_data ( ) [private]`

invalidates any allocated window samples.

**5.447.3.5 update\_parser()** `void windowselector_t::update_parser ( ) [private]`

invoked when a parser parameter changes.

Calls **invalidate\_window\_data()** (p. 1516) and emits the updated event.

### 5.447.4 Member Data Documentation

**5.447.4.1 updated** `MHAEevents::emitter_t windowselector_t::updated`

A collector event that fires when any of the window parameters managed here is written to.

**5.447.4.2 wnd** `MHAWindow::base_t* windowselector_t::wnd [private]`

Storage for the window data returned by **get\_window\_data()** (p. 1515)

**5.447.4.3 wndtype** `MHAParser::kw_t` `windowselector_t::wndtype` [private]

parser variable for window type

**5.447.4.4 wndexp** `MHAParser::float_t` `windowselector_t::wndexp` [private]

parser variable for window exponent

**5.447.4.5 userwnd** `MHAParser::vfloat_t` `windowselector_t::userwnd` [private]

parser variable for user window samples to use

**5.447.4.6 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`

## 6 File Documentation

### 6.1 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

- **ac2lsl**

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

## Variables

- const std::map< int, type\_info > **ac2lsl::types**

## 6.2 ac2osc.cpp File Reference

### Classes

- class **ac2osc\_t**

*Plugin class of the ac2osc plugin.*

## 6.3 ac2wave.cpp File Reference

### Classes

- class **ac2wave\_t**
- class **ac2wave\_if\_t**

## 6.4 ac\_monitor\_type.cpp File Reference

## 6.5 ac\_monitor\_type.hh File Reference

### Classes

- class **acmon::ac\_monitor\_t**

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

## Namespaces

- **acmon**

*Namespace for displaying ac variables as parser monitors.*

## 6.6 ac\_mul.cpp File Reference

## 6.7 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.*

#### 6.7.1 Enumeration Type Documentation

##### 6.7.1.1 **arg\_type\_t** `enum arg_type_t`

Indicates whether the factors of the product are real or complex valued.

###### Enumerator

<b>ARG_RR</b>	Both factors are real.
<b>ARG_RC</b>	First factor is real, second is complex.
<b>ARG_CR</b>	First factor is complex, second is real.
<b>ARG_CC</b>	Both factors are complex.

##### 6.7.1.2 **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.

## 6.8 ac\_proc.cpp File Reference

### Classes

- class `ac_proc::interface_t`

### Namespaces

- `ac_proc`

## 6.9 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)`

### 6.9.1 Macro Definition Documentation

#### 6.9.1.1 PATCH\_VAR `#define PATCH_VAR(`

```
var ) patchbay.connect(&var.valuechanged, this, & acConcat_wave::update_cfg)
```

#### 6.9.1.2 INSERT\_PATCH `#define INSERT_PATCH(`

```
var ) insert_member(var); PATCH_VAR(var)
```

## 6.10 acConcat\_wave.h File Reference

### Classes

- class `acConcat_wave_config`
- class `acConcat_wave`

## 6.11 acmon.cpp File Reference

### Classes

- class `acmon::acmon_t`

### Namespaces

- **acmon**  
*Namespace for displaying ac variables as parser monitors.*

## 6.12 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)`

### 6.12.1 Macro Definition Documentation

#### 6.12.1.1 `PATCH_VAR` #define PATCH\_VAR(

```
var ) patchbay.connect(&var.valuechanged, this, & acPooling_wave::update_cfg)
```

#### 6.12.1.2 `INSERT_PATCH` #define INSERT\_PATCH(

```
var ) insert_member(var); PATCH_VAR(var)
```

## 6.13 acPooling\_wave.h File Reference

### Classes

- class **acPooling\_wave\_config**
- class **acPooling\_wave**

## 6.14 acrec.cpp File Reference

## 6.15 acrec.hh File Reference

### Classes

- class **plugins::hoertech::acrec::acwriter\_t**  
*acwriter\_t* (p. 1378) decouples signal processing from writing to disk.
- class **plugins::hoertech::acrec::acrec\_t**  
*Plugin interface class of plugin acrec.*

### Namespaces

- **plugins**
- **plugins::hoertech**
- **plugins::hoertech::acrec**

### Functions

- std::string **plugins::hoertech::acrec::to\_iso8601** (time\_t tm)

## 6.16 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"

### 6.16.1 Macro Definition Documentation

#### 6.16.1.1 **ACSAVE\_FMT\_TXT** #define ACSAVE\_FMT\_TXT 0

#### 6.16.1.2 **ACSAVE\_SFMT\_TXT** #define ACSAVE\_SFMT\_TXT "txt"

#### 6.16.1.3 **ACSAVE\_FMT\_MAT4** #define ACSAVE\_FMT\_MAT4 1

#### 6.16.1.4 **ACSAVE\_SFMT\_MAT4** #define ACSAVE\_SFMT\_MAT4 "mat4"

#### 6.16.1.5 **ACSAVE\_FMT\_M** #define ACSAVE\_FMT\_M 2

#### 6.16.1.6 **ACSAVE\_SFMT\_M** #define ACSAVE\_SFMT\_M "m"

## 6.17 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)

### 6.17.1 Macro Definition Documentation

**6.17.1.1 PATCH\_VAR** #define PATCH\_VAR(  
    var ) patchbay.connect(&var.valuechanged, this, & **acSteer**::**update\_cfg**)

**6.17.1.2 INSERT\_PATCH** #define INSERT\_PATCH(  
    var ) **insert\_member**(var); **PATCH\_VAR**(var)

## 6.18 acSteer.h File Reference

### Classes

- class **acSteer\_config**
- class **acSteer**

## 6.19 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)

### 6.19.1 Macro Definition Documentation

**6.19.1.1 PATCH\_VAR** #define PATCH\_VAR(  
    var ) patchbay.connect(&var.valuechanged, this, & **acTransform\_wave**::  
    ::update\_cfg)

**6.19.1.2 INSERT\_PATCH** #define INSERT\_PATCH(  
    var ) insert\_member(var); **PATCH\_VAR**(var)

## 6.20 acTransform\_wave.h File Reference

### Classes

- class **acTransform\_wave\_config**
- class **acTransform\_wave**

## 6.21 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)

## 6.21.1 Macro Definition Documentation

**6.21.1.1 PATCH\_VAR** #define PATCH\_VAR(  
    var ) patchbay.connect(&var.valuechanged, this, & **adaptive\_feedback\_canceller**::  
    ::update\_cfg)

**6.21.1.2 INSERT\_PATCH**

```
#define INSERT_PATCH(
```

```
    var )  insert_member(var);  PATCH_VAR(var)
```

**6.21.2 Function Documentation****6.21.2.1 make\_friendly\_number\_by\_limiting()**

```
void make_friendly_number_by_limiting(
```

```
    mha_real_t & x )  [inline]
```

**6.22 adaptive\_feedback\_canceller.h File Reference****Classes**

- class **adaptive\_feedback\_canceller\_config**
- class **adaptive\_feedback\_canceller**

**6.23 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)`

### 6.23.1 Macro Definition Documentation

```
6.23.1.1 DEBUG #define DEBUG( x ) std::cerr << __FILE__ << ":" << __LINE__ << " " #x "=" << x << std::endl
```

## 6.24 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)`

## 6.24.1 Function Documentation

### 6.24.1.1 **adm\_fir\_lp()** `MHASignal::waveform_t* adm_fir_lp (`

```
    unsigned int fs,
    unsigned f_pass,
    unsigned int f_stop,
    unsigned int order )
```

### 6.24.1.2 **adm\_fir\_decomb()** `MHASignal::waveform_t* adm_fir_decomb (`

```
    unsigned int fs,
    float dist_m,
    unsigned int order )
```

## 6.25 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

## 6.26 altconfig.cpp File Reference

### 6.27 altconfig.hh File Reference

## Classes

- class **altconfig\_t**  
*Single class implementing plugin altconfig.*

## Macros

- #define **MHAPLUGIN\_OVERLOAD\_OUTDOMAIN**

### 6.27.1 Macro Definition Documentation

**6.27.1.1 MHAPLUGIN\_OVERLOAD\_OUTDOMAIN** #define MHAPLUGIN\_OVERLOAD\_OUTDOM←  
AIN

## 6.28 altplugs.cpp File Reference

## Classes

- class **mhaplug\_cfg\_t**
- class **altplugs\_t**

## Macros

- #define **MHAPLUGIN\_OVERLOAD\_OUTDOMAIN**

## 6.28.1 Macro Definition Documentation

**6.28.1.1 MHAPLUGIN\_OVERLOAD\_OUTDOMAIN** #define MHAPLUGIN\_OVERLOAD\_OUTDOM↔  
AIN

## 6.29 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)

### 6.29.1 Function Documentation

**6.29.1.1 strdom()** std::string strdom ( mha\_domain\_t d )

**6.29.1.2 print\_ac()** void print\_ac ( MHAKernel::algo\_comm\_class\_t & ac,  
std::string txt )

**6.29.1.3 document\_plugin()** int document\_plugin ( MHAKernel::algo\_comm\_class\_t & ac,  
PluginLoader::mhapluginloader\_t & load,  
int argc,  
char \*\* argv )

**6.29.1.4 document\_io\_plugin()** void document\_io\_plugin ( char \* lib\_name )

**6.29.1.5 main()** int main ( int argc, char \*\* argv )

## 6.30 analysispath.cpp File Reference

### Classes

- class **analysepath\_t**
- class **plug\_t**
- class **analysispath\_if\_t**

### Functions

- static void \* **thread\_start** (void \*instance)

## 6.30.1 Function Documentation

**6.30.1.1 thread\_start()** static void\* thread\_start ( void \* instance ) [static]

## 6.31 attenuate20.cpp File Reference

### Classes

- class **attenuate20\_t**

## 6.32 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)

#### 6.32.1 Macro Definition Documentation

```
6.32.1.1 DEBUG #define DEBUG(
    x ) std::cerr << __FILE__ << ":" << __LINE__ << " " #x "=" << x <<
std::endl
```

## 6.33 auditory\_profile.cpp File Reference

### 6.34 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)*

## 6.35 browsemhaplugins.cpp File Reference

#### Macros

- #define **DEBUG(x)** std::cerr << \_\_FILE\_\_ << ":" << \_\_LINE\_\_ << " " << #x << "=" << x  
<< std::endl

#### Functions

- int **main** (int argc, char \*\*argv)

### 6.35.1 Macro Definition Documentation

```
6.35.1.1 DEBUG #define DEBUG( x ) std::cerr << __FILE__ << ":" << __LINE__ << " " << #x << "=" << x  
<< std::endl
```

---

### 6.35.2 Function Documentation

**6.35.2.1 main()** `int main (`  
    `int argc,`  
    `char ** argv )`

## 6.36 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)`

## 6.37 combinechannels.cpp File Reference

### Classes

- class `combc_t`
- class `combc_if_t`

## 6.38 compiler\_id.cpp File Reference

### 6.39 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**

#### 6.39.1 Macro Definition Documentation

##### 6.39.1.1 **COMPILER\_ID\_VENDOR** #define COMPILER\_ID\_VENDOR "gcc"

##### 6.39.1.2 **COMPILER\_ID\_MAJOR** #define COMPILER\_ID\_MAJOR \_\_GNUC\_\_

##### 6.39.1.3 **COMPILER\_ID\_MINOR** #define COMPILER\_ID\_MINOR \_\_GNUC\_MINOR\_\_

##### 6.39.1.4 **COMPILER\_ID\_PATCH** #define COMPILER\_ID\_PATCH \_\_GNUC\_PATCHLEVEL\_\_

---

**6.39.1.5 COMPILER\_ID\_VERSION\_HELPER2** #define COMPILER\_ID\_VERSION\_HELPER2 (   
 x,  
 y,  
 z ) #x "." #y "." #z

**6.39.1.6 COMPILER\_ID\_VERSION\_HELPER1** #define COMPILER\_ID\_VERSION\_HELPER1 (   
 x,  
 y,  
 z ) COMPILER\_ID\_VERSION\_HELPER2 (x, y, z)

**6.39.1.7 COMPILER\_ID\_VERSION** #define COMPILER\_ID\_VERSION

**6.39.1.8 COMPILER\_ID** #define COMPILER\_ID COMPILER\_ID\_VENDOR "-" COMPILER\_ID\_VERSION "-" COMPILER\_ID\_STANDARD

## 6.40 complex\_filter.cpp File Reference

## 6.41 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.*

## 6.42 complex\_scale\_channel.cpp File Reference

### Classes

- class `cfg_t`
- class `complex_scale_channel_t`

## 6.43 cpupload.cpp File Reference

### Classes

- class `cpupload::cpupload_cfg_t`
- class `cpupload::cpupload_if_t`

### Namespaces

- `cpupload`

## 6.44 db.cpp File Reference

### Classes

- class `db_t`
- class `db_if_t`

## 6.45 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)

## 6.46 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)

#### 6.46.1 Macro Definition Documentation

##### 6.46.1.1 DUPVEC #define DUPVEC(

```
x ) v.x.data = MHASignal::dupvec_chk(v.x.data,s)
```

#### 6.46.2 Function Documentation

##### 6.46.2.1 get\_audiochannels() unsigned int get\_audiochannels (

```
unsigned int totalchannels,
std::string acname,
algo_comm_t ac )
```

## 6.47 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`

## 6.48 dc\_afterburn.cpp File Reference

### Namespaces

- **DynComp**  
*dynamic compression related classes and functions*

### Functions

- float `mylogf` (float x)

#### 6.48.1 Function Documentation

**6.48.1.1 mylogf()** float mylogf (   
                   float x )

## 6.49 dc\_afterburn.h File Reference

### Classes

- class **DynComp::dc\_afterburn\_vars\_t**  
*Variables for `dc_afterburn_t` (p. 449) class.*
- class **DynComp::dc\_afterburn\_rt\_t**  
*Real-time class for after burn effect.*
- class **DynComp::dc\_afterburn\_t**  
*Afterburn class, to be defined as a member of compressors.*

## Namespaces

- **DynComp**

*dynamic compression related classes and functions*

## 6.50 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.*

## 6.51 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.*

## 6.52 delay.cpp File Reference

### Namespaces

- `delay`

## 6.53 delay.hh File Reference

### Classes

- class `delay::interface_t`

### Namespaces

- `delay`

## 6.54 delaysum\_spec.cpp File Reference

### Classes

- class `delaysum_spec::delaysum_t`
- class `delaysum_spec::delaysum_spec_if_t`

## Namespaces

- **delaysum\_spec**

## 6.55 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.*

## 6.56 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)

### 6.56.1 Macro Definition Documentation

#### 6.56.1.1 PATCH\_VAR #define PATCH\_VAR(

```
var ) patchbay.connect(&var.valuechanged, this, & doasvm_classification::update_cfg)
```

#### 6.56.1.2 INSERT\_PATCH #define INSERT\_PATCH(

```
var ) insert_member(var); PATCH_VAR(var)
```

## 6.57 doasvm\_classification.h File Reference

### Classes

- class **doasvm\_classification\_config**
- class **doasvm\_classification**

## 6.58 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)

### 6.58.1 Macro Definition Documentation

**6.58.1.1 PATCH\_VAR** #define PATCH\_VAR(  
    var ) patchbay.connect(&var.valuechanged, this, & **doasvm\_feature\_extraction::update\_cfg**)

**6.58.1.2 INSERT\_PATCH** #define INSERT\_PATCH(  
    var ) **insert\_member**(var); **PATCH\_VAR**(var)

## 6.59 doasvm\_feature\_extraction.h File Reference

### Classes

- class **doasvm\_feature\_extraction\_config**
- class **doasvm\_feature\_extraction**

**6.60 doc\_appendix.h File Reference****6.61 doc\_examples.h File Reference****6.62 doc\_frameworks.h File Reference****6.63 doc\_general.h File Reference****6.64 doc\_kernel.h File Reference****6.65 doc\_matlab.h File Reference****6.66 doc\_mhamain.h File Reference****6.67 doc\_parser.h File Reference****6.68 doc\_plugins.h File Reference****6.69 doc\_system.h File Reference****6.70 doc\_toolbox.h File Reference****6.71 double2acvar.cpp File Reference****Classes**

- class **double2acvar::double2acvar\_t**  
*Plugin interface class for **double2acvar** (p. 90).*

**Namespaces**

- **double2acvar**

**6.72 downsample.cpp File Reference****Classes**

- class **ds\_t**

## 6.73 dropgen.cpp File Reference

### Classes

- class **dropgen\_t**

## 6.74 droptect.cpp File Reference

### Classes

- class **droptect\_t**

*Detect dropouts in a signal with a constant spectrum.*

## 6.75 equalize.cpp File Reference

### Classes

- class **equalize::cfg\_t**
- class **equalize::freqgains\_t**

### Namespaces

- **equalize**

## 6.76 example1.cpp File Reference

### Classes

- class **example1\_t**

*This C++ class implements the simplest example plugin for the step-by-step tutorial.*

## 6.77 example2.cpp File Reference

### Classes

- class **example2\_t**

*This C++ class implements the second example plugin for the step-by-step tutorial.*

## 6.78 example3.cpp File Reference

### Classes

- class **example3\_t**  
*A Plugin class using the openMHA Event mechanism.*

## 6.79 example4.cpp File Reference

### Classes

- class **example4\_t**  
*A Plugin class using the spectral signal.*

## 6.80 example5.cpp File Reference

### Classes

- class **example5\_t**
- class **plugin\_interface\_t**

### Macros

- #define **\_\_declspec(p)**

### 6.80.1 Macro Definition Documentation

#### 6.80.1.1 **\_\_declspec** #define \_\_declspec( p )

## 6.81 example6.cpp File Reference

### Classes

- class **cfg\_t**
- class **example6\_t**

## Macros

- #define \_\_declspec(p)

### 6.81.1 Macro Definition Documentation

**6.81.1.1 \_\_declspec** #define \_\_declspec(  
    p )

## 6.82 example7.cpp File Reference

### 6.83 example7.hh File Reference

## Classes

- class example7\_t

### 6.84 fader\_spec.cpp File Reference

## Classes

- class spec\_fader\_t
- class fader\_if\_t

### 6.85 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
```

### 6.85.1 Macro Definition Documentation

**6.85.1.1 DEBUG**

```
#define DEBUG(
```

  
`x ) std::cerr << __FILE__ << ":" << __LINE__ << " " #x "=" << x <<`  
`std::endl`

### 6.86 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.*

### 6.87 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)`

## 6.88 fftfilterbank.cpp File Reference

### Classes

- class `fftfilterbank::fftfb_plug_t`
- class `fftfilterbank::fftfb_interface_t`

## Namespaces

- `fftfilterbank`

## 6.89 fshift.cpp File Reference

## 6.90 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.*

## 6.91 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.*

## 6.92 gain.cpp File Reference

### Classes

- class **gain::scaler\_t**
- class **gain::gain\_if\_t**

### Namespaces

- **gain**

## 6.93 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)

### 6.93.1 Function Documentation

```
6.93.1.1 convert_f2logf() std::vector< mha_real_t> convert_f2logf (
    const std::vector< mha_real_t > & vF )
```

```
6.93.1.2 isempty() bool isempty (
    const std::vector< std::vector< mha_real_t > > & arg )
```

## 6.94 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.*

## 6.95 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)

### 6.95.1 Function Documentation

**6.95.1.1 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 "}"

**6.95.1.2 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

#### 6.95.1.3 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", ...)

#### 6.95.1.4 main() int main (

```
int argc,  
char ** argv )
```

## 6.96 gsc\_adaptive\_stage.cpp File Reference

### 6.97 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.*

## 6.98 gsc\_adaptive\_stage\_if.cpp File Reference

### 6.99 gsc\_adaptive\_stage\_if.hh File Reference

#### Classes

- class **gsc\_adaptive\_stage::gsc\_adaptive\_stage\_if**  
*Plugin interface class.*

#### Namespaces

- **gsc\_adaptive\_stage**

## 6.100 gtfb\_analyzer.cpp File Reference

Gammatone Filterbank Analyzer Plugin.

## Classes

- struct **gtfb\_analyzer::gtfb\_analyzer\_cfg\_t**  
*Configuration for Gammatone Filterbank Analyzer.*
- class **gtfb\_analyzer::gtfb\_analyzer\_t**  
*Gammatone Filterbank Analyzer Plugin.*

## Namespaces

- **gtfb\_analyzer**

## Functions

- static const **mha\_complex\_t & filter\_complex** (const **mha\_complex\_t &input**, const **mha\_complex\_t &coeff**, **mha\_complex\_t \*states**, unsigned **orders**)  
*Filters a complex input sample with the given filter coefficient.*
- static const **mha\_complex\_t & filter\_real** ( **mha\_real\_t input**, **mha\_complex\_t &tmp←\_complex**, const **mha\_complex\_t &coeff**, **mha\_complex\_t \*states**, unsigned **orders**, const **mha\_complex\_t &normphase**)  
*Filters a real input sample with the given filter coefficient and applies the given normalization with phase correction.*

### 6.100.1 Detailed Description

Gammatone Filterbank Analyzer Plugin.

### 6.100.2 Function Documentation

#### 6.100.2.1 **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

```
6.100.2.2 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 <b>mha_complex_t</b> (p. 741) 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 <b>mha_complex_t</b> (p. 741)
<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

## 6.101 gtfb\_simd.cpp File Reference

Gammatone Filterbank Analyzer Plugin using SIMD.

## Classes

- class `gtfb_simd_cfg_t`
- class `gtfb_simd_t`

## Macros

- `#define add4f(a, b) __builtin_ia32_addps(a,b)`
- `#define sub4f(a, b) __builtin_ia32_subps(a,b)`
- `#define mul4f(a, b) __builtin_ia32_mulps(a,b)`
- `#define MXCSR_DAZ (1 << 6) /* Enable denormals are zero mode */`
- `#define MXCSR_FTZ (1 << 15) /* Enable flush to zero mode */`
- `#define check_alignment(ptr, alignment)`

*Checks alignment of pointer address.*

## Functions

- void `filter_sisd_complex` (const unsigned bands, const unsigned order, const `mha_complex_t` \*inputs, `mha_complex_t` \*outputs, const `mha_complex_t` \*coefficients, `mha_complex_t` \*states)  
*Filters one sample per band, using SISD operations and the mha\_complex operations.*
- void `filter_sisd_real` (const unsigned bands, const unsigned order, const `mha_complex_t` \*inputs, `mha_complex_t` \*outputs, const `mha_complex_t` \*coefficients, `mha_complex_t` \*states)  
*Filters one sample per band, using SISD operations and real operations (operating on real and imaginary part as necessary).*
- void `filter_simd` (const unsigned bands, const unsigned order, const `mha_real_t` \*rinputs, const `mha_real_t` \*iinputs, `mha_real_t` \*routputs, `mha_real_t` \*ioutputs, const `mha_real_t` \*rcoefficients, const `mha_real_t` \*icoefficients, `mha_real_t` \*rstates, `mha_real_t` \*istates)  
*Filters one sample per band, using SIMD operations on float32 To process more than one sample, the function must be called repeatedly in correct order (from oldest sample to newest sample), and the calling function must preserve the filter state (parameters rstates and istates).*

### 6.101.1 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  $(a+b).real == a.real + b.real$ ,  $(a+b).imag == a.imag + b.imag$ ,  $(a*b).real == a.real * b.real - a.imag * b.imag$ ,  $(a*b).imag == a.real * b.imag + a.imag * b.real$ .

## 6.101.2 Macro Definition Documentation

**6.101.2.1 add4f** #define add4f(  
    *a*,  
    *b* ) \_\_builtin\_ia32\_addps(*a*,*b*)

**6.101.2.2 sub4f** #define sub4f(  
    *a*,  
    *b* ) \_\_builtin\_ia32\_subps(*a*,*b*)

**6.101.2.3 mul4f** #define mul4f(  
    *a*,  
    *b* ) \_\_builtin\_ia32\_mulps(*a*,*b*)

**6.101.2.4 MXCSR\_DAZ** #define MXCSR\_DAZ (1 << 6) /\* Enable denormals are zero mode \*/

**6.101.2.5 MXCSR\_FTZ** #define MXCSR\_FTZ (1 << 15) /\* Enable flush to zero mode \*/

**6.101.2.6 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. 760) if ptr is not aligned as required.

## 6.101.3 Function Documentation

```
6.101.3.1 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]

```
6.101.3.2 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]

```
6.101.3.3 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 rinputs, iinputs, routputs, ioutputs, rcoefficients, icoefficients.
<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]

## 6.102 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.*

## 6.103 hann.cpp File Reference

### Macros

- #define **PI** 3.14159265358979323846

### Functions

- float \* **hannf** (const unsigned int N)
- double \* **hann** (const unsigned int N)

#### 6.103.1 Macro Definition Documentation

##### 6.103.1.1 **PI** #define PI 3.14159265358979323846

#### 6.103.2 Function Documentation

##### 6.103.2.1 **hannf()** float\* hannf ( const unsigned int N )

##### 6.103.2.2 **hann()** double\* hann ( const unsigned int N )

## 6.104 hann.h File Reference

### Functions

- float \* **hannf** (const unsigned int N)
- double \* **hann** (const unsigned int N)

### 6.104.1 Function Documentation

**6.104.1.1 `hannf()`** `float* hannf (`  
`const unsigned int N )`

**6.104.1.2 `hann()`** `double* hann (`  
`const unsigned int N )`

## 6.105 identity.cpp File Reference

### Classes

- class `identity_t`

## 6.106 ifftshift.cpp File Reference

### Functions

- void `ifftshift ( mha_wave_t *spec)`

### 6.106.1 Function Documentation

**6.106.1.1 `ifftshift()`** `void ifftshift (`  
`mha_wave_t * spec )`

## 6.107 ifftshift.h File Reference

### Functions

- void `ifftshift ( mha_wave_t *spec)`

### 6.107.1 Function Documentation

**6.107.1.1 ifftshift()** `void ifftshift (`  
 `mha_wave_t * spec )`

### 6.108 iirfilter.cpp File Reference

#### Classes

- class `iirfilter_t`

### 6.109 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)`

### 6.109.1 Macro Definition Documentation

**6.109.1.1 PATCH\_VAR** `#define PATCH_VAR(`  
 `var ) patchbay.connect(&var.valuechanged, this, & level_matching::level_matching_t::update_cfg)`

**6.109.1.2 INSERT\_PATCH** `#define INSERT_PATCH(`  
 `var ) insert_member(var); PATCH_VAR(var)`

## 6.110 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`

## 6.111 levelmeter.cpp File Reference

### Classes

- class `levelmeter_t`

### Macros

- #define **PASCALE** 93.979400086720374929

### 6.111.1 Macro Definition Documentation

#### 6.111.1.1 PASCALE #define PASCALE 93.979400086720374929

## 6.112 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)

### 6.112.1 Macro Definition Documentation

#### 6.112.1.1 **PATCH\_VAR** #define PATCH\_VAR( var ) patchbay.connect(&var.valuechanged, this, & **lpc::update\_cfg**)

#### 6.112.1.2 **INSERT\_PATCH** #define INSERT\_PATCH( var ) insert\_member(var); **PATCH\_VAR**(var)

### 6.112.2 Function Documentation

#### 6.112.2.1 **Levinson2()** void Levinson2 ( unsigned int P, const std::vector< **mha\_real\_t** > & R, std::vector< **mha\_real\_t** > & A )

### 6.113 Ipc.h File Reference

## Classes

- class **Ipc\_config**
- class **Ipc**

### 6.114 Ipc\_bl\_predictor.cpp File Reference

## Macros

- #define **PATCH\_VAR**(var) patchbay.connect(&var.valuechanged, this, & **lpc::update\_cfg**)
- #define **INSERT\_PATCH**(var) insert\_member(var); **PATCH\_VAR**(var)

### 6.114.1 Macro Definition Documentation

**6.114.1.1 PATCH\_VAR** #define PATCH\_VAR(  
    var ) patchbay.connect(&var.valuechanged, this, & **lpc\_b1\_predictor**←  
    ::update\_cfg)

**6.114.1.2 INSERT\_PATCH** #define INSERT\_PATCH(  
    var ) insert\_member(var); **PATCH\_VAR**(var)

## 6.115 **Ipc\_b1\_predictor.h** File Reference

### Classes

- class **Ipc\_b1\_predictor\_config**
- class **Ipc\_b1\_predictor**

### Macros

- #define **EPSILON** 1e-10

### 6.115.1 Macro Definition Documentation

**6.115.1.1 EPSILON** #define EPSILON 1e-10

## 6.116 **Ipc\_burg-lattice.cpp** File Reference

### Macros

- #define **PATCH\_VAR**(var) patchbay.connect(&var.valuechanged, this, & **lpc\_burglattice**::update\_cfg)
- #define **INSERT\_PATCH**(var) insert\_member(var); **PATCH\_VAR**(var)

## 6.116.1 Macro Definition Documentation

**6.116.1.1 PATCH\_VAR** #define PATCH\_VAR(  
     var) patchbay.connect(&var.valuechanged, this, & **lpc\_burglattice**::  
     ::update\_cfg)

**6.116.1.2 INSERT\_PATCH** #define INSERT\_PATCH(  
     var) insert\_member(var); PATCH\_VAR(var)

## 6.117 lpc\_burg-lattice.h File Reference

### Classes

- class **lpc\_burglattice\_config**
- class **lpc\_burglattice**

### Macros

- #define **EPSILON** 1e-10

## 6.117.1 Macro Definition Documentation

**6.117.1.1 EPSILON** #define EPSILON 1e-10

## 6.118 Isl2ac.cpp File Reference

## 6.119 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` }

## 6.120 matlab\_wrapper.cpp File Reference

### 6.121 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 waper plugin interface class.*
- class `matlab_wrapper::matlab_wrapper_t::wrapped_plugin_t`  
*Wrapper class around the matlab-generated library.*

## Namespaces

- `matlab_wrapper`  
*Namespace where all classes of the matlab wrapper plugin live.*

## Macros

- #define `MHAPLUGIN_OVERLOAD_OUTDOMAIN`

### 6.121.1 Macro Definition Documentation

**6.121.1.1 MHAPLUGIN\_OVERLOAD\_OUTDOMAIN** #define MHAPLUGIN\_OVERLOAD\_OUTDO←  
MAIN

## 6.122 matrixmixer.cpp File Reference

### Classes

- class **matrixmixer::cfg\_t**
- class **matrixmixer::matmix\_t**

### Namespaces

- **matrixmixer**

## 6.123 mconv.cpp File Reference

### Classes

- class **mconv::MConv**

### Namespaces

- **mconv**

## 6.124 mha.cpp File Reference

### Functions

- int **mhamain** (int argc, char \*argv[ ])
- int **main** (int argc, char \*argv[ ])

### 6.124.1 Function Documentation

**6.124.1.1 mhamain()** `int mhamain (`  
    `int argc,`  
    `char * argv[ ] )`

**6.124.1.2 main()** `int main (`  
    `int argc,`  
    `char * argv[ ] )`

## 6.125 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. 741) `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. 847)).*
- struct **mha\_audio\_t**  
*An audio fragment in the openMHA (planned as a replacement of **mha\_wave\_t** (p. 836) and **mha\_spec\_t** (p. 790)).*
- 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 16`  
*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 *algo_name, void **h)`
- `typedef int(*) MHAPrepare_t)(void *h, mhaconfig_t *cfg)`
- `typedef int(*) MHARelease_t)(void *h)`
- `typedef void(* MHADestroy_t)(void *h)`
- `typedef int(*) MHASet_t)(void *h, const char *cmd, char *retval, unsigned int len)`
- `typedef std::string(* MHASetcpp_t)(void *h, const std::string &command)`
- `typedef 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)`

### 6.125.1 Detailed Description

common types for MHA kernel, MHA framework applications and external plugins

### 6.125.2 Macro Definition Documentation

#### 6.125.2.1 MHA\_CALLBACK\_TEST #define MHA\_CALLBACK\_TEST( x )

Test macro to compare function type definition and declaration.

**6.125.2.2 MHA\_CALLBACK\_TEST\_PREFIX** #define MHA\_CALLBACK\_TEST\_PREFIX(  
    *prefix*,  
    *x* )

**6.125.2.3 MHA\_XSTRF** #define MHA\_XSTRF(  
    *x* ) MHA\_STRF( *x* )

**6.125.2.4 MHA\_STRF** #define MHA\_STRF(  
    *x* ) #*x*

**6.125.2.5 MHA\_VERSION\_MAJOR** #define MHA\_VERSION\_MAJOR 4

Major version number of MHA.

**6.125.2.6 MHA\_VERSION\_MINOR** #define MHA\_VERSION\_MINOR 16

Minor version number of MHA.

**6.125.2.7 MHA\_VERSION\_RELEASE** #define MHA\_VERSION\_RELEASE 0

Release number of MHA.

**6.125.2.8 MHA\_VERSION\_BUILD** #define MHA\_VERSION\_BUILD 0

Build number of MHA (currently unused)

**6.125.2.9 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.

**6.125.2.10 MHA\_VERSION** #define MHA\_VERSION (unsigned int)(( MHA\_STRUCT\_SIZEMAT←  
TCH | ( MHA\_VERSION\_RELEASE << 8) | ( MHA\_VERSION\_MINOR << 16) | ( MHA\_VERSION\_MAJOR  
<< 24)))

Full version number of MHA kernel.

**6.125.2.11 MHA\_VERSION\_STRING** #define MHA\_VERSION\_STRING MHA\_XSTRF( MHA\_VE←  
RSION\_MAJOR) " ." MHA\_XSTRF( MHA\_VERSION\_MINOR)

Version string of MHA kernel (major.minor)

**6.125.2.12 MHA\_RELEASE\_VERSION\_STRING** #define MHA\_RELEASE\_VERSION\_STRI←  
NG MHA\_XSTRF( MHA\_VERSION\_MAJOR) " ." MHA\_XSTRF( MHA\_VERSION\_MINOR) " ." MHA\_XSTRF( MHA\_VERSI←  
ON\_RELEASE)

Version string of MHA kernel (major.minor.release)

**6.125.2.13 MHA\_WAVEFORM** #define MHA\_WAVEFORM 0

**6.125.2.14 MHA\_SPECTRUM** #define MHA\_SPECTRUM 1

**6.125.2.15 MHA\_DOMAIN\_MAX** #define MHA\_DOMAIN\_MAX 2

**6.125.2.16 MHA\_DOMAIN\_UNKNOWN** #define MHA\_DOMAIN\_UNKNOWN MHA\_DOMAIN\_MAX

**6.125.2.17 MHA\_AC\_UNKNOWN** #define MHA\_AC\_UNKNOWN 0

**6.125.2.18 MHA\_AC\_CHAR** #define MHA\_AC\_CHAR 1

**6.125.2.19 MHA\_AC\_INT** #define MHA\_AC\_INT 2

**6.125.2.20 MHA\_AC\_MHAREAL** #define MHA\_AC\_MHAREAL 3

**6.125.2.21 MHA\_AC\_FLOAT** #define MHA\_AC\_FLOAT 4

**6.125.2.22 MHA\_AC\_DOUBLE** #define MHA\_AC\_DOUBLE 5

**6.125.2.23 MHA\_AC\_MHACOMPLEX** #define MHA\_AC\_MHACOMPLEX 6

**6.125.2.24 MHA\_AC\_VEC\_FLOAT** #define MHA\_AC\_VEC\_FLOAT 51

**6.125.2.25 MHA\_AC\_USER** #define MHA\_AC\_USER 1000

### 6.125.3 Typedef Documentation

**6.125.3.1 `mha_domain_t`** `typedef unsigned int mha_domain_t`

**6.125.3.2 `algo_comm_t`** `typedef struct algo_comm_t algo_comm_t`

**6.125.3.3 `MHAGetVersion_t`** `typedef unsigned int (* MHAGetVersion_t) (void)`

**6.125.3.4 `MHAInit_t`** `typedef int(* MHAInit_t) ( algo_comm_t algo_comm, const char *algo_name, void **h)`

**6.125.3.5 `MHAPrepare_t`** `typedef int(* MHAPrepare_t) (void *h, mhaconfig_t *cfg)`

**6.125.3.6 `MHARelease_t`** `typedef int(* MHARelease_t) (void *h)`

**6.125.3.7 `MHADestroy_t`** `typedef void(* MHADestroy_t) (void *h)`

**6.125.3.8 `MHASet_t`** `typedef int(* MHASet_t) (void *h, const char *cmd, char *retval, unsigned int len)`

**6.125.3.9 MHASetcpp\_t** `typedef std::string(* MHASetcpp_t) (void *h, const std::string &command)`

**6.125.3.10 MHAProc\_wave2wave\_t** `typedef int(* MHAProc_wave2wave_t) (void *h, mha_wave_t *sIn, mha_wave_t **sOut)`

**6.125.3.11 MHAProc\_wave2spec\_t** `typedef int(* MHAProc_wave2spec_t) (void *h, mha_wave_t *sIn, mha_spec_t **sOut)`

**6.125.3.12 MHAProc\_spec2wave\_t** `typedef int(* MHAProc_spec2wave_t) (void *h, mha_spec_t *sIn, mha_wave_t **sOut)`

**6.125.3.13 MHAProc\_spec2spec\_t** `typedef int(* MHAProc_spec2spec_t) (void *h, mha_spec_t *sIn, mha_spec_t **sOut)`

## 6.125.4 Variable Documentation

**6.125.4.1 MHAStrError\_t** `const typedef char*(* MHAStrError_t) (void *h, int err)`

**6.125.4.2 MHAPluginDocumentation\_t** `const typedef char*(* MHAPluginDocumentation_t) (void)`

**6.125.4.3 MHAPluginCategory\_t** `const typedef char*(* MHAPluginCategory_t) (void)`

## 6.126 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**

#### 6.126.1 Macro Definition Documentation

##### 6.126.1.1 **AC\_SUCCESS** #define AC\_SUCCESS 0

##### 6.126.1.2 **AC\_INVALID\_HANDLE** #define AC\_INVALID\_HANDLE -1

##### 6.126.1.3 **AC\_INVALID\_NAME** #define AC\_INVALID\_NAME -2

##### 6.126.1.4 **AC\_STRING\_TRUNCATED** #define AC\_STRING\_TRUNCATED -3

##### 6.126.1.5 **AC\_INVALID\_OUTPTR** #define AC\_INVALID\_OUTPTR -4

### 6.126.1.6 AC\_TYPE\_MISMATCH #define AC\_TYPE\_MISMATCH -5

### 6.126.1.7 AC\_DIM\_MISMATCH #define AC\_DIM\_MISMATCH -6

## 6.126.2 Variable Documentation

### 6.126.2.1 algo\_comm\_default algo\_comm\_t algo\_comm\_default

## 6.127 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.*

### 6.127.1 Detailed Description

Header file for Algorithm Communication.

## 6.128 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** "MFVK3jL5rmeus1XtggEl971aXCR/GU7RRehKz4k←  
Qtrg=

## Functions

- algo\_comm\_class\_t \* **MHAKernel::algo\_comm\_safe\_cast** (void \*)

## Variables

- **algo\_comm\_t algo\_comm\_default**

### 6.128.1 Macro Definition Documentation

**6.128.1.1 ALGO\_COMM\_ID\_STR** #define ALGO\_COMM\_ID\_STR "MFVK3jL5rmeus1XtggE↔  
I971aXCR/GU7RRehKz4kQtrg=

### 6.128.2 Variable Documentation

**6.128.2.1 algo\_comm\_default** algo\_comm\_t algo\_comm\_default

## 6.129 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

### 6.129.1 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.

### 6.129.2 Macro Definition Documentation

#### 6.129.2.1 **\_MHA\_FUN\_** #define \_\_MHA\_FUN\_\_ \_\_FUNC\_\_

```
6.129.2.2 CHECK_EXPR #define CHECK_EXPR(  
    x ) {if(! (x)){throw  MHA_Error(__FILE__, __LINE__, "The expression \"#" x  
"\" is invalid.");}}
```

```
6.129.2.3 CHECK_VAR #define CHECK_VAR(  
    x ) {if(! (x)){throw  MHA_Error(__FILE__, __LINE__, "The variable \"#" x  
"\" is not defined.");}}
```

```
6.129.2.4 _declspec #define __declspec(  
    P )
```

```
6.129.2.5 M_PI #define M_PI 3.14159265358979323846
```

Define pi if it is not defined yet.

**6.129.2.6 MIN** #define MIN(  
    a,  
    b ) (( (a) < (b) ) ? (a) : (b) )

Macro for minimum function.

**6.129.2.7 MAX** #define MAX(  
    a,  
    b ) (( (a) > (b) ) ? (a) : (b) )

Macro for maximum function.

**6.129.2.8 MHA\_EAR\_LEFT** #define MHA\_EAR\_LEFT 0

**6.129.2.9 MHA\_EAR\_RIGHT** #define MHA\_EAR\_RIGHT 1

**6.129.2.10 MHA\_EAR\_MAX** #define MHA\_EAR\_MAX 2

## 6.130 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**]

### 6.130.1 Macro Definition Documentation

#### 6.130.1.1 **STRLEN** #define STRLEN 0x1000

### 6.130.2 Function Documentation

#### 6.130.2.1 **mha\_strerror()** const char\* mha\_strerror ( int mhaerrno )

#### 6.130.2.2 **mha\_set\_user\_error()** void mha\_set\_user\_error ( const char \* str )

### 6.130.3 Variable Documentation

#### 6.130.3.1 **next\_except\_str** char next\_except\_str[ **STRLEN**] = ""

#### 6.130.3.2 **cstr\_strerror** const char\* cstr\_strerror[ **MHA\_ERR\_USER**]

## 6.131 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)

#### 6.131.1 Macro Definition Documentation

##### 6.131.1.1 **MHA\_ERR\_SUCCESS** #define MHA\_ERR\_SUCCESS 0

##### 6.131.1.2 **MHA\_ERR\_UNKNOWN** #define MHA\_ERR\_UNKNOWN 1

##### 6.131.1.3 **MHA\_ERR\_INVALID\_HANDLE** #define MHA\_ERR\_INVALID\_HANDLE 2

##### 6.131.1.4 **MHA\_ERR\_NULL** #define MHA\_ERR\_NULL 3

**6.131.1.5 MHA\_ERR\_VARRANGE** #define MHA\_ERR\_VARRANGE 4

**6.131.1.6 MHA\_ERR\_VARFMT** #define MHA\_ERR\_VARFMT 5

**6.131.1.7 MHA\_ERR\_USER** #define MHA\_ERR\_USER 10000

## 6.131.2 Function Documentation

**6.131.2.1 mha\_strerror()** const char\* mha\_strerror ( int mhaerrno )

**6.131.2.2 mha\_set\_user\_error()** void mha\_set\_user\_error ( const char \* str )

## 6.132 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,...)

### 6.132.1 Detailed Description

Implementation of openMHA error handling.

This file forms a separate library.

### 6.132.2 Function Documentation

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

## 6.133 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. 760).*
- #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. 760) with the values.*

## Functions

- void **mha\_debug** (const char \*fmt,...) \_\_attribute\_\_((\_\_format\_\_(printf  
*Print an info message (stderr on Linux, OutputDebugString in Windows).*
- unsigned **mha\_error\_helpers::digits** (unsigned n)  
*Compute number of decimal digits required to represent an unsigned integer.*
- unsigned **mha\_error\_helpers::snprintf\_required\_length** (const char \*formatstring,...)  
*snprintf\_required\_length Compute the number of bytes (excluding the terminating nul) required to store the result of an snprintf.*

### 6.133.1 Macro Definition Documentation

#### 6.133.1.1 Getmsg #define Getmsg( e ) ((e).get\_msg())

## 6.134 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.*

## 6.135 mha\_events.cpp File Reference

## 6.136 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.*

## 6.137 mha\_fffb.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 filtershaperfun ( 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]**

### 6.137.1 Macro Definition Documentation

#### 6.137.1.1 BARKSCALE\_ENTRIES #define BARKSCALE\_ENTRIES 50

### 6.137.2 Function Documentation

#### 6.137.2.1 filtershaperfun() mha\_real\_t filtershaperfun (

```
    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.

## 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.

## 6.138 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)`

## 6.139 mha\_fifo.cpp File Reference

## 6.140 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. 786).*
- class **mha\_rt\_fifo\_t< T >**  
*Template class for thread safe, half real time safe fifo without explicit locks.*

## Macros

- `#define mha_fifo_thread_platform_implementation_t mha_fifo_posix_threads_t`

### 6.140.1 Macro Definition Documentation

**6.140.1.1 mha\_fifo\_thread\_platform\_implementation\_t** `#define mha_fifo_thread_←  
platform_implementation_t mha_fifo_posix_threads_t`

## 6.141 mha\_filter.cpp File Reference

### Functions

- `std::vector< mha_real_t > diff_coeffs ()`

### 6.141.1 Function Documentation

**6.141.1.1 diff\_coeffs()** `std::vector< mha_real_t > diff_coeffs ( )`

## 6.142 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.*

### 6.142.1 Detailed Description

Header file for IIR filter classes.

## 6.143 mha\_generic\_chain.cpp File Reference

### Functions

- void **mhaconfig\_compare** ( **mhaconfig\_t** req, **mhaconfig\_t** avail, const char \*cpref)

### 6.143.1 Function Documentation

```
6.143.1.1 mhaconfig_compare() void mhaconfig_compare (
    mhaconfig_t req,
    mhaconfig_t avail,
    const char * cpref )
```

## 6.144 mha\_generic\_chain.h File Reference

### Classes

- class **mhachain::plugs\_t**
- class **mhachain::chain\_base\_t**

## Namespaces

- **mhachain**

## Macros

- #define **MHAPLUGIN\_OVERLOAD\_OUTDOMAIN**

### 6.144.1 Macro Definition Documentation

**6.144.1.1 MHAPLUGIN\_OVERLOAD\_OUTDOMAIN** #define MHAPLUGIN\_OVERLOAD\_OUTDO←  
MAIN

## 6.145 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*

### 6.145.1 Macro Definition Documentation

**6.145.1.1 GITCOMMITHASH** #define GITCOMMITHASH "independent-plugin-build"

### 6.145.2 Variable Documentation

### 6.145.2.1 **mha\_git\_commit\_hash** const char\* mha\_git\_commit\_hash

store git commit hash in every binary plugin to support reproducible research

## 6.146 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*

### 6.146.1 Variable Documentation

#### 6.146.1.1 **mha\_git\_commit\_hash** const char\* mha\_git\_commit\_hash

store git commit hash in every binary plugin to support reproducible research

## 6.147 mha\_io\_ifc.h File Reference

### Macros

- #define **MHAIO\_DOCUMENTATION\_PREFIX**(prefix, cat, doc)
- #define **MHAIO\_DOCUMENTATION**(pluginname, cat, doc) **MHAIO\_DOCUMENTATIONPREFIX**(MHA\_STATIC\_ ## pluginname ## \_cat,doc)

### 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)

### 6.147.1 Macro Definition Documentation

**6.147.1.1 MHAIO\_DOCUMENTATION\_PREFIX** #define MHAIO\_DOCUMENTATION\_PREFIX (  
    *prefix*,  
    *cat*,  
    *doc* )

**6.147.1.2 MHAIO\_DOCUMENTATION** #define MHAIO\_DOCUMENTATION (  
    *pluginname*,  
    *cat*,  
    *doc* )   **MHAIO\_DOCUMENTATION\_PREFIX**(MHA\_STATIC\_ ## *pluginname* ## \_,*cat*,*doc*)

### 6.147.2 Typedef Documentation

**6.147.2.1 IOProcessEvent\_t** typedef int(\* IOProcessEvent\_t) (void \*handle, **mha\_wave\_t** \**sIn*, **mha\_wave\_t** \*\**sOut*)

Event handler for signal stream.

This event handler needs to be realtime compatible. All signal path processing will be performed in this callback.

**6.147.2.2 IOStoppedEvent\_t** typedef void(\* IOStoppedEvent\_t) (void \*handle, int *proc\_err*, int *io\_err*)

Event handler for stop event.

This event handler needs to be realtime compatible. The function must return immediatly.

**6.147.2.3 IOStartedEvent\_t** `typedef void(* IOStartedEvent_t) (void *handle)`

Event handler for start event.

This event handler needs to be realtime compatible. The function must return immediatly.

**6.147.2.4 IOInit\_t** `typedef int(* IOInit_t) (int fragsize, float samplerate, IOProcessEvent_t proc_event, void *proc_handle, IOStartedEvent_t start_event, void *start_handle, IOStoppedEvent_t stop_event, void *stop_handle, void **handle)`**6.147.2.5 IOPrepare\_t** `typedef int(* IOPrepare_t) (void *handle, int num_inchannels, int num_outchannels)`**6.147.2.6 IOStart\_t** `typedef int(* IOStart_t) (void *handle)`**6.147.2.7 IOStop\_t** `typedef int(* IOStop_t) (void *handle)`**6.147.2.8 IOReset\_t** `typedef int(* IOReset_t) (void *handle)`**6.147.2.9 IOSetVar\_t** `typedef int(* IOSetVar_t) (void *handle, const char *cmd, char *retval, unsigned int len)`**6.147.2.10 IODestroy\_t** `typedef void(* IODestroy_t) (void *handle)`**6.147.3 Variable Documentation**

**6.147.3.1 IOStrError\_t** const typedef char\*(\* IOStrError\_t) (void \*handle, int err)

## 6.148 mha\_io\_utils.cpp File Reference

### 6.149 mha\_io\_utils.hh File Reference

#### Namespaces

- **mhaioutils**

#### Functions

- template<typename T >  
T **mhaioutils::to\_int\_clamped** (float val)

## 6.150 mha\_multisrc.cpp File Reference

#### Namespaces

- **MHAMultiSrc**

*Collection of classes for selecting audio chunks from multiple sources.*

## 6.151 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.*

---

## 6.152 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)`

### 6.152.1 Function Documentation

#### 6.152.1.1 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

#### 6.152.1.2 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

**6.152.1.3 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
---------------------	--

**6.152.1.4 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

**6.152.1.5 mha\_library\_paths()** `std::list<std::string> mha_library_paths ( )`

---

**6.152.1.6 `list_dir()`** `std::list<std::string> list_dir (`  
`const std::string & path,`  
`const std::string & pattern )`

## 6.153 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. 442) for mha plugin libraries.*

### Macros

- #define **mha\_loadlib(x)** `dlopen(x,RTLD_NOW)`
- #define **mha\_freelib(x)** `dlclose(x)`
- #define **mha\_freelib\_success(x)** `(x == 0)`
- #define **mha\_getlibfun(h, x)** `x ## _cb = (x ## _t)dlsym(h,#x)`
- #define **mha\_getlibfun\_checked(h, x)** `x ## _cb = (x ## _t)dlsym(h,#x);if(! x ## _cb) throw MHA_Error(__FILE__,__LINE__,"Function " #x " is undefined.")`
- #define **mha\_loadlib\_error(x)** `dlerror()`
- #define **mha\_lib\_extension** ".so"
- #define **mha\_msleep(milliseconds)** `usleep((milliseconds)*1000)`
- #define **FMTsz** "%zu"  
*printf modifier to print integers of type size\_t*
- #define **MHA\_RESOLVE(h, t)** `t ## _cb = (t ## _t)(h->resolve(#t))`
- #define **MHA\_RESOLVE\_CHECKED(h, t)** `t ## _cb = (t ## _t)(h->resolve_checked(#t))`

### Typedefs

- typedef void \* **mha\_libhandle\_t**

## Functions

- std::string **mha\_getenv** (const std::string &envvar)  
*Get value of environment variable.*
- bool **mha\_hasenv** (const std::string &envvar)  
*Checks if environment variable exists.*
- int **mha\_setenv** (const std::string &envvar, const std::string & **value**)  
*Set value of environment variable.*
- void **mha\_delenv** (const std::string &envvar)  
*Deletes environment variable from process environment if it exists.*
- std::list< std::string > **mha\_library\_paths** ()
- std::list< std::string > **list\_dir** (const std::string &path, const std::string &pattern)
- void **mha\_hton** (float \*data, unsigned int len)
- void **mha\_ntoh** (float \*data, unsigned int len)
- void **mha\_hton** (uint32\_t \*data, unsigned int len)
- void **mha\_ntoh** (uint32\_t \*data, unsigned int len)
- void **mha\_hton** (int32\_t \*data, unsigned int len)
- void **mha\_ntoh** (int32\_t \*data, unsigned int len)

### 6.153.1 Macro Definition Documentation

**6.153.1.1 mha\_loadlib** #define mha\_loadlib(  
x ) dlopen(x,RTLD\_NOW)

**6.153.1.2 mha\_freelib** #define mha\_freelib(  
x ) dlclose(x)

**6.153.1.3 mha\_freelib\_success** #define mha\_freelib\_success(  
x ) (x == 0)

**6.153.1.4 mha\_getlibfun** #define mha\_getlibfun(  
h,  
x ) x ## \_cb = (x ## \_t)dlsym(h,#x)

---

**6.153.1.5 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.")

**6.153.1.6 mha\_loadlib\_error** #define mha\_loadlib\_error(  
*x*) dlerror()

**6.153.1.7 mha\_lib\_extension** #define mha\_lib\_extension ".so"

**6.153.1.8 mha\_msleep** #define mha\_msleep(  
*milliseconds*) usleep((*milliseconds*)\*1000)

**6.153.1.9 FMTsz** #define FMTsz "%zu"

printf modifier to print integers of type size\_t

**6.153.1.10 MHA\_RESOLVE** #define MHA\_RESOLVE(  
*h*,  
*t*) *t* ## \_cb = (*t* ## \_t) (*h*->resolve(#*t*))

**6.153.1.11 MHA\_RESOLVE\_CHECKED** #define MHA\_RESOLVE\_CHECKED(  
*h*,  
*t*) *t* ## \_cb = (*t* ## \_t) (*h*->resolve\_checked(#*t*))

## 6.153.2 Typedef Documentation

**6.153.2.1 mha\_libhandle\_t** `typedef void* mha_libhandle_t`**6.153.3 Function Documentation****6.153.3.1 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

**6.153.3.2 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

**6.153.3.3 mha\_setenv()** `int mha_setenv (`  
`const std::string & envvar,`  
`const std::string & value )`

Set value of environment variable.

**Parameters**

<i>envvar</i>	Name of environment variable to set
<i>value</i>	New content for environment variable

**Returns**

error code: 0 on success, OS dependent error code on failure

**6.153.3.4 mha\_delenv()** `void mha_delenv (`  
`const std::string & envvar )`

Deletes environment variable from process environment if it exists.

**Parameters**

<i>envvar</i>	Name of environment variable to delete
---------------	--

**6.153.3.5 mha\_library\_paths()** `std::list<std::string> mha_library_paths ( )`

**6.153.3.6 list\_dir()** `std::list<std::string> list_dir (`  
`const std::string & path,`  
`const std::string & pattern )`

**6.153.3.7 mha\_hton()** [1/3] `void mha_hton (`  
`float * data,`  
`unsigned int len ) [inline]`

**6.153.3.8 mha\_ntoh()** [1/3] void mha\_ntoh ( float \* *data*, unsigned int *len* ) [inline]

**6.153.3.9 mha\_hton()** [2/3] void mha\_hton ( uint32\_t \* *data*, unsigned int *len* ) [inline]

**6.153.3.10 mha\_ntoh()** [2/3] void mha\_ntoh ( uint32\_t \* *data*, unsigned int *len* ) [inline]

**6.153.3.11 mha\_hton()** [3/3] void mha\_hton ( int32\_t \* *data*, unsigned int *len* ) [inline]

**6.153.3.12 mha\_ntoh()** [3/3] void mha\_ntoh ( int32\_t \* *data*, unsigned int *len* ) [inline]

## 6.154 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.*

### 6.154.1 Macro Definition Documentation

#### 6.154.1.1 MHAPLATFORM #define MHAPLATFORM "undefined-linux"

### 6.154.2 Function Documentation

#### 6.154.2.1 **write\_float()** static std::ostream& **write\_float** ( std::ostream & o, const float & f ) [inline], [static]

#### 6.154.2.2 **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.

**6.154.2.3 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 ')'.
--------------	--

**6.154.2.4 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

### 6.154.2.5 `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.

## 6.155 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)

### 6.155.1 Detailed Description

Header file for the MHA-Parser script language.

## 6.155.2 Macro Definition Documentation

### 6.155.2.1 DEFAULT\_RET\_SIZE #define DEFAULT\_RET\_SIZE 0x100000

### 6.155.2.2 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. 1100).

## 6.156 mha\_plugin.cpp File Reference

## 6.157 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.*

### 6.157.1 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`).

### 6.157.2 Macro Definition Documentation

#### 6.157.2.1 **\_\_declspec** #define **\_\_declspec**( p )

**6.157.2.2 WINAPI** #define WINAPI**6.157.2.3 HINSTANCE** #define HINSTANCE int

```
6.157.2.4 MHAPLUGIN_PROC_CALLBACK_PREFIX #define MHAPLUGIN_PROC_CALLBACKPREFIX(
    prefix,
    classname,
    indom,
    outdom )
```

```
6.157.2.5 MHAPLUGIN_SETCPP_CALLBACK_PREFIX #define MHAPLUGIN_SETCPP_CALLBACKPREFIX(
    prefix,
    classname )
```

```
6.157.2.6 MHAPLUGIN_INIT_CALLBACKS_PREFIX #define MHAPLUGIN_INIT_CALLBACKSPREFIX(
    prefix,
    classname )
```

```
6.157.2.7 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. 1147) template class. Exceptions of type **MHA\_Error** (p. 760) are caught and transformed into appropriate error codes with their corresponding error messages.

#### 6.157.2.8 MHAPLUGIN\_DOCUMENTATION\_PREFIX

```
#define MHAPLUGIN_DOCUMENTATIONPREFIX(
    prefix,
    cat,
    doc )
```

#### 6.157.2.9 MHAPLUGIN\_PROC\_CALLBACK

```
#define MHAPLUGIN_PROC_CALLBACK (
    plugname,
    classname,
    indom,
    outdom )  MHAPLUGIN_PROC_CALLBACK_PREFIX(MHA_STATIC_ ## plugname ## _←
,classname,indom,outdom)
```

#### 6.157.2.10 MHAPLUGIN\_INIT\_CALLBACKS

```
#define MHAPLUGIN_INIT_CALLBACKS (
    plugname,
    classname )  MHAPLUGIN_INIT_CALLBACKS_PREFIX(MHA_STATIC_ ## plugname ## _←
,classname)
```

#### 6.157.2.11 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. 1147) template class. Exceptions of type **MHA\_Error** (p. 760) are caught and transformed into appropriate error codes with their corresponding error messages.

```
6.157.2.12 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.

## 6.158 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*)

### 6.158.1 Function Documentation

**6.158.1.1 mha\_tic()** void mha\_tic (   
   mha\_tictoc\_t \* t )

**6.158.1.2 mha\_platform\_tic()** void mha\_platform\_tic (   
   mha\_platform\_tictoc\_t \* t )

**6.158.1.3 mha\_toc()** float mha\_toc (   
   mha\_tictoc\_t \* t )

**6.158.1.4 mha\_platform\_toc()** float mha\_platform\_toc (   
   mha\_platform\_tictoc\_t \* t )

## 6.159 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)**

### 6.159.1 TypeDef Documentation

**6.159.1.1 mha\_platform\_tictoc\_t** `typedef mha_tictoc_t mha_platform_tictoc_t`**6.159.2 Function Documentation****6.159.2.1 mha\_platform\_tic()** `void mha_platform_tic (`  
`mha_platform_tictoc_t * t )`**6.159.2.2 mha\_platform\_toc()** `float mha_platform_toc (`  
`mha_platform_tictoc_t * t )`**6.160 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 ()`

**6.160.1 Typedef Documentation****6.160.1.1 rb\_f\_t** `typedef VALUE(* rb_f_t) (...)`**6.160.2 Function Documentation**

**6.160.2.1 `mha_free()`** static void mha\_free ( void \* *mha* ) [static]

**6.160.2.2 `mha_alloc()`** static VALUE mha\_alloc ( VALUE *klass* ) [static]

**6.160.2.3 `mha_exit_request()`** static VALUE mha\_exit\_request ( VALUE *self* ) [static]

**6.160.2.4 `mha_parse()`** static VALUE mha\_parse ( VALUE *self*, VALUE *request* ) [static]

**6.160.2.5 `Init_mha_ruby()`** void Init\_mha\_ruby ( )

## 6.161 `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. 836) 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. 836) 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. 790) structure into a std::vector< std::vector< mha\_complex\_t > > (outer vector represents channels).*
- static **mha\_real\_t intensity** (const **mha\_spec\_t** &s, unsigned int channel, unsigned int fftlen, **mha\_real\_t** \*sqfreq\_response=0)
- void **integrate** ( **mha\_wave\_t** &s)  
*Numeric integration of a signal vector (real values)*
- void **integrate** ( **mha\_spec\_t** &s)  
*Numeric integration of a signal vector (complex values)*
- **mha\_wave\_t** & **operator^=** ( **mha\_wave\_t** &self, const **mha\_real\_t** &arg)  
*Exponent operator.*
- **mha\_wave\_t** **range** ( **mha\_wave\_t** s, unsigned int k0, unsigned int len)  
*Return a time interval from a waveform chunk.*
- **mha\_spec\_t** **channels** ( **mha\_spec\_t** s, unsigned int ch\_start, unsigned int nch)  
*Return a channel interval from a spectrum.*
- void **assign** ( **mha\_wave\_t** self, const **mha\_wave\_t** &val)  
*Set all values of waveform 'self' to 'val'.*
- void **assign** ( **mha\_spec\_t** self, const **mha\_spec\_t** &val)  
*Set all values of spectrum 'self' to 'val'.*
- void **timeshift** ( **mha\_wave\_t** &self, int shift)  
*Time shift of waveform chunk.*

### 6.161.1 Macro Definition Documentation

**6.161.1.1 MHA\_ID\_UINT\_VECTOR** #define MHA\_ID\_UINT\_VECTOR "MHASignal::uint\_<vector\_t"

**6.161.1.2 MHA\_ID\_MATRIX** #define MHA\_ID\_MATRIX "MHASignal::matrix\_t"

**6.161.1.3 ASSERT\_EQUAL\_DIM** #define ASSERT\_EQUAL\_DIM(  
    a,  
    b )

**6.161.1.4 ASSERT\_EQUAL\_DIM\_PTR** #define ASSERT\_EQUAL\_DIM\_PTR(  
    a,  
    b )

## 6.161.2 Function Documentation

**6.161.2.1 set\_minabs()** void set\_minabs (  
    mha\_spec\_t & self,  
    const mha\_real\_t & m )

**6.161.2.2 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.

```
6.161.2.3 intensity() static mha_real_t intensity (
    const mha_spec_t & s,
    unsigned int channel,
    unsigned int fftlen,
    mha_real_t * sqfreq_response = 0 ) [static]
```

## 6.162 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. 790))*
- class **MHASignal::waveform\_t**  
*signal processing class for waveform data (based on **mha\_wave\_t** (p. 836))*
- 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. 836).*
- **mha\_real\_t** **MHASignal::lin2db** ( **mha\_real\_t** x, **mha\_real\_t** eps)  
*Conversion from linear scale to dB (no SPL reference)*
- **mha\_real\_t** **MHASignal::lin2db** ( **mha\_real\_t** x)  
*Conversion from linear scale to dB (no SPL reference)*
- **mha\_real\_t** **MHASignal::db2lin** ( **mha\_real\_t** x)  
*Conversion from dB scale to linear (no SPL reference)*
- **mha\_real\_t** **MHASignal::sq2db** ( **mha\_real\_t** x, **mha\_real\_t** eps=0.0f)  
*conversion from squared values to dB (no SPL reference)*
- **mha\_real\_t** **MHASignal::db2sq** ( **mha\_real\_t** x)  
*conversion from dB to squared values (no SPL reference)*
- **mha\_real\_t** **MHASignal::pa2dbspl** ( **mha\_real\_t** x, **mha\_real\_t** eps)  
*Conversion from linear Pascal scale to dB SPL.*
- **mha\_real\_t** **MHASignal::pa2dbspl** ( **mha\_real\_t** x)  
*Conversion from linear Pascal scale to dB SPL.*
- **mha\_real\_t** **MHASignal::dbspl2pa** ( **mha\_real\_t** x)  
*Conversion from dB SPL to linear Pascal scale.*
- **mha\_real\_t** **MHASignal::pa22dbspl** ( **mha\_real\_t** x, **mha\_real\_t** eps=0.0f)  
*Conversion from squared Pascal scale to dB SPL.*
- **mha\_real\_t** **MHASignal::dbspl2pa2** ( **mha\_real\_t** x)  
*conversion from dB SPL to squared Pascal scale*
- **mha\_real\_t** **MHASignal::smp2sec** ( **mha\_real\_t** n, **mha\_real\_t** srate)  
*conversion from samples to seconds*
- **mha\_real\_t** **MHASignal::sec2smp** ( **mha\_real\_t** sec, **mha\_real\_t** srate)  
*conversion from seconds to samples*
- **mha\_real\_t** **MHASignal::bin2freq** ( **mha\_real\_t** bin, unsigned fftlen, **mha\_real\_t** srate)  
*conversion from fft bin index to frequency*
- **mha\_real\_t** **MHASignal::freq2bin** ( **mha\_real\_t** freq, unsigned fftlen, **mha\_real\_t** srate)  
*conversion from frequency to fft bin index*

- conversion from frequency to fft bin index*
- **mha\_real\_t MHASignal::smp2rad** ( **mha\_real\_t** samples, unsigned bin, unsigned fftlen)
  - conversion from delay in samples to phase shift*
- **mha\_real\_t MHASignal::rad2smp** ( **mha\_real\_t** phase\_shift, unsigned bin, unsigned fftlen)
  - conversion from phase shift to delay in samples*
- template<class elem\_type >  
**std::vector< elem\_type > MHASignal::dupvec** (std::vector< elem\_type > vec, unsigned n)
  - Duplicate last vector element to match desired size.*
- template<class elem\_type >  
**std::vector< elem\_type > MHASignal::dupvec\_chk** (std::vector< elem\_type > vec, unsigned n)
  - Duplicate last vector element to match desired size, check for dimension.*
- **bool equal\_dim** (const **mha\_wave\_t** &a, const **mha\_wave\_t** &b)
  - Test for equal dimension of waveform structures.*
- **bool equal\_dim** (const **mha\_wave\_t** &a, const **mhaconfig\_t** &b)
  - Test for match of waveform dimension with mhaconfig structure.*
- **bool equal\_dim** (const **mha\_spec\_t** &a, const **mha\_spec\_t** &b)
  - Test for equal dimension of spectrum structures.*
- **bool equal\_dim** (const **mha\_spec\_t** &a, const **mhaconfig\_t** &b)
  - Test for match of spectrum dimension with mhaconfig structure.*
- **bool equal\_dim** (const **mha\_wave\_t** &a, const **mha\_spec\_t** &b)
  - Test for equal dimension of waveform/spectrum structures.*
- **bool equal\_dim** (const **mha\_spec\_t** &a, const **mha\_wave\_t** &b)
  - Test for equal dimension of waveform/spectrum structures.*
- **void integrate** ( **mha\_wave\_t** &s)
  - Numeric integration of a signal vector (real values)*
- **void integrate** ( **mha\_spec\_t** &s)
  - Numeric integration of a signal vector (complex values)*
- **unsigned int mha\_min\_1** (unsigned int a)
- **unsigned int size** (const **mha\_wave\_t** &s)
  - Return size of a waveform structure.*
- **unsigned int size** (const **mha\_spec\_t** &s)
  - Return size of a spectrum structure.*
- **unsigned int size** (const **mha\_wave\_t** \*s)
  - Return size of a waveform structure.*
- **unsigned int size** (const **mha\_spec\_t** \*s)
  - Return size of a spectrum structure.*
- **void clear** ( **mha\_wave\_t** &s)
  - Set all values of waveform to zero.*
- **void clear** ( **mha\_wave\_t** \*s)
  - Set all values of waveform to zero.*
- **void clear** ( **mha\_spec\_t** &s)
  - Set all values of spectrum to zero.*

- void **clear** ( **mha\_spec\_t** \*s)
 

*Set all values of spectrum to zero.*
- void **assign** ( **mha\_wave\_t** self, **mha\_real\_t** val)
 

*Set all values of waveform 'self' to 'val'.*
- void **assign** ( **mha\_wave\_t** self, const **mha\_wave\_t** &val)
 

*Set all values of waveform 'self' to 'val'.*
- void **assign** ( **mha\_spec\_t** self, const **mha\_spec\_t** &val)
 

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

*Time shift of waveform chunk.*
- **mha\_wave\_t** **range** ( **mha\_wave\_t** s, unsigned int k0, unsigned int len)
 

*Return a time interval from a waveform chunk.*
- **mha\_spec\_t** **channels** ( **mha\_spec\_t** s, unsigned int ch\_start, unsigned int nch)
 

*Return a channel interval from a spectrum.*
- **mha\_real\_t** & **value** ( **mha\_wave\_t** \*s, unsigned int fr, unsigned int ch)
 

*Access an element of a waveform structure.*
- const **mha\_real\_t** & **value** (const **mha\_wave\_t** \*s, unsigned int fr, unsigned int ch)
 

*Constant access to an element of a waveform structure.*
- **mha\_real\_t** & **value** ( **mha\_wave\_t** \*s, unsigned int k)
 

*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. 836) 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. 836) 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. 790) 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. 741) variable.*
- **mha\_complex\_t mha\_complex (mha\_real\_t real, mha\_real\_t imag=0)**  
*Create a new **mha\_complex\_t** (p. 741) 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. 741) 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. 741).*
- **mha\_complex\_t & expi (mha\_complex\_t &self, mha\_real\_t angle)**  
*replaces the value of the given **mha\_complex\_t** (p. 741) 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. 741) 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. 741) with its conjugate.*
- **void conjugate ( mha\_spec\_t &self)**

*Replace (!) the value of this **mha\_spec\_t** (p. 790) 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. 741)*
- **std::istream & operator>> (std::istream &i, mha\_complex\_t &c)**

*preliminary istream operator for mha\_complex\_t (p. 741) 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.*

### 6.162.1 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.

### 6.162.2 Macro Definition Documentation

#### 6.162.2.1 M\_PI #define M\_PI 3.14159265358979323846

**6.162.2.2 mha\_round** #define mha\_round(  
x ) (int)((float)x+0.5)

### 6.162.3 Function Documentation

**6.162.3.1 mha\_min\_1()** unsigned int mha\_min\_1 (  
unsigned int a ) [inline]

**6.162.3.2 value() [1/2]** mha\_real\_t& value (  
mha\_wave\_t \* s,  
unsigned int k ) [inline]

**6.162.3.3 value() [2/2]** mha\_complex\_t& value (  
mha\_spec\_t \* s,  
unsigned int k ) [inline]

**6.162.3.4 set\_minabs()** void set\_minabs (  
mha\_spec\_t & ,  
const mha\_real\_t & )

**6.162.3.5 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.

```
6.162.3.6 operator<<() std::ostream& operator<< (
    std::ostream & o,
    const mha_complex_t & c ) [inline]
```

ostream operator for **mha\_complex\_t** (p. 741)

```
6.162.3.7 operator>>() std::istream& operator>> (
    std::istream & i,
    mha_complex_t & c ) [inline]
```

preliminary istream operator for **mha\_complex\_t** (p. 741) without error checking

## 6.163 mha\_signal\_fft.h File Reference

### Classes

- class **MHASignal::fft\_t**

### Namespaces

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

## 6.164 mha\_tablelookup.cpp File Reference

## 6.165 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.*

### 6.165.1 Detailed Description

Header file for table lookup classes.

## 6.166 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**

### 6.166.1 Macro Definition Documentation

#### 6.166.1.1 INVALID\_SOCKET #define INVALID\_SOCKET (-1)

#### 6.166.1.2 SOCKET\_ERROR #define SOCKET\_ERROR (-1)

#### 6.166.1.3 closesocket #define closesocket( fd ) (close((fd)))

**6.166.1.4 ASYNC\_CONNECT\_STARTED** #define ASYNC\_CONNECT\_STARTED EINPROGRESS**6.166.2 Typedef Documentation****6.166.2.1 SOCKET** `typedef int SOCKET`**6.166.3 Function Documentation****6.166.3.1 host\_port\_to\_sock\_addr()** `static sockaddr_in host_port_to_sock_addr (`  
`const std::string & host,`  
`unsigned short port ) [static]`**6.166.3.2 tcp\_connect\_to()** `static SOCKET tcp_connect_to (`  
`const std::string & host,`  
`unsigned short port ) [static]`**6.166.3.3 tcp\_connect\_to\_with\_timeout()** `static SOCKET tcp_connect_to_with_timeout (`  
`const std::string & host,`  
`unsigned short port,`  
`Timeout_Watcher & timeout_watcher ) [static]`**6.166.3.4 thread\_start\_func()** `static void* thread_start_func (`  
`void * thread ) [static]`

## 6.167 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. 799) handles Communication between client and server, is used on both sides.*
- class **MHA\_TCP::Server**
- class **MHA\_TCP::Client**  
*A portable class for a tcp client connections.*
- class **MHA\_TCP::Thread**  
*A very simple class for portable threads.*

### Namespaces

- **MHA\_TCP**  
*A Namespace for TCP helper classes.*

### Macros

- #define **Sleep(x)** usleep((x)\*1000);

### Typedefs

- typedef int **MHA\_TCP::SOCKET**

## Functions

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

### 6.167.1 Macro Definition Documentation

#### 6.167.1.1 Sleep #define Sleep( x ) usleep((x)\*1000);

## 6.168 mha\_tcp\_server.cpp File Reference

### Namespaces

- **mha\_tcp**  
*namespace for network communication classes of MHA*

## 6.169 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*

**6.170 mha\_toolbox.h File Reference****6.171 mha\_utils.cpp File Reference****6.172 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.*

**6.173 mha\_windowparser.cpp File Reference****Variables**

- float(\*) **wnd\_funcs** [])(float)

### 6.173.1 Variable Documentation

#### 6.173.1.1 **wnd\_funcs** float(\* wnd\_funcs[ ]) (float)

## 6.174 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.*

**6.175 mhachain.cpp File Reference****Classes**

- class **mhachain::mhachain\_t**

**Namespaces**

- **mhachain**

**6.176 mhafw\_lib.cpp File Reference****6.177 mhafw\_lib.h File Reference****Classes**

- class **io\_lib\_t**  
*Class for loading MHA sound IO module.*
- class **fw\_vars\_t**
- class **fw\_t**

**6.178 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 \*, int err)
- void **IODestroy** (void \*handle)

## Variables

- static char **user\_err\_msg** [ **MAX\_USER\_ERR**]

### 6.178.1 Macro Definition Documentation

**6.178.1.1 `DBG`** `#define DBG(`  
`x ) fprintf(stderr,"%s:%d\n",__FILE__,__LINE__)`

**6.178.1.2 `ERR_SUCCESS`** `#define ERR_SUCCESS 0`

**6.178.1.3 `ERR_IHANDLE`** `#define ERR_IHANDLE -1`

**6.178.1.4 `ERR_USER`** `#define ERR_USER -1000`

**6.178.1.5 `MAX_USER_ERR`** `#define MAX_USER_ERR 0x500`

**6.178.1.6 `IOInit`** `#define IOInit MHA_STATIC_MHAIoalsa_IOInit`

**6.178.1.7 `IOPrepare`** `#define IOPrepare MHA_STATIC_MHAIoalsa_IOPrepare`

**6.178.1.8 `IOStart`** `#define IOStart MHA_STATIC_MHAIoalsa_IOStart`

**6.178.1.9 `IOStop`** `#define IOStop MHA_STATIC_MHAIoalsa_IOStop`

**6.178.1.10 IORelease** #define IORelease MHA\_STATIC\_MHAIOalsa\_IORelease

**6.178.1.11 IOSetVar** #define IOSetVar MHA\_STATIC\_MHAIOalsa\_IOSetVar

**6.178.1.12 IOStrError** #define IOStrError MHA\_STATIC\_MHAIOalsa\_IOStrError

**6.178.1.13 IODestroy** #define IODestroy MHA\_STATIC\_MHAIOalsa\_IODestroy

**6.178.1.14 dummy\_interface\_test** void dummy\_interface\_test MHA\_STATIC\_MHAIOalsa\_<→  
dummy\_interface\_test

## 6.178.2 Function Documentation

**6.178.2.1 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.
---------------	---

**6.178.2.2 IOPrepare()** int IOPrepare (

```
    void * handle,  
    int nch_in,  
    int nch_out )
```

IO library prepare function, called after the MHA prepared the processing plugins.

**6.178.2.3 IOStart()** int IOStart (

```
    void * handle )
```

**6.178.2.4 IOStop()** int IOStop (

```
    void * handle )
```

**6.178.2.5 IOReset()** int IOReset (

```
    void * handle )
```

**6.178.2.6 IOSetVar()** int IOSetVar (

```
    void * handle,  
    const char * command,  
    char * retval,  
    unsigned int maxretlen )
```

```
6.178.2.7 IOStrError() const char* IOStrError (
    void * ,
    int err )
```

```
6.178.2.8 IODestroy() void IODestroy (
    void * handle )
```

### 6.178.3 Variable Documentation

```
6.178.3.1 user_err_msg char user_err_msg[ MAX_USER_ERR] [static]
```

## 6.179 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**]

### 6.179.1 Macro Definition Documentation

#### 6.179.1.1 **ERR\_SUCCESS** #define **ERR\_SUCCESS** 0

#### 6.179.1.2 **ERR\_IHANDLE** #define **ERR\_IHANDLE** -1

**6.179.1.3 ERR\_USER** #define ERR\_USER -1000

**6.179.1.4 MAX\_USER\_ERR** #define MAX\_USER\_ERR 0x2000

**6.179.1.5 MHA\_ErrorMsg2** #define MHA\_ErrorMsg2(

```
    x,  
    y )  MHA_Error(__FILE__, __LINE__, (x), (y))
```

**6.179.1.6 MHA\_ErrorMsg3** #define MHA\_ErrorMsg3(

```
    x,  
    y,  
    z )  MHA_Error(__FILE__, __LINE__, (x), (y), (z))
```

**6.179.1.7 MIN\_TCP\_PORT** #define MIN\_TCP\_PORT 0

**6.179.1.8 MIN\_TCP\_PORT\_STR** #define MIN\_TCP\_PORT\_STR "0"

**6.179.1.9 MAX\_TCP\_PORT** #define MAX\_TCP\_PORT 65535

**6.179.1.10 MAX\_TCP\_PORT\_STR** #define MAX\_TCP\_PORT\_STR "65535"

**6.179.1.11 IOInit** #define IOInit MHA\_STATIC\_MHAIOTCP\_IOInit

**6.179.1.12 IOPrepare** #define IOPrepare MHA\_STATIC\_MHAIOTCP\_IOPrepare

**6.179.1.13 IOStart** #define IOStart MHA\_STATIC\_MHAIOTCP\_IOStart

**6.179.1.14 IOStop** #define IOStop MHA\_STATIC\_MHAIOTCP\_IOStop

**6.179.1.15 IOReset** #define IOReset MHA\_STATIC\_MHAIOTCP\_IOReset

**6.179.1.16 IOSetVar** #define IOSetVar MHA\_STATIC\_MHAIOTCP\_IOSetVar

**6.179.1.17 IOStrError** #define IOStrError MHA\_STATIC\_MHAIOTCP\_IOStrError

**6.179.1.18 IODestroy** #define IODestroy MHA\_STATIC\_MHAIOTCP\_IODestroy

**6.179.1.19 dummy\_interface\_test** #define dummy\_interface\_test MHA\_STATIC\_MHAIOTC←  
P\_dummy\_interface\_test

## 6.179.2 Function Documentation

**6.179.2.1 copy\_error()** static int copy\_error (   
   **MHA\_Error** & e ) [static]

**6.179.2.2 thread\_startup\_function()** static void\* thread\_startup\_function (   
   void \* parameter ) [static]

**6.179.2.3 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 )

**6.179.2.4 IOPrepare()** int IOPrepare (   
   void \* handle,   
   int num\_inchannels,   
   int num\_outchannels )

**6.179.2.5 IOStart()** int IOStart (   
   void \* handle )

**6.179.2.6 IOStop()** int IOStop (   
   void \* handle )

**6.179.2.7 IORelease()** int IORelease (   
 void \* *handle* )

**6.179.2.8 IOSetVar()** int IOSetVar (   
 void \* *handle*,  
 const char \* *cmd*,  
 char \* *retval*,  
 unsigned int *len* )

**6.179.2.9 IOStrError()** const char\* IOStrError (   
 void \* *handle*,  
 int *err* )

**6.179.2.10 IODestroy()** void IODestroy (   
 void \* *handle* )

### 6.179.3 Variable Documentation

**6.179.3.1 user\_err\_msg** char user\_err\_msg[ **MAX\_USER\_ERR**] [static]

## 6.180 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 \*, int err)
- void **IODestroy** (void \*handle)

## Variables

- static char **user\_err\_msg** [ **MAX\_USER\_ERR** ]

### 6.180.1 Macro Definition Documentation

```
6.180.1.1 DEBUG #define DEBUG(
    x ) std::cerr << __FILE__ << ":" << __LINE__ << " " << #x " = " <<
x << std::endl
```

**6.180.1.2 ERR\_SUCCESS** #define ERR\_SUCCESS 0

**6.180.1.3 ERR\_IHANDLE** #define ERR\_IHANDLE -1

**6.180.1.4 ERR\_USER** #define ERR\_USER -1000

**6.180.1.5 MAX\_USER\_ERR** #define MAX\_USER\_ERR 0x500

**6.180.1.6 IOInit** #define IOInit MHA\_STATIC\_MHAIOFile\_IOInit

**6.180.1.7 IOPrepare** #define IOPrepare MHA\_STATIC\_MHAIOFile\_IOPrepare

**6.180.1.8 IOStart** #define IOStart MHA\_STATIC\_MHAIOFile\_IOStart

**6.180.1.9 IOStop** #define IOStop MHA\_STATIC\_MHAIOFile\_IOStop

**6.180.1.10 IOReset** #define IOReset MHA\_STATIC\_MHAIOFile\_IOReset

**6.180.1.11 IOSetVar** #define IOSetVar MHA\_STATIC\_MHAIOFile\_IOSetVar

**6.180.1.12 IOStrError** #define IOStrError MHA\_STATIC\_MHAIOFile\_IOStrError

**6.180.1.13 IODestroy** #define IODestroy MHA\_STATIC\_MHAIOFile\_IODestroy

**6.180.1.14 dummy\_interface\_test** #define dummy\_interface\_test MHA\_STATIC\_MHAIO←  
File\_dummy\_interface\_test

## 6.180.2 Function Documentation

**6.180.2.1 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 )
```

**6.180.2.2 IOPrepare()** int IOPrepare (

```
    void * handle,
    int nch_in,
    int nch_out )
```

**6.180.2.3 IOStart()** int IOStart (   
 void \* *handle* )

**6.180.2.4 IOStop()** int IOStop (   
 void \* *handle* )

**6.180.2.5 IOReset()** int IOReset (   
 void \* *handle* )

**6.180.2.6 IOSetVar()** int IOSetVar (   
 void \* *handle*,  
 const char \* *command*,  
 char \* *retval*,  
 unsigned int *maxretlen* )

**6.180.2.7 IOStrError()** const char\* IOStrError (   
 void \* ,  
 int *err* )

**6.180.2.8 IODestroy()** void IODestroy (   
 void \* *handle* )

### 6.180.3 Variable Documentation

**6.180.3.1 user\_err\_msg** char user\_err\_msg[ **MAX\_USER\_ERR**] [static]

## 6.181 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 \*, int err)
- void **IODestroy** (void \*handle)

## Variables

- static char **user\_err\_msg** [ **MAX\_USER\_ERR**] = ""

### 6.181.1 Macro Definition Documentation

#### 6.181.1.1 **ERR\_SUCCESS** #define ERR\_SUCCESS 0

#### 6.181.1.2 **ERR\_IHANDLE** #define ERR\_IHANDLE -1

#### 6.181.1.3 **ERR\_USER** #define ERR\_USER -1000

#### 6.181.1.4 **MAX\_USER\_ERR** #define MAX\_USER\_ERR 0x500

#### 6.181.1.5 **IOInit** #define IOInit MHA\_STATIC\_MHAIOJack\_IOInit

#### 6.181.1.6 **IOPrepare** #define IOPrepare MHA\_STATIC\_MHAIOJack\_IOPrepare

#### 6.181.1.7 **IOStart** #define IOStart MHA\_STATIC\_MHAIOJack\_IOStart

**6.181.1.8 IOStop** #define IOStop MHA\_STATIC\_MHAIOJack\_IOStop

**6.181.1.9 IOReset** #define IOReset MHA\_STATIC\_MHAIOJack\_IOReset

**6.181.1.10 IOSetVar** #define IOSetVar MHA\_STATIC\_MHAIOJack\_IOSetVar

**6.181.1.11 IOStrError** #define IOStrError MHA\_STATIC\_MHAIOJack\_IOStrError

**6.181.1.12 IODestroy** #define IODestroy MHA\_STATIC\_MHAIOJack\_IODestroy

**6.181.1.13 dummy\_interface\_test** #define dummy\_interface\_test MHA\_STATIC\_MHAIOJack\_dummy\_interface\_test

## 6.181.2 Function Documentation

**6.181.2.1 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 )
```

**6.181.2.2 IOPrepare()** int IOPrepare (

```
void * handle,
int nch_in,
int nch_out )
```

**6.181.2.3 IOStart()** int IOStart (

```
void * handle )
```

**6.181.2.4 IOStop()** int IOStop (

```
void * handle )
```

**6.181.2.5 IOResearch()** int IOResearch (

```
void * handle )
```

**6.181.2.6 IOSetVar()** int IOSetVar (

```
void * handle,
const char * command,
char * retval,
unsigned int maxretlen )
```

**6.181.2.7 IOStrError()** const char\* IOStrError (

```
void * ,
int err )
```

**6.181.2.8 IODestroy()** void IODestroy (

```
void * handle )
```

### 6.181.3 Variable Documentation

**6.181.3.1 user\_err\_msg** char user\_err\_msg[ **MAX\_USER\_ERR**] = "" [static]

## 6.182 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 **IOResume** MHA\_STATIC\_MHAIOJackdb\_IOResume
- #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 **IOResume** (void \*handle)
- int **IOSetVar** (void \*handle, const char \*command, char \*retval, unsigned int maxretlen)
- const char \* **IOStrError** (void \*, int err)
- void **IODestroy** (void \*handle)

## Variables

- static char **user\_err\_msg** [ **MAX\_USER\_ERR**] = ""

### 6.182.1 Macro Definition Documentation

#### 6.182.1.1 **ERR\_SUCCESS** #define ERR\_SUCCESS 0

#### 6.182.1.2 **ERR\_IHANDLE** #define ERR\_IHANDLE -1

#### 6.182.1.3 **ERR\_USER** #define ERR\_USER -1000

#### 6.182.1.4 **MAX\_USER\_ERR** #define MAX\_USER\_ERR 0x500

#### 6.182.1.5 **IOInit** #define IOInit MHA\_STATIC\_MHAIOJackdb\_IOInit

#### 6.182.1.6 **IOPrepare** #define IOPrepare MHA\_STATIC\_MHAIOJackdb\_IOPrepare

#### 6.182.1.7 **IOStart** #define IOStart MHA\_STATIC\_MHAIOJackdb\_IOStart

**6.182.1.8 IOStop** #define IOStop MHA\_STATIC\_MHAIOJackdb\_IOStop

**6.182.1.9 IOReset** #define IOReset MHA\_STATIC\_MHAIOJackdb\_IOReset

**6.182.1.10 IOSetVar** #define IOSetVar MHA\_STATIC\_MHAIOJackdb\_IOSetVar

**6.182.1.11 IOStrError** #define IOStrError MHA\_STATIC\_MHAIOJackdb\_IOStrError

**6.182.1.12 IODestroy** #define IODestroy MHA\_STATIC\_MHAIOJackdb\_IODestroy

**6.182.1.13 dummy\_interface\_test** #define dummy\_interface\_test MHA\_STATIC\_MHAIOJackdb\_dummy\_interface\_test

## 6.182.2 Function Documentation

**6.182.2.1 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 )
```

**6.182.2.2 IOPrepare()** int IOPrepare (

```
void * handle,
int nch_in,
int nch_out )
```

**6.182.2.3 IOStart()** int IOStart (

```
void * handle )
```

**6.182.2.4 IOStop()** int IOStop (

```
void * handle )
```

**6.182.2.5 IOReset()** int IOReset (

```
void * handle )
```

**6.182.2.6 IOSetVar()** int IOSetVar (

```
void * handle,
const char * command,
char * retval,
unsigned int maxretlen )
```

**6.182.2.7 IOStrError()** const char\* IOStrError (

```
void * ,
int err )
```

**6.182.2.8 IODestroy()** void IODestroy (

```
void * handle )
```

### 6.182.3 Variable Documentation

**6.182.3.1 user\_err\_msg** char user\_err\_msg[ **MAX\_USER\_ERR**] = "" [static]

## 6.183 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\_IORelease
- #define **IOSetVar** MHA\_STATIC\_MHAIOParser\_IOSetVar
- #define **IOStrError** MHA\_STATIC\_MHAIOParser\_IOStrError
- #define **IODestroy** MHA\_STATIC\_MHAIOParser\_IODestroy
- #define **dummy\_interface\_test** MHA\_STATIC\_MHAIOParser\_dummy\_interface\_test

### Functions

- int **IOInit** (int fragsize, float, **IOProcessEvent\_t** proc\_event, void \*proc\_handle, **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 \*, int err)
- void **IODestroy** (void \*handle)

## Variables

- static char **user\_err\_msg** [ **MAX\_USER\_ERR**]

### 6.183.1 Macro Definition Documentation

#### 6.183.1.1 **ERR\_SUCCESS** #define ERR\_SUCCESS 0

#### 6.183.1.2 **ERR\_IHANDLE** #define ERR\_IHANDLE -1

#### 6.183.1.3 **ERR\_USER** #define ERR\_USER -1000

#### 6.183.1.4 **MAX\_USER\_ERR** #define MAX\_USER\_ERR 0x500

#### 6.183.1.5 **IOInit** #define IOInit MHA\_STATIC\_MHAIOParser\_IOInit

#### 6.183.1.6 **IOPrepare** #define IOPrepare MHA\_STATIC\_MHAIOParser\_IOPrepare

#### 6.183.1.7 **IOStart** #define IOStart MHA\_STATIC\_MHAIOParser\_IOStart

**6.183.1.8 IOStop** #define IOStop MHA\_STATIC\_MHAIOParser\_IOStop

**6.183.1.9 IOReset** #define IOReset MHA\_STATIC\_MHAIOParser\_IOReset

**6.183.1.10 IOSetVar** #define IOSetVar MHA\_STATIC\_MHAIOParser\_IOSetVar

**6.183.1.11 IOStrError** #define IOStrError MHA\_STATIC\_MHAIOParser\_IOStrError

**6.183.1.12 IODestroy** #define IODestroy MHA\_STATIC\_MHAIOParser\_IODestroy

**6.183.1.13 dummy\_interface\_test** #define dummy\_interface\_test MHA\_STATIC\_MHAIOParser\_dummy\_interface\_test

## 6.183.2 Function Documentation

**6.183.2.1 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 )
```

**6.183.2.2 IOPrepare()** int IOPrepare (

```
void * handle,
int nch_in,
int nch_out )
```

**6.183.2.3 IOStart()** int IOStart (

```
void * handle )
```

**6.183.2.4 IOStop()** int IOStop (

```
void * handle )
```

**6.183.2.5 IOResearch()** int IOResearch (

```
void * handle )
```

**6.183.2.6 IOSetVar()** int IOSetVar (

```
void * handle,
const char * command,
char * retval,
unsigned int maxretlen )
```

**6.183.2.7 IOStrError()** const char\* IOStrError (

```
void * ,
int err )
```

**6.183.2.8 IODestroy()** void IODestroy (

```
void * handle )
```

### 6.183.3 Variable Documentation

#### 6.183.3.1 `user_err_msg` `char user_err_msg[ MAX_USER_ERR ] [static]`

## 6.184 MHAIOPortAudio.cpp File Reference

### Classes

- class `MHAIOPortAudio::stream_info_t`
- class `MHAIOPortAudio::device_info_t`
- class `MHAIOPortAudio::io_portaudio_t`

*Main class for Portaudio sound IO.*

### Namespaces

- `MHAIOPortAudio`

### Macros

- `#define ERR_SUCCESS 0`
- `#define ERR_IHANDLE -1`
- `#define ERR_USER -1000`
- `#define MAX_USER_ERR 0x500`
- `#define IOInit MHA_STATIC_MHAIOPortAudio_IOInit`
- `#define IOPrepare MHA_STATIC_MHAIOPortAudio_IOPrepare`
- `#define IOStart MHA_STATIC_MHAIOPortAudio_IOStart`
- `#define IOStop MHA_STATIC_MHAIOPortAudio_IOStop`
- `#define IOReset MHA_STATIC_MHAIOPortAudio_IOReset`
- `#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 \*, int err)
- void **IODestroy** (void \*handle)

## Variables

- static char **user\_err\_msg** [ **MAX\_USER\_ERR**] = ""
- PaStreamCallback **portaudio\_callback**

### 6.184.1 Macro Definition Documentation

#### 6.184.1.1 **ERR\_SUCCESS** #define ERR\_SUCCESS 0

#### 6.184.1.2 **ERR\_IHANDLE** #define ERR\_IHANDLE -1

#### 6.184.1.3 **ERR\_USER** #define ERR\_USER -1000

#### 6.184.1.4 **MAX\_USER\_ERR** #define MAX\_USER\_ERR 0x500

**6.184.1.5 IOInit** #define IOInit MHA\_STATIC\_MHAIOPortAudio\_IOInit

**6.184.1.6 IOPrepare** #define IOPrepare MHA\_STATIC\_MHAIOPortAudio\_IOPrepare

**6.184.1.7 IOStart** #define IOStart MHA\_STATIC\_MHAIOPortAudio\_IOStart

**6.184.1.8 IOStop** #define IOStop MHA\_STATIC\_MHAIOPortAudio\_IOStop

**6.184.1.9 IOReset** #define IOReset MHA\_STATIC\_MHAIOPortAudio\_IOReset

**6.184.1.10 IOSetVar** #define IOSetVar MHA\_STATIC\_MHAIOPortAudio\_IOSetVar

**6.184.1.11 IOStrError** #define IOStrError MHA\_STATIC\_MHAIOPortAudio\_IOStrError

**6.184.1.12 IODestroy** #define IODestroy MHA\_STATIC\_MHAIOPortAudio\_IODestroy

**6.184.1.13 dummy\_interface\_test** #define dummy\_interface\_test MHA\_STATIC\_MHAIOPortAudio\_dummy\_interface\_test

---

## 6.184.2 Function Documentation

**6.184.2.1 portaudio\_callback()** int portaudio\_callback (

```
const void * input,
void * output,
unsigned long frameCount,
const PaStreamCallbackTimeInfo * timeInfo,
PaStreamCallbackFlags statusFlags,
void * userData )
```

**6.184.2.2 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 )
```

**6.184.2.3 IOPrepare()** int IOPrepare (

```
void * handle,
int nch_in,
int nch_out )
```

**6.184.2.4 IOStart()** int IOStart (

```
void * handle )
```

**6.184.2.5 IOStop()** int IOStop (

```
void * handle )
```

**6.184.2.6 IORelease()** int IORelease (

```
void * handle )
```

**6.184.2.7 IOSetVar()** int IOSetVar (

```
void * handle,
const char * command,
char * retval,
unsigned int maxretlen )
```

**6.184.2.8 IOStrError()** const char\* IOStrError (

```
void * ,
int err )
```

**6.184.2.9 IODestroy()** void IODestroy (

```
void * handle )
```

### 6.184.3 Variable Documentation

**6.184.3.1 user\_err\_msg** char user\_err\_msg[ **MAX\_USER\_ERR**] = "" [static]

**6.184.3.2 portaudio\_callback** PaStreamCallback portaudio\_callback

---

## 6.185 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**]

### 6.185.1 Macro Definition Documentation

#### 6.185.1.1 **ERR\_SUCCESS** #define ERR\_SUCCESS 0

#### 6.185.1.2 **ERR\_IHANDLE** #define ERR\_IHANDLE -1

#### 6.185.1.3 **ERR\_USER** #define ERR\_USER -1000

#### 6.185.1.4 **MAX\_USER\_ERR** #define MAX\_USER\_ERR 0x2000

**6.185.1.5 MHA\_ErrorMsg2** #define MHA\_ErrorMsg2(

```
  x,
  y )  MHA_Error(__FILE__, __LINE__, (x), (y))
```

**6.185.1.6 MHA\_ErrorMsg3** #define MHA\_ErrorMsg3(

```
  x,
  y,
  z )  MHA_Error(__FILE__, __LINE__, (x), (y), (z))
```

**6.185.1.7 MIN\_TCP\_PORT** #define MIN\_TCP\_PORT 0

**6.185.1.8 MIN\_TCP\_PORT\_STR** #define MIN\_TCP\_PORT\_STR "0"

**6.185.1.9 MAX\_TCP\_PORT** #define MAX\_TCP\_PORT 65535

**6.185.1.10 MAX\_TCP\_PORT\_STR** #define MAX\_TCP\_PORT\_STR "65535"

**6.185.1.11 IOInit** #define IOInit MHA\_STATIC\_MHAIOTCP\_IOInit

**6.185.1.12 IOPrepare** #define IOPrepare MHA\_STATIC\_MHAIOTCP\_IOPrepare

**6.185.1.13 IOStart** #define IOStart MHA\_STATIC\_MHAIOTCP\_IOStart

**6.185.1.14 IOStop** #define IOStop MHA\_STATIC\_MHAIOTCP\_IOStop

**6.185.1.15 IOReset** #define IOReset MHA\_STATIC\_MHAIOTCP\_IOReset

**6.185.1.16 IOSetVar** #define IOSetVar MHA\_STATIC\_MHAIOTCP\_IOSetVar

**6.185.1.17 IOStrError** #define IOStrError MHA\_STATIC\_MHAIOTCP\_IOStrError

**6.185.1.18 IODestroy** #define IODestroy MHA\_STATIC\_MHAIOTCP\_IODestroy

**6.185.1.19 dummy\_interface\_test** #define dummy\_interface\_test MHA\_STATIC\_MHAIOTCP\_dummy\_interface\_test

## 6.185.2 Function Documentation

**6.185.2.1 copy\_error()** static int copy\_error (  
    **MHA\_Error** & e ) [static]

**6.185.2.2 `thread_startup_function()`** static void\* thread\_startup\_function ( void \* parameter ) [static]

**6.185.2.3 `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 )

**6.185.2.4 `IOPrepare()`** int IOPrepare ( void \* handle, int num\_inchannels, int num\_outchannels )

**6.185.2.5 `IOStart()`** int IOStart ( void \* handle )

**6.185.2.6 `IOStop()`** int IOStop ( void \* handle )

**6.185.2.7 `IOResale()`** int IOResale ( void \* handle )

```
6.185.2.8 IOSetVar() int IOSetVar (
    void * handle,
    const char * cmd,
    char * retval,
    unsigned int len )
```

```
6.185.2.9 IOStrError() const char* IOStrError (
    void * handle,
    int err )
```

```
6.185.2.10 IODestroy() void IODestroy (
    void * handle )
```

### 6.185.3 Variable Documentation

```
6.185.3.1 user_err_msg char user_err_msg[ MAX_USER_ERR] [static]
```

## 6.186 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

### 6.186.1 Function Documentation

**6.186.1.1 `jack_error_handler()`** static void jack\_error\_handler ( const char \* msg ) [static]

**6.186.1.2 `dummy_jack_proc_cb()`** static int dummy\_jack\_proc\_cb ( jack\_nframes\_t , void \* ) [static]

**6.186.1.3 `make_friendly_number()`** void make\_friendly\_number ( jack\_default\_audio\_sample\_t & x ) [inline]

## 6.186.2 Variable Documentation

**6.186.2.1 `last_jack_err_msg`** char last\_jack\_err\_msg[ MAX\_USER\_ERR ] = ""

**6.186.2.2 `last_jack_err`** int last\_jack\_err = 0

## 6.187 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**]

### 6.187.1 Macro Definition Documentation

**6.187.1.1 MHAJACK\_FW\_STARTED** #define MHAJACK\_FW\_STARTED 1

**6.187.1.2 MHAJACK\_STOPPED** #define MHAJACK\_STOPPED 2

**6.187.1.3 MHAJACK\_STARTING** #define MHAJACK\_STARTING 8

**6.187.1.4 IO\_ERROR\_JACK** #define IO\_ERROR\_JACK 11

**6.187.1.5 IO\_ERROR\_MHAJACKLIB** #define IO\_ERROR\_MHAJACKLIB 12

**6.187.1.6 MAX\_USER\_ERR** #define MAX\_USER\_ERR 0x500

## 6.187.2 Variable Documentation

**6.187.2.1 last\_jack\_err\_msg** char last\_jack\_err\_msg[ MAX\_USER\_ERR]

## 6.188 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

- int **mhamain** (int argc, char \*argv[ ])

### 6.188.1 Macro Definition Documentation

#### 6.188.1.1 **HELP\_TEXT** #define HELP\_TEXT

#### 6.188.1.2 **NORELEASE\_WARNING** #define NORELEASE\_WARNING

#### 6.188.1.3 **VERSION\_EXTENSION** #define VERSION\_EXTENSION "+"

#### 6.188.1.4 **GREETING\_TEXT** #define GREETING\_TEXT

### 6.188.2 Function Documentation

#### 6.188.2.1 **mhamain()** int mhamain (

```
    int argc,
    char * argv[ ] )
```

## 6.189 mhapluginloader.cpp File Reference

### 6.190 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. 847) structures, and report differences as an error.*

## 6.191 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)

### 6.191.1 Function Documentation

```
6.191.1.1 write_wave() void write_wave (
    const mha_wave_t & sig,
    const char * fname,
    const float & srate,
    const int & format )
```

```
6.191.1.2 validator_channels() unsigned int validator_channels (
    std::vector< int > channel_map,
    unsigned int channels )
```

```
6.191.1.3 validator_length() unsigned int validator_length (
    unsigned int maxlen,
    unsigned int frames,
    unsigned int startpos )
```

## 6.192 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)

### 6.192.1 Function Documentation

```
6.192.1.1 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 )
```

## 6.193 multibandcompressor.cpp File Reference

### Classes

- class **multibandcompressor::plugin\_signals\_t**
- class **multibandcompressor::ffftfb\_plug\_t**
- class **multibandcompressor::interface\_t**

### Namespaces

- **multibandcompressor**

## 6.194 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)

### 6.194.1 Macro Definition Documentation

**6.194.1.1 NORMALIZATION\_TYPES** `#define NORMALIZATION_TYPES "[none default sum]"`

**6.194.1.2 NORM\_NONE** `#define NORM_NONE 0`

**6.194.1.3 NORM\_DEFAULT** `#define NORM_DEFAULT 1`

**6.194.1.4 NORM\_SUM** `#define NORM_SUM 2`

**6.194.1.5 ESTIMATION\_TYPES** `#define ESTIMATION_TYPES "[previous current]"`

**6.194.1.6 ESTIM\_PREV** `#define ESTIM_PREV 0`

**6.194.1.7 ESTIM\_CUR** `#define ESTIM_CUR 1`

### 6.194.2 Function Documentation

**6.194.2.1 make\_friendly\_number\_by\_limiting()** void make\_friendly\_number\_by\_limiting  
(  
    mha\_real\_t & x ) [inline]

## 6.195 noise.cpp File Reference

### Classes

- class **cfg\_t**
- class **noise\_t**

## 6.196 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

## 6.196.1 Macro Definition Documentation

### 6.196.1.1 POWSPEC\_FACTOR #define POWSPEC\_FACTOR 0.0025

## 6.197 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**

## 6.198 overlapadd.cpp File Reference

### Namespaces

- **overlapadd**

## 6.199 overlapadd.hh File Reference

### Classes

- class **overlapadd::overlapadd\_t**
- class **overlapadd::overlapadd\_if\_t**

### Namespaces

- **overlapadd**

## 6.200 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)

## 6.201 pluginbrowser.cpp File Reference

### 6.202 pluginbrowser.h File Reference

#### Classes

- class `plugindescription_t`
- class `pluginloader_t`
- class `pluginbrowser_t`

## 6.203 proc\_counter.cpp File Reference

#### Classes

- class `proc_counter_t`

## 6.204 resampling.cpp File Reference

#### Classes

- class `MHAPlugIn_Resampling::resampling_t`
- class `MHAPlugIn_Resampling::resampling_if_t`

#### Namespaces

- `MHAPlugIn_Resampling`

## 6.205 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 }

## 6.206 rohBeam.cpp File Reference

### Namespaces

- **rohBeam**

### Variables

- auto **rohBeam::scalarify** =[ ](auto t){return t(0);}

## 6.207 rohBeam.hh File Reference

### Classes

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

### Namespaces

- **rohBeam**

### Macros

- #define **NDEBUG**

### Functions

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

## Variables

- constexpr float `rohBeam::CONST_C` = 343.0115f
- constexpr int `rohBeam::refL` = 0
- constexpr int `rohBeam::refR` = 3

## 6.207.1 Macro Definition Documentation

### 6.207.1.1 `NDEBUG` #define NDEBUG

## 6.208 route.cpp File Reference

### Classes

- class `route::process_t`
- class `route::interface_t`

### Namespaces

- `route`

## 6.209 save\_spec.cpp File Reference

### Classes

- class `save_spec_t`

## 6.210 save\_wave.cpp File Reference

### Classes

- class `save_wave_t`

## 6.211 shadowfilter\_begin.cpp File Reference

### Classes

- class `shadowfilter_begin::cfg_t`
- class `shadowfilter_begin::shadowfilter_begin_t`

### Namespaces

- `shadowfilter_begin`

## 6.212 shadowfilter\_end.cpp File Reference

### Classes

- class `shadowfilter_end::cfg_t`
- class `shadowfilter_end::shadowfilter_end_t`

### Namespaces

- `shadowfilter_end`

## 6.213 sine.cpp File Reference

### Classes

- struct `sine_cfg_t`  
*Runtime configuration of the sine plugin.*
- class `sine_t`  
*Interface class of plugin sine, a sinusoid generator plugin.*

## 6.214 smooth\_cepstrum.cpp File Reference

### Macros

- #define `INSERT_VAR(var)` `insert_item(#var, &var)`
- #define `PATCH_VAR(var)`
- #define `INSERT_PATCH(var)` `INSERT_VAR(var); PATCH_VAR(var)`

## 6.214.1 Macro Definition Documentation

### 6.214.1.1 **INSERT\_VAR** #define INSERT\_VAR( var ) insert\_item(#var, &var)

### 6.214.1.2 **PATCH\_VAR** #define PATCH\_VAR( var )

### 6.214.1.3 **INSERT\_PATCH** #define INSERT\_PATCH( var ) INSERT\_VAR(var); PATCH\_VAR(var)

## 6.215 smooth\_cepstrum.hh File Reference

### Classes

- class **smooth\_cepstrum::smooth\_params**
- class **smooth\_cepstrum::smooth\_cepstrum\_t**
- class **smooth\_cepstrum::smooth\_cepstrum\_if\_t**

### Namespaces

- **smooth\_cepstrum**

## 6.216 smoothgains\_bridge.cpp File Reference

### Classes

- class **smoothgains\_bridge::smoothspec\_wrap\_t**
- class **smoothgains\_bridge::overlapadd\_if\_t**

### Namespaces

- **smoothgains\_bridge**

## 6.217 softclip.cpp File Reference

### Classes

- class `cfg_t`
- class `softclip_t`

## 6.218 spec2wave.cpp File Reference

### Classes

- class `hanning_ramps_t`
- class `spec2wave_t`
- class `spec2wave_if_t`

### Functions

- unsigned int `max` (unsigned int a, unsigned int b)
- unsigned int `min` (unsigned int a, unsigned int b)

### 6.218.1 Function Documentation

**6.218.1.1 `max()`** `unsigned int max (`  
    `unsigned int a,`  
    `unsigned int b ) [inline]`

**6.218.1.2 `min()`** `unsigned int min (`  
    `unsigned int a,`  
    `unsigned int b ) [inline]`

## 6.219 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}

### 6.219.1 Macro Definition Documentation

**6.219.1.1 NUM\_ENTR\_MHAORIG** #define NUM\_ENTR\_MHAORIG 76

**6.219.1.2 NUM\_ENTR\_LTASS** #define NUM\_ENTR\_LTASS 25

**6.219.1.3 NUM\_ENTR\_OLNOISE** #define NUM\_ENTR\_OLNOISE 49

### 6.219.2 Function Documentation

**6.219.2.1 fhz2bandno()** float fhz2bandno ( float x )

**6.219.2.2 erb\_hz\_f\_hz()** float erb\_hz\_f\_hz ( float f\_hz )

**6.219.2.3 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

**6.219.2.4 bandw\_correction()** float bandw\_correction ( float *f*, float *ldb* )

**6.219.3 Variable Documentation**

**6.219.3.1 vMHAOrigSpec** float vMHAOrigSpec[ **NUM\_ENTR\_MHAORIG** ] = { -1.473, 0, -4.←  
 939, -10.14, -13.94, -14.83, -14.27, -15.66, -16.16, -18.22, -20.5, -21.23, -22.13,  
 -22.58, -23.98, -26.58, -26.4, -25.15, -23.89, -25.54, -27, -30.15, -31.68, -30.←  
 14, -27.55, -25.79, -25.89, -26.11, -27.48, -30.37, -33.13, -36.23, -36.64, -36.←  
 35, -35.03, -35.48, -36.35, -37.95, -40.53, -42.37, -41.29, -38.49, -36.32, -34.85,  
 -34.05, -33.81, -33.48, -34.1, -35.19, -36.29, -36.94, -37.53, -38.71, -38.7, -38.←  
 92, -40.36, -41.26, -42.19, -43.65, -44.37, -43.95, -43.15, -42.57, -41.57, -41.←  
 86, -42.34, -42.87, -42.35, -42.71, -42.85, -43.47, -47.43, -67.54, -76.3, -77.43,  
 -77.43 }

**6.219.3.2 vMHAOrigFreq** float vMHAOrigFreq[ **NUM\_ENTR\_MHAORIG** ] = { 172.266, 344.←  
 532, 516.797, 689.063, 861.329, 1033.59, 1205.86, 1378.13, 1550.39, 1722.66, 1894.92, 2067.←  
 19, 2239.46, 2411.72, 2583.99, 2756.25, 2928.52, 3100.78, 3273.05, 3445.32, 3617.58, 3789.←  
 85, 3962.11, 4134.38, 4306.64, 4478.91, 4651.18, 4823.44, 4995.71, 5167.97, 5340.24, 5512.←  
 51, 5684.77, 5857.04, 6029.3, 6201.57, 6373.83, 6546.1, 6718.37, 6890.63, 7062.9, 7235.16, 7407.←  
 43, 7579.69, 7751.96, 7924.23, 8096.49, 8268.76, 8441.02, 8613.29, 8785.56, 8957.82, 9130.←  
 09, 9302.35, 9474.62, 9646.88, 9819.15, 9991.42, 10163.7, 10335.9, 10508.2, 10680.5, 10852.←  
 7, 11025, 11197.3, 11369.5, 11541.8, 11714.1, 11886.3, 12058.6, 12230.9, 12403.1, 12575.←  
 4, 12747.7, 12919.9, 13092.2 }

**6.219.3.3 vLTASS\_freq** float vLTASS\_freq[ **NUM\_ENTR\_LTASS** ] = { 63, 80, 100, 125, 160, 200, 250, 315, 400, 500, 630, 800, 1000, 1250, 1600, 2000, 2500, 3150, 4000, 5000, 6300, 8000, 10000, 12500, 16000 }

**6.219.3.4 vLTASS\_combined\_lev** float vLTASS\_combined\_lev[ **NUM\_ENTR\_LTASS** ] = { 38. ← 6, 43.5, 54.4, 57.7, 56.8, 60.2, 60.3, 59.0, 62.1, 62.1, 60.5, 56.8, 53.7, 53.0, 52.0, 48.7, 48.1, 46.8, 45.6, 44.5, 44.3, 43.7, 43.4, 41.3, 40.7 }

**6.219.3.5 vLTASS\_female\_lev** float vLTASS\_female\_lev[ **NUM\_ENTR\_LTASS** ] = { 37. ← 0, 36.0, 37.5, 40.1, 53.4, 62.2, 60.9, 58.1, 61.7, 61.7, 60.4, 58, 54.3, 52.3, 51.7, 48.8, 47. ← 3, 46.7, 45.3, 44.6, 45.2, 44.9, 45.0, 42.8, 41.1 }

**6.219.3.6 vLTASS\_male\_lev** float vLTASS\_male\_lev[ **NUM\_ENTR\_LTASS** ] = { 38.6, 43. ← 5, 54.4, 57.7, 56.8, 58.2, 59.7, 60.0, 62.4, 62.6, 60.6, 55.7, 53.1, 53.7, 52.3, 48.7, 48.9, 47. ← 0, 46.0, 44.4, 43.3, 42.4, 41.9, 39.8, 40.4 }

**6.219.3.7 vOlnoiseFreq** float vOlnoiseFreq[ **NUM\_ENTR\_OLNOISE** ] = { 62.5, 70.1539, 78. ← 7451, 88.3884, 99.2126, 111.362, 125, 140.308, 157.49, 176.777, 198.425, 222.725, 250, 280. ← 616, 314.98, 353.553, 396.85, 445.449, 500, 561.231, 629.961, 707.107, 793.701, 890.899, 1000, 1122. ← 46, 1259.92, 1414.21, 1587.4, 1781.8, 2000, 2244.92, 2519.84, 2828.43, 3174.8, 3563.59, 4000, 4489. ← 85, 5039.68, 5656.85, 6349.6, 7127.19, 8000, 8979.7, 10079.4, 11313.7, 12699.2, 14254.4, 16000 }

**6.219.3.8 vOlnoiseLev** float vOlnoiseLev[ **NUM\_ENTR\_OLNOISE** ] = { 45.9042, 38.044, 48. ← 9444, 61.3697, 67.6953, 69.7451, 71.6201, 71.2431, 65.2754, 63.2547, 70.2264, 72.1434, 73. ← 4433, 73.2659, 69.8424, 71.0132, 70.9577, 70.3492, 68.691, 64.8436, 64.0435, 64.2879, 60. ← 5889, 60.6596, 60.3727, 61.2003, 61.8477, 61.1478, 61.2312, 58.6584, 57.2892, 56.8299, 56. ← 0191, 53.3018, 56.0525, 54.3592, 50.8823, 55.992, 54.6768, 47.2616, 46.9914, 45.209, 50. ← 413, 47.5848, 43.3215, 43.754, 38.5773, -0.39427, 5.74224 }

## 6.220 speechnoise.h File Reference

### Classes

- class **speechnoise\_t**

## 6.221 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. 1179) instance manages the plugin that performs processing on the reduced set of channels.*
- class **MHAPlugin\_Split::split\_t**  
*Implements split plugin.*

### Namespaces

- **MHAPlugin\_Split**

### Macros

- #define **MHAPLUGIN\_OVERLOAD\_OUTDOMAIN**  
*This define modifies the definition of MHAPLUGIN\_CALLBACKS and friends.*
- #define **posixthreads** 1
- #define **native\_thread\_platform\_type** posix\_threads\_t

### Enumerations

- enum { **MHAPlugin\_Split::INVALID\_THREAD\_PRIORITY** = 999999999 }  
*Invalid thread priority.*

### 6.221.1 Detailed Description

Source code for the split plugin. The split plugin splits the audio signal by channel. The splitted paths execute in parallel.

### 6.221.2 Macro Definition Documentation

**6.221.2.1 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.

**6.221.2.2 posixthreads** #define posixthreads 1

**6.221.2.3 native\_thread\_platform\_type** #define native\_thread\_platform\_type posix\_←  
threads\_t

## 6.222 steerbf.cpp File Reference

### Macros

- #define **PATCH\_VAR**(var) patchbay.connect(&var.valuechanged, this, & **steerbf**::  
::**update\_cfg**)
- #define **INSERT\_PATCH**(var) **insert\_member**(var); **PATCH\_VAR**(var)

### 6.222.1 Macro Definition Documentation

**6.222.1.1 PATCH\_VAR** #define PATCH\_VAR(  
var ) patchbay.connect(&var.valuechanged, this, & **steerbf**::**update\_cfg**)

**6.222.1.2 INSERT\_PATCH** #define INSERT\_PATCH(  
var ) **insert\_member**(var); **PATCH\_VAR**(var)

## 6.223 steerbf.h File Reference

### Classes

- class **parser\_int\_dyn**
- class **steerbf\_config**
- class **steerbf**

## 6.224 testalsadvice.c File Reference

### Functions

- int **main** (int argc, char \*\*argv)

### 6.224.1 Function Documentation

```
6.224.1.1 main() int main (
    int argc,
    char ** argv )
```

## 6.225 testplugin.cpp File Reference

### Classes

- class **testplugin::config\_parser\_t**
- class **testplugin::ac\_parser\_t**
- class **testplugin::signal\_parser\_t**
- class **testplugin::if\_t**

### Namespaces

- **testplugin**

## 6.226 transducers.cpp File Reference

### Classes

- class **softclipper\_variables\_t**
- class **softclipper\_t**
- class **calibrator\_variables\_t**
- class **calibrator\_runtime\_layer\_t**
- class **calibrator\_t**
- class **bbcalib\_interface\_t**

### Typedefs

- typedef **MHAPlugin::config\_t< MHASignal::async\_rmslevel\_t > rmslevelmeter**
- typedef **MHAPlugin::plugin\_t< calibrator\_runtime\_layer\_t > rtcalibrator**

### Functions

- **speechnoise\_t::noise\_type\_t kw\_index2type** (unsigned int idx)
- std::vector< int > **vint\_0123n1** (unsigned int n)

#### 6.226.1 Typedef Documentation

**6.226.1.1 rmslevelmeter** `typedef MHAPlugin::config_t< MHASignal::async_rmslevel_t > rmslevelmeter`

**6.226.1.2 rtcalibrator** `typedef MHAPlugin::plugin_t< calibrator_runtime_layer_t > rtcalibrator`

#### 6.226.2 Function Documentation

**6.226.2.1 `kw_index2type()`** `speechnoise_t::noise_type_t kw_index2type (`  
`unsigned int idx )`

**6.226.2.2 `vint_0123n1()`** `std::vector<int> vint_0123n1 (`  
`unsigned int n )`

## 6.227 upsample.cpp File Reference

### Classes

- class `us_t`

## 6.228 wave2spec.cpp File Reference

## 6.229 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. 1492) as runtime configuration.*

### Macros

- `#define MHAPLUGIN_OVERLOAD_OUTDOMAIN`

## 6.229.1 Macro Definition Documentation

**6.229.1.1 `MHAPLUGIN_OVERLOAD_OUTDOMAIN`** `#define MHAPLUGIN_OVERLOAD_OUTDO→`  
`MAIN`

## 6.230 wavrec.cpp File Reference

### Classes

- class **wavwriter\_t**
- class **wavrec\_t**

## 6.231 windnoise.cpp File Reference

### Namespaces

- **windnoise**

*namespace for plugin windnoise which detects and cancels wind noise*

### Macros

- #define **register\_configuration\_variable(v)**

### 6.231.1 Macro Definition Documentation

## 6.231.1.1 register\_configuration\_variable #define register\_configuration\_variable( v )

## 6.232 windnoise.hh File Reference

### Classes

- class **windnoise::cfg\_t**  
*Runtime config class for windnoise plugin.*
- class **windnoise::if\_t**  
*interface class for windnoise plugin*

### Namespaces

- **windnoise**

*namespace for plugin windnoise which detects and cancels wind noise*

## 6.233 windowselector.cpp File Reference

## 6.234 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, 1476

\_MHA\_AC\_DOUBLE  
    testplugin::ac\_parser\_t, 1476

\_MHA\_AC\_FLOAT  
    testplugin::ac\_parser\_t, 1476

\_MHA\_AC\_INT  
    testplugin::ac\_parser\_t, 1476

\_MHA\_AC\_MHACOMPLEX  
    testplugin::ac\_parser\_t, 1476

\_MHA\_AC\_MHAREAL  
    testplugin::ac\_parser\_t, 1476

\_MHA\_FUN\_  
    mha\_defs.h, 1583

\_declspec  
    example5.cpp, 1546  
    example6.cpp, 1547  
    mha\_defs.h, 1583  
    mha\_plugin.hh, 1618

\_ac  
    gtfb\_simple\_rt\_t, 565

\_cf  
    DynComp::dc\_afterburn\_t, 451

\_channels  
    DynComp::dc\_afterburn\_t, 451

\_conjugate  
    Complex arithmetics in the openMHA, 67

\_fmax  
    audiometerbackend::lnn3rdoct\_t, 322

\_fmin  
    audiometerbackend::lnn3rdoct\_t, 321

\_linphase\_asym  
    MHAFilter::smoothspec\_t, 930

\_order  
    gtfb\_simple\_rt\_t, 563

\_pre\_stages  
    gtfb\_simple\_rt\_t, 564

\_prepare  
    testplugin::if\_t, 1483

\_reciprocal  
    Complex arithmetics in the openMHA, 67

\_srates  
    DynComp::dc\_afterburn\_t, 451

\_steerbf  
    steerbf\_config, 1474

\_unknown  
    testplugin::ac\_parser\_t, 1476

~Async\_Notify  
    MHA\_TCP::Async\_Notify, 795

~Connection  
    MHA\_TCP::Connection, 801

~Delay  
    ADM::Delay< F >, 267

~Event\_Watcher  
    MHA\_TCP::Event\_Watcher, 809

~Linearphase\_FIR  
    ADM::Linearphase\_FIR< F >, 269

~MHA\_Error  
    MHA\_Error, 761

~Server  
    MHA\_TCP::Server, 812

~Thread  
    MHA\_TCP::Thread, 827

~Timeout\_Watcher  
    MHA\_TCP::Timeout\_Watcher, 831

~Wakeups\_Event  
    MHA\_TCP::Wakeups\_Event, 833

~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, 229

~acSteer\_config  
    acSteer\_config, 231

~actransform\_wave  
    actransform\_wave, 234

~actransform\_wave\_config  
    actransform\_wave\_config, 238

~acmon\_t  
    acmon::acmon\_t, 208

~acspace2matrix\_t  
    MHA\_AC::acspace2matrix\_t, 723

~acwriter\_t  
    plugins::hoertech::acrec::acwriter\_t, 1380

~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, 1031  
 ~bbcalib\_interface\_t  
     bbcalib\_interface\_t, 334  
 ~blockprocessing\_polyphase\_resampling\_t  
     MHAFilter::blockprocessing\_polyphase\_resamp  
         866  
 ~c\_ifc\_parser\_t  
     MHAParser::c\_ifc\_parser\_t, 1047  
 ~cfg\_node\_t  
     MHAPlugin::cfg\_node\_t< runtime\_cfg\_t >, 1140  
 ~cfg\_t  
     acsave::cfg\_t, 223  
     equalize::cfg\_t, 459  
 ~config\_t  
     MHAPlugin::config\_t< runtime\_cfg\_t >, 1144  
 ~connector\_base\_t  
     MHAEvents::connector\_base\_t, 850  
 ~connector\_t  
     MHAEvents::connector\_t< receiver\_t >, 853  
 ~db\_if\_t  
     db\_if\_t, 370  
     dbasync\_native::db\_if\_t, 374  
 ~dbasync\_t  
     dbasync\_native::dbasync\_t, 377  
 ~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, 418  
 ~doasvm\_classification\_config  
     doasvm\_classification\_config, 421  
 ~doasvm\_feature\_extraction  
     doasvm\_feature\_extraction, 424  
     ~doasvm\_feature\_extraction\_config  
         doasvm\_feature\_extraction\_config, 427  
 ~domain\_handler\_t  
     MHAPlugin\_Split::domain\_handler\_t, 1161  
 ~double2acvar\_t  
     double2acvar::double2acvar\_t, 431  
 ~double\_t  
     MHA\_AC::double\_t, 726  
 ~doublebuffer\_t  
     MHASignal::doublebuffer\_t, 1205  
 ~dynamiclib\_t  
     dynamiclib\_t, 444  
 ~emitter\_t  
     MHAEvents::emitter\_t, 856  
 ~fft\_t  
     MHAEvents::fft\_t, 1208  
 ~fftfb\_t  
     MHAOvIFilter::fftfb\_t, 1004  
 ~fftfilter\_t  
     MHAFilter::fftfilter\_t, 874  
 ~fftfilterbank\_t  
     MHAFilter::fftfilterbank\_t, 880  
 ~filter\_t  
     MHAFilter::filter\_t, 886  
 ~float\_t  
     MHA\_AC::float\_t, 728  
 ~fourway\_processor\_t  
     PluginLoader::fourway\_processor\_t, 1362  
 ~fshift\_config\_t  
     fshift::fshift\_config\_t, 506  
 ~fshift\_t  
     fshift::fshift\_t, 509  
 ~fw\_t  
     fw\_t, 521  
 ~gaintable\_t  
     DynComp::gaintable\_t, 455  
 ~gamma\_flt\_t  
     MHAFilter::gamma\_flt\_t, 890  
 ~gsc\_adaptive\_stage  
     gsc\_adaptive\_stage::gsc\_adaptive\_stage, 535  
 ~gsc\_adaptive\_stage\_if  
     gsc\_adaptive\_stage::gsc\_adaptive\_stage\_if, 542  
 ~gtfb\_analyzer\_cfg\_t  
     gtfb\_analyzer::gtfb\_analyzer\_cfg\_t, 547  
 ~gtfb\_simd\_cfg\_t  
     gtfb\_simd\_cfg\_t, 554  
 ~hanning\_ramps\_t  
     hanning\_ramps\_t, 571

~hilbert\_fftw\_t  
    MHASignal::hilbert\_fftw\_t, 1212

~hilbert\_shifter\_t  
    fshift\_hilbert::hilbert\_shifter\_t, 516

~hilbert\_t  
    MHASignal::hilbert\_t, 1215

~hz2bark\_t  
    MHAOvIFilter::barkscale::hz2bark\_t, 1000

~int\_t  
    MHA\_AC::int\_t, 730

~io\_asterisk\_fwcb\_t  
    io\_asterisk\_fwcb\_t, 582

~io\_asterisk\_parser\_t  
    io\_asterisk\_parser\_t, 587

~io\_asterisk\_sound\_t  
    io\_asterisk\_sound\_t, 594

~io\_asterisk\_t  
    io\_asterisk\_t, 598

~io\_file\_t  
    io\_file\_t, 603

~io\_lib\_t  
    io\_lib\_t, 609

~io\_parser\_t  
    io\_parser\_t, 614

~io\_portaudio\_t  
    MHAIOPortAudio::io\_portaudio\_t, 956

~io\_tcp\_fwcb\_t  
    io\_tcp\_fwcb\_t, 618

~io\_tcp\_parser\_t  
    io\_tcp\_parser\_t, 623

~io\_tcp\_sound\_t  
    io\_tcp\_sound\_t, 630

~io\_tcp\_t  
    io\_tcp\_t, 636

~io\_wrapper  
    io\_wrapper, 639

~level\_matching\_config\_t  
    level\_matching::level\_matching\_config\_t,  
        648

~level\_matching\_t  
    level\_matching::level\_matching\_t, 651

~linear\_table\_t  
    MHATableLookup::linear\_table\_t, 1276

~lpc  
    lpc, 658

~lpc\_bl\_predictor  
    lpc\_bl\_predictor, 661

~lpc\_bl\_predictor\_config  
    lpc\_bl\_predictor\_config, 664

~lpc\_burglattice  
    lpc\_burglattice, 667

~lpc\_burglattice\_config  
    lpc\_burglattice\_config, 670

~lpc\_config  
    lpc\_config, 673

~matlab\_wrapper\_rt\_cfg\_t  
    matlab\_wrapper::matlab\_wrapper\_rt\_cfg\_t,  
        692

~matrix\_t  
    MHASignal::matrix\_t, 1226

~mha\_dblbuf\_t  
    mha\_dblbuf\_t< FIFO >, 746

~mha\_fifo\_lw\_t  
    mha\_fifo\_lw\_t< T >, 767

~mha\_fifo\_posix\_threads\_t  
    mha\_fifo\_posix\_threads\_t, 770

~mha\_fifo\_t  
    mha\_fifo\_t< T >, 775

~mha\_fifo\_thread\_guard\_t  
    mha\_fifo\_thread\_guard\_t, 780

~mha\_fifo\_thread\_platform\_t  
    mha\_fifo\_thread\_platform\_t, 781

~mha\_rt\_fifo\_element\_t  
    mha\_rt\_fifo\_element\_t< T >, 785

~mha\_rt\_fifo\_t  
    mha\_rt\_fifo\_t< T >, 787

~mha\_stash\_environment\_variable\_t  
    mha\_stash\_environment\_variable\_t, 793

~mhaplug\_cfg\_t  
    mhaplug\_cfg\_t, 1139

~mhaplugloader\_t  
    MHAParser::mhaplugloader\_t, 1088  
        PluginLoader::mhaplugloader\_t, 1367

~mhaserver\_t  
    mhaserver\_t, 1191

~osc\_server\_t  
    osc\_server\_t, 1320

~overlapadd\_if\_t  
    overlapadd::overlapadd\_if\_t, 1328  
        smoothgains\_bridge::overlapadd\_if\_t,  
            1450

~overlapadd\_t  
    overlapadd::overlapadd\_t, 1331

~parser\_t  
    MHAParser::parser\_t, 1100

~partitioned\_convolution\_t  
    MHAFilter::partitioned\_convolution\_t, 914

~patchbay\_t  
    MHAEvents::patchbay\_t< receiver\_t >,  
        858

~plug\_t  
    plug\_t, 1345

~plug\_wrapper  
     plug\_wrapper, 1347  
 ~plug\_wrapperl  
     plug\_wrapperl, 1349  
 ~plugin\_t  
     MHAPlugin::plugin\_t< runtime\_cfg\_t >, 1149  
 ~pluginlib\_t  
     pluginlib\_t, 1357  
 ~pluginloader\_t  
     pluginloader\_t, 1373  
 ~plugs\_t  
     mhachain::plugs\_t, 844  
 ~port\_t  
     MHAJack::port\_t, 982  
 ~posix\_threads\_t  
     MHAPlugin\_Split::posix\_threads\_t, 1169  
 ~proc\_counter\_t  
     proc\_counter\_t, 1385  
 ~rmslevel\_t  
     rmslevel::rmslevel\_t, 1391  
 ~rohBeam  
     rohBeam::rohBeam, 1396  
 ~rohConfig  
     rohBeam::rohConfig, 1404  
 ~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, 226  
     lsl2ac::save\_var\_t, 684  
 ~server\_t  
     mha\_tcp::server\_t, 817  
 ~sf\_t  
     MHASndFile::sf\_t, 1272  
 ~smooth\_cepstrum\_t  
     smooth\_cepstrum::smooth\_cepstrum\_t, 1440  
 ~smoothspec\_t  
     MHAFilter::smoothspec\_t, 928  
 ~spec2wave\_t  
     spec2wave\_t, 1464  
 ~spec\_fader\_t  
     spec\_fader\_t, 1466  
 ~spectrum\_t  
     MHA\_AC::spectrum\_t, 732  
     MHASignal::spectrum\_t, 1246  
     ~split\_t  
         MHAPlugin\_Split::split\_t, 1174  
     ~splitted\_part\_t  
         MHAPlugin\_Split::splitted\_part\_t, 1181  
     ~steerbf  
         steerbf, 1471  
     ~steerbf\_config  
         steerbf\_config, 1473  
     ~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, 1494  
     ~waveform\_t  
         MHA\_AC::waveform\_t, 736  
         MHASignal::waveform\_t, 1262  
     ~wavewriter\_t  
         wavewriter\_t, 1502  
     ~windowselector\_t  
         windowselector\_t, 1515  
     ~wrapped\_plugin\_t  
         matlab\_wrapper::matlab\_wrapper\_t::wrapped\_plugin\_t, 701

A

A  
     ipc\_config, 674  
     MHAFilter::filter\_t, 887  
     MHAFilter::gamma\_flt\_t, 892  
     MHAFilter::iir\_filter\_t, 898  
 a  
     MHParse::base\_t::replace\_t, 1041  
 A\_  
     MHAFilter::complex\_bandpass\_t, 871  
     MHAFilter::iir\_ord1\_real\_t, 901  
 abandonned  
     mha\_rt\_fifo\_element\_t< T >, 786  
 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, 216  
acsave::cfg\_t, 223  
acsave::save\_var\_t, 227  
acTransform\_wave\_config, 238  
adaptive\_feedback\_canceller\_config, 246  
doasvm\_classification\_config, 422  
fw\_t, 526  
gsc\_adaptive\_stage::gsc\_adaptive\_stage, 536  
latex\_doc\_t, 643  
lpc\_bl\_predictor\_config, 665  
lpc\_burglattice\_config, 670  
lsl2ac::save\_var\_t, 687  
MHA\_AC::ac2matrix\_helper\_t, 718  
MHA\_AC::double\_t, 727  
MHA\_AC::float\_t, 728  
MHA\_AC::int\_t, 730  
MHA\_AC::spectrum\_t, 733  
MHA\_AC::waveform\_t, 737  
mhachain::plugs\_t, 845  
MHAKernel::algo\_comm\_class\_t, 989  
MHAMultiSrc::base\_t, 992  
MHAPlugin::plugin\_t< runtime\_cfg\_t >, 1152  
PluginLoader::mhaplugloader\_t, 1370  
plugins::hoertech::acrec::acrec\_t, 1378  
proc\_counter\_t, 1386  
rt\_nlms\_t, 1415  
shadowfilter\_end::cfg\_t, 1426  
smooth\_cepstrum::smooth\_cepstrum\_t, 1441  
steerbf\_config, 1474  
testplugin::if\_t, 1483  
AC variable, 4  
ac2lsl, 78  
    types, 78  
ac2lsl.cpp, 1517  
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, 718  
ac2matrix\_t

MHA\_AC::ac2matrix\_t, 720  
 ac2osc.cpp, 1518  
 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, 1518  
 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, 358  
   MHAParser::mhaplugloader\_t, 1090  
 ac\_data  
   osc\_variable\_t, 1325  
 AC\_DIM\_MISMATCH  
   mha\_algo\_comm.cpp, 1580  
 ac\_double  
   double2acvar::double2acvar\_t, 432  
 ac\_fifo  
   analysepath\_t, 308  
 ac\_insert  
   osc\_server\_t, 1321  
   osc\_variable\_t, 1324  
 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, 204  
 ac\_monitor\_type.cpp, 1518  
 ac\_monitor\_type.hh, 1518  
 ac\_mul.cpp, 1519  
 ac\_mul.hh, 1519  
   ARG\_CC, 1519  
   ARG\_CR, 1519  
   ARG\_RC, 1519  
   ARG\_RR, 1519  
   arg\_type\_t, 1519  
   VAL\_COMPLEX, 1520  
   VAL\_REAL, 1520  
   val\_type\_t, 1519  
 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, 1476  
ac\_proc, 78  
ac\_proc.cpp, 1520  
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, 565  
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, 1497  
acb  
    gtfb\_simple\_rt\_t, 564  
accept  
    MHA\_TCP::Server, 813  
accept\_event  
    MHA\_TCP::Server, 814  
accept\_loop  
    io\_asterisk\_t, 599  
    io\_tcp\_t, 637  
acceptor  
    mha\_tcp::server\_t, 820  
acceptor\_started  
    mhaserver\_t, 1191  
accf  
    gtfb\_simple\_rt\_t, 564  
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, 1520  
    INSERT\_PATCH, 1520  
    PATCH\_VAR, 1520  
acConcat\_wave.h, 1521  
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, 1521  
acmon::ac\_monitor\_t, 203  
    ac\_monitor\_t, 204  
    dimstr, 205  
    getvar, 205  
    mon, 205  
    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  
    dimensions, 209  
    disemode, 209  
    patchbay, 210  
    prepare, 208  
    process, 209  
    recmode, 210  
    release, 208

save\_vars, 209  
 update\_recmode, 209  
 varlist, 209  
 vars, 210  
 acmon\_t  
     acmon::acmon\_t, 208  
 acname  
     osc\_variable\_t, 1325  
 acPooling\_wave, 211  
     ~acPooling\_wave, 212  
     acPooling\_wave, 212  
     alpha, 214  
     like\_ratio\_name, 214  
     lower\_threshold, 213  
     max\_pool\_ind\_name, 214  
     neighbourhood, 214  
     numsamples, 213  
     p\_biased\_name, 214  
     p\_name, 214  
     patchbay, 214  
     pool\_name, 214  
     pooling\_type, 213  
     pooling\_wndlen, 213  
     prepare, 212  
     prob\_bias, 214  
     process, 212  
     release, 213  
     update\_cfg, 213  
     upper\_threshold, 213  
 acPooling\_wave.cpp, 1521  
     INSERT\_PATCH, 1521  
     PATCH\_VAR, 1521  
 acPooling\_wave.h, 1522  
 acPooling\_wave\_config, 215  
     ~acPooling\_wave\_config, 216  
     ac, 216  
     acPooling\_wave\_config, 215  
     alpha, 218  
     c, 217  
     insert, 216  
     like\_ratio, 217  
     low\_thresh, 217  
     neigh, 217  
     p, 216  
     p\_biased, 216  
     p\_max, 217  
     pool, 218  
     pooling\_ind, 217  
     pooling\_option, 217  
     pooling\_size, 217  
     prob\_bias\_func, 218  
     process, 216  
     raw\_p\_name, 216  
     up\_thresh, 217  
 acrec.cpp, 1522  
 acrec.hh, 1522  
 acrec\_t  
     plugins::hoertech::acrecrec::acrecrec\_t, 1375  
 acsave, 79  
 acsave.cpp, 1522  
     ACSAVE\_FMT\_M, 1523  
     ACSAVE\_FMT\_MAT4, 1523  
     ACSAVE\_FMT\_TXT, 1523  
     ACSAVE\_SFMT\_M, 1523  
     ACSAVE\_SFMT\_MAT4, 1523  
     ACSAVE\_SFMT\_TXT, 1523  
 acsave::acsave\_t, 218  
     acsave\_t, 220  
     algo, 221  
     b\_flushed, 222  
     b\_prepared, 221  
     bflush, 221  
     event\_start\_recording, 220  
     event\_stop\_and\_flush, 220  
     fileformat, 221  
     fname, 221  
     patchbay, 222  
     prepare, 220  
     process, 220, 221  
     reclen, 221  
     release, 220  
     variables, 221  
     varlist, 221  
     varlist\_t, 219  
 acsave::cfg\_t, 222  
     ~cfg\_t, 223  
     ac, 223  
     cfg\_t, 222  
     flush\_data, 223  
     max\_frames, 224  
     nvars, 223  
     rec\_frames, 224  
     store\_frame, 223  
     varlist, 223  
 acsave::mat4head\_t, 224  
     cols, 224  
     imag, 225  
     namelen, 225  
     rows, 224  
     t, 224  
 acsave::save\_var\_t, 225  
     ~save\_var\_t, 226

ac, 227  
b\_complex, 227  
data, 227  
framecnt, 227  
maxframe, 227  
name, 227  
ndim, 227  
nframes, 227  
save\_m, 226  
save\_mat4, 226  
save\_txt, 226  
save\_var\_t, 226  
store\_frame, 226  
ACSAVE\_FMT\_M  
acsave.cpp, 1523  
ACSAVE\_FMT\_MAT4  
acsave.cpp, 1523  
ACSAVE\_FMT\_TXT  
acsave.cpp, 1523  
ACSAVE\_SFMT\_M  
acsave.cpp, 1523  
ACSAVE\_SFMT\_MAT4  
acsave.cpp, 1523  
ACSAVE\_SFMT\_TXT  
acsave.cpp, 1523  
acsave\_t  
acsave::acsave\_t, 220  
acspace  
ac2osc\_t, 184  
acspace2matrix\_t  
MHA\_AC::acspace2matrix\_t, 722  
acspace\_template  
analysispath\_if\_t, 312  
acSteer, 228  
~acSteer, 229  
acSteer, 229  
acSteerName1, 230  
acSteerName2, 230  
nrefmic, 230  
nsteerchan, 230  
patchbay, 231  
prepare, 229  
process, 229  
release, 230  
steerFile, 230  
update\_cfg, 230  
acSteer.cpp, 1524  
INSERT\_PATCH, 1524  
PATCH\_VAR, 1524  
acSteer.h, 1524  
acSteer\_config, 231  
~acSteer\_config, 231  
acSteer\_config, 231  
insert, 232  
nangle, 232  
nchan, 232  
nfreq, 232  
nrefmic, 232  
nsteerchan, 232  
specSteer1, 232  
specSteer2, 232  
acSteerName1  
acSteer, 230  
acSteerName2  
acSteer, 230  
act\_  
wavwriter\_t, 1503  
actgains  
fader\_if\_t, 485  
activate  
ac2lsl::ac2lsl\_t, 165  
lsl2ac::lsl2ac\_t, 680  
activate\_query  
MHAParser::base\_t, 1037  
active  
addsndfile::addsndfile\_if\_t, 254  
plugins::hoertech::acrec::acwriter\_t, 1382  
acTransform\_wave, 233  
~acTransform\_wave, 234  
acTransform\_wave, 234  
ang\_name, 236  
numsamples, 237  
patchbay, 237  
prepare, 235  
process, 234  
raw\_p\_max\_name, 236  
raw\_p\_name, 236  
release, 236  
rotated\_p\_max\_name, 236  
rotated\_p\_name, 236  
to\_from, 237  
update\_cfg, 236  
acTransform\_wave.cpp, 1524  
INSERT\_PATCH, 1525  
PATCH\_VAR, 1524  
acTransform\_wave.h, 1525  
acTransform\_wave\_config, 237  
~acTransform\_wave\_config, 238  
ac, 238  
acTransform\_wave\_config, 238  
ang\_name, 238  
offset, 239

process, 238  
 raw\_p\_max\_name, 238  
 raw\_p\_name, 238  
 resolution, 239  
 rotated\_i, 239  
 rotated\_p, 239  
 to\_from, 239  
 acvar  
     MHA\_AC::ac2matrix\_helper\_t, 719  
 acwriter\_t  
     plugins::hoertech::acrec::acwriter\_t, 1380  
 adapt\_filter\_param\_t  
     MHAFilter::adapt\_filter\_param\_t, 860  
 adapt\_filter\_state\_t  
     MHAFilter::adapt\_filter\_state\_t, 861  
 adapt\_filter\_t  
     MHAFilter::adapt\_filter\_t, 863  
 adaptation\_ratio  
     adm\_if\_t, 274  
     adm\_rtconfig\_t, 278  
 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, 1525  
     INSERT\_PATCH, 1525  
     make\_friendly\_number\_by\_limiting, 1526  
     PATCH\_VAR, 1525  
 adaptive\_feedback\_canceller.h, 1526  
 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, 1558  
 add\_connection  
     mha\_tcp::server\_t, 820  
 add\_entry  
     MHAParser::keyword\_list\_t, 1070  
     MHATableLookup::linear\_table\_t, 1277  
     MHATableLookup::xy\_table\_t, 1283  
 add\_fun  
     MHAOvlFilter::scale\_var\_t, 1025  
 add\_parent\_on\_insert  
     MHAParser::base\_t, 1037  
 add\_plug  
     altplugs\_t, 304  
 add\_plugin  
     pluginbrowser\_t, 1353  
 add\_plugins  
     pluginbrowser\_t, 1353

add\_replace\_pair  
    MHParse::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, 1526  
    DEBUG, 1527

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, 277

adm.cpp, 1527  
    adm\_fir\_decomb, 1528  
    adm\_fir\_lp, 1528

adm.hh, 1528

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, 1528  
 adm\_fir\_lp  
     adm.cpp, 1528  
 adm\_if\_t, 271  
     adaptation\_ratio, 274  
     adm\_if\_t, 272  
     beta, 274  
     bypass, 274  
     coeff\_decomb, 274  
     coeff\_lp, 274  
     decomb\_order, 274  
     distances, 273  
     framecnt, 275  
     front\_channels, 273  
     input\_channels, 274  
     is\_prepared, 273  
     lp\_order, 273  
     mu\_beta, 274  
     out, 273  
     patchbay, 275  
     prepare, 272  
     process, 272  
     rear\_channels, 273  
     release, 273  
     srate, 275  
     tau\_beta, 274  
     update, 273  
 adm\_rtconfig\_t, 275  
     ~adm\_rtconfig\_t, 277  
     adaptation\_ratio, 278  
     adm, 277  
     adm\_rtconfig\_t, 276  
     adm\_t, 276  
     adms, 279  
     check\_index, 277  
     decomb\_coeffs, 279  
     front\_channel, 278  
     front\_channels, 278  
     get\_adaptation\_ratio, 278  
     lp\_coeffs, 279  
     num\_adms, 277  
     rear\_channel, 278  
     rear\_channels, 278  
 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, 221  
     analysispath\_if\_t, 312  
     coherence::cohflt\_if\_t, 349  
     db\_if\_t, 371  
     dbasync\_native::db\_if\_t, 376  
     dc::dc\_if\_t, 382  
     fftfilterbank::fftfb\_interface\_t, 502  
     MHAPlugin\_Resampling::resampling\_if\_t,  
         1156  
     multibandcompressor::interface\_t, 1302  
     nlms\_t, 1308  
     overlapadd::overlapadd\_if\_t, 1330  
     route::interface\_t, 1411  
     smoothgains\_bridge::overlapadd\_if\_t,  
         1452  
     wave2spec\_if\_t, 1492  
 algo\_comm\_class\_t  
     MHAKernel::algo\_comm\_class\_t, 985  
 algo\_comm\_default  
     mha\_algo\_comm.cpp, 1580  
     mha\_algo\_comm.hh, 1582  
 ALGO\_COMM\_ID\_STR  
     mha\_algo\_comm.hh, 1582  
 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, 279  
     get\_entries, 285  
     get\_error, 285  
     get\_var, 283  
     get\_var\_double, 285  
     get\_var\_float, 284  
     get\_var\_int, 284  
     handle, 280  
     insert\_var, 281  
     insert\_var\_double, 282  
     insert\_var\_float, 281  
     insert\_var\_int, 281  
     is\_var, 283  
     mha.hh, 1577  
     remove\_ref, 282  
     remove\_var, 282

algo\_name  
    lpc, 659

algos  
    mhachain::chain\_base\_t, 840  
    mhachain::plugs\_t, 845  
    MHAParser\_Split::split\_t, 1177

all\_dump  
    MHAParser, 127

all\_ids  
    MHAParser, 127

alloc\_plugs  
    mhachain::plugs\_t, 844

almost  
    Complex arithmetics in the openMHA, 68

alp  
    gsc\_adaptive\_stage::gsc\_adaptive\_stage,  
        538  
    gsc\_adaptive\_stage::gsc\_adaptive\_stage\_ifalphaPSD  
        545

alph  
    plingploing::plingploing\_t, 1344

alpha  
    acPooling\_wave, 214  
    acPooling\_wave\_config, 218  
    cfg\_t, 346  
    coherence::cohfilt\_t, 351  
    coherence::vars\_t, 354

alpha\_blocking\_XkXi  
    rohBeam::configOptions, 1394  
    rohBeam::rohConfig, 1406

alpha\_blocking\_XkY  
    rohBeam::configOptions, 1394  
    rohBeam::rohConfig, 1406

alpha\_const  
    smooth\_cepstrum::smooth\_cepstrum\_t,  
        1442

alpha\_const\_limits\_hz  
    smooth\_cepstrum::smooth\_cepstrum\_if\_t,  
        1438  
    smooth\_cepstrum::smooth\_params, 1448

alpha\_const\_vals  
    smooth\_cepstrum::smooth\_cepstrum\_if\_t,  
        1438  
    smooth\_cepstrum::smooth\_params, 1448

alpha\_frame  
    smooth\_cepstrum::smooth\_cepstrum\_t,  
        1444

alpha\_hat  
    smooth\_cepstrum::smooth\_cepstrum\_t,  
        1443

alpha\_Lowpass

windnoise::cfg\_t, 1507

alpha\_pitch  
    smooth\_cepstrum::smooth\_cepstrum\_if\_t,  
        1437  
    smooth\_cepstrum::smooth\_params, 1448

alpha\_postfilter  
    rohBeam::configOptions, 1394  
    rohBeam::rohConfig, 1406

alpha\_prev  
    smooth\_cepstrum::smooth\_cepstrum\_t,  
        1442

alphaPH1mean  
    noise\_psd\_estimator::noise\_psd\_estimator\_if\_t,  
        1310

alphaPH1mean\_  
    noise\_psd\_estimator::noise\_psd\_estimator\_t,  
        1313

alphaPSD\_  
    noise\_psd\_estimator::noise\_psd\_estimator\_if\_t,  
        1310

alsa\_base\_t, 286  
    ~alsa\_base\_t, 287  
    alsa\_base\_t, 287  
    pcm, 288  
    read, 287  
    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, 580

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, 1529  
 altconfig.hh, 1529  
     MHAPLUGIN\_OVERLOAD\_OUTDOMAIN, amplitude  
         1529  
 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, 1529  
     MHAPLUGIN\_OVERLOAD\_OUTDOMAIN,  
         1530  
 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  
     sine\_cfg\_t, 1430  
 analysesemhaplugin.cpp, 1530  
     document\_io\_plugin, 1530  
     document\_plugin, 1530  
     main, 1531  
     print\_ac, 1530  
     strdom, 1530  
 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, 308  
     inner\_ac\_copy, 308  
     inner\_error, 308  
     inner\_input, 307  
     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, 1531  
     thread\_start, 1531  
 analysispath\_if\_t, 310  
     ~analysispath\_if\_t, 311  
     acspace\_template, 312  
     algo, 312  
     analysispath\_if\_t, 311  
     fifolen, 312  
     fragsize, 312  
     libname, 312  
     loadlib, 311  
     patchbay, 312

plug, 312  
prepare, 311  
priority, 312  
process, 311  
release, 311  
vars, 312  
analytic  
    fshift\_hilbert::hilbert\_shifter\_t, 516  
ang\_name  
    acTransform\_wave, 236  
    acTransform\_wave\_config, 238  
angle  
    Complex arithmetics in the openMHA, 62  
angle\_ind  
    steerbf, 1472  
angle\_src  
    steerbf, 1472  
angles  
    doasvm\_classification, 420  
announce\_port  
    mhasurerver\_t, 1193  
antialias  
    ds\_t, 441  
    us\_t, 1487  
apply\_gains  
    MHAoVlFilter::fftfb\_t, 1004  
    multibandcompressor::plugin\_signals\_t,  
        1303  
aquire\_mutex  
    mha\_fifo\_posix\_threads\_t, 771  
    mha\_fifo\_thread\_platform\_t, 782  
arg  
    MHA\_TCP::Thread, 827  
ARG\_CC  
    ac\_mul.hh, 1519  
ARG\_CR  
    ac\_mul.hh, 1519  
ARG\_RC  
    ac\_mul.hh, 1519  
ARG\_RR  
    ac\_mul.hh, 1519  
arg\_type\_t  
    ac\_mul.hh, 1519  
argt  
    ac\_mul\_t, 195  
array  
    matlab\_wrapper::types< MHA\_WAVEFORM  
        >, 708  
array\_type  
    matlab\_wrapper::types< MHA\_SPECTRUM  
        >, 707  
ASSERT\_EQUAL\_DIM  
    mha\_signal.cpp, 1627  
ASSERT\_EQUAL\_DIM\_PTR  
    mha\_signal.cpp, 1627  
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, 820  
ASYNC\_CONNECT\_STARTED  
    mha\_tcp.cpp, 1640  
Async\_Notify  
    MHA\_TCP::Async\_Notify, 794  
async\_poll\_msg  
    fw\_t, 524  
async\_read  
    fw\_t, 524  
async\_rmslevel\_t  
    MHASignal::async\_rmslevel\_t, 1196  
atomic\_read\_ptr  
    mha\_fifo\_if\_t< T >, 766  
atomic\_write\_ptr  
    mha\_fifo\_if\_t< T >, 766  
attack  
    cfg\_t, 346  
    dc::dc\_t, 385  
    dc\_simple::level\_smoothen\_t, 405  
    softclip\_t, 1456  
    softclipper\_t, 1457  
attenuate20.cpp, 1531  
attenuate20\_t, 313  
    attenuate20\_t, 313  
    prepare, 314  
    process, 314  
    release, 314  
attr  
    analysepath\_t, 309  
    dbasync\_native::dbasync\_t, 378  
    MHAPlugin\_Split::posix\_threads\_t, 1171  
audiometer\_if\_t  
    audiometerbackend::audiometer\_if\_t, 316  
    audiometerbackend, 82  
    gcd, 83  
    generator, 83  
    level\_adaptor, 83  
    return\_sig, 83

audiometerbackend.cpp, 1532  
     DEBUG, 1532  
 audiometerbackend::audiometer\_if\_t, 315  
     audiometer\_if\_t, 316  
     change\_mode, 316  
     freq, 317  
     level, 317  
     mode, 317  
     outchannel, 317  
     patchbay, 317  
     pmode, 317  
     prepare, 316  
     process, 316  
     ramplen, 317  
     set\_level, 316  
     sigtype, 317  
     update, 316  
 audiometerbackend::level\_adapt\_t, 318  
     can\_update, 319  
     get\_level, 319  
     ilen, 319  
     l\_new, 320  
     l\_old, 320  
     level\_adapt\_t, 319  
     pos, 319  
     update\_frame, 319  
     wnd, 320  
 audiometerbackend::Inn3rdoct\_t, 320  
     \_fmax, 322  
     \_fmin, 321  
     bandpass, 321  
     dev, 322  
     iterate\_Inn, 321  
     Inn3rdoct\_t, 321  
     random, 322  
     rng, 322  
 audiometerbackend::signal\_gen\_t, 322  
     signal\_gen\_t, 323  
 audiometerbackend::sine\_t, 323  
     sine\_t, 324  
 auditory\_profile.cpp, 1533  
 auditory\_profile.h, 1533  
 AuditoryProfile, 84  
 AuditoryProfile::fmap\_t, 324  
     get\_frequencies, 325  
     get\_values, 325  
     isempty, 325  
 AuditoryProfile::parser\_t, 325  
     get\_current\_profile, 326  
     L, 327  
     parser\_t, 326  
     R, 327  
 AuditoryProfile::parser\_t::ear\_t, 327  
     ear\_t, 328  
     get\_ear, 328  
     HTL, 328  
     UCL, 328  
 AuditoryProfile::parser\_t::fmap\_t, 329  
     f, 330  
     fmap\_t, 329  
     get\_fmap, 330  
     name\_, 330  
     patchbay, 330  
     validate, 330  
     value, 330  
 AuditoryProfile::profile\_t, 331  
     get\_ear, 331  
     L, 332  
     R, 332  
 AuditoryProfile::profile\_t::ear\_t, 332  
     convert\_empty2normal, 333  
     HTL, 333  
     UCL, 333  
 available\_streams  
     Isl2ac::Isl2ac\_t, 682  
 average  
     coherence::vars\_t, 354  
 avg\_ipd  
     coherence::cohflt\_t, 351  
 azimuth  
     mha\_direction\_t, 751  
 B  
     MHAFilter::filter\_t, 888  
     MHAFilter::iir\_filter\_t, 898  
 b  
     doasvm\_classification, 420  
     MHAParser::base\_t::replace\_t, 1041  
 B\_  
     MHAFilter::complex\_bandpass\_t, 872  
     MHAFilter::iir\_ord1\_real\_t, 901  
 b\_check\_version  
     PluginLoader::mhaplugloader\_t, 1372  
 b\_complex  
     acsave::save\_var\_t, 227  
 b\_cont  
     acmon::acmon\_t, 210  
 b\_est  
     lpc\_bl\_predictor\_config, 665  
 b\_exit\_request  
     fw\_t, 527  
 b\_flushed  
     acsave::acsave\_t, 222

b\_fw\_started  
    io\_parser\_t, 616

b\_interactive  
    mhaserver\_t, 1193

b\_is\_input  
    calibrator\_runtime\_layer\_t, 337  
    calibrator\_t, 340

b\_is\_prepared  
    PluginLoader::mhaplugloader\_t, 1372

b\_loop  
    MHASignal::loop\_wavefragment\_t, 1221

b\_ltg  
    coherence::cohfilt\_t, 352

b\_permute  
    ac\_proc::interface\_t, 198

b\_prepared  
    acsave::acsave\_t, 221  
    io\_file\_t, 606  
    io\_parser\_t, 617  
    mhachain::chain\_base\_t, 841  
    mhachain::plugs\_t, 845  
    MHAJack::client\_t, 979

b\_process  
    io\_alsa\_t, 578

b\_ready  
    MHAJack::client\_avg\_t, 967

b\_record  
    ac2osc\_t, 184

b\_snapshot  
    acmon::acmon\_t, 210

b\_starting  
    io\_parser\_t, 617

b\_stopped  
    io\_parser\_t, 617  
    MHAJack::client\_avg\_t, 966  
    MHAJack::client\_noncont\_t, 969

b\_use\_clipping  
    calibrator\_runtime\_layer\_t, 337

b\_use\_fir  
    calibrator\_runtime\_layer\_t, 337

b\_use\_profiling  
    mhachain::plugs\_t, 847

backward  
    lpc\_bl\_predictor\_config, 665  
    lpc\_burglattice\_config, 671  
    MHASignal::fft\_t, 1209

backward\_scale  
    MHASignal::fft\_t, 1210

band\_weights  
    dc::dc\_vars\_t, 390

bandpass

audiometerbackend::Inn3rdoct\_t, 321

bands  
    gtfb\_analyzer::gtfb\_analyzer\_cfg\_t, 547  
    gtfb\_simd\_cfg\_t, 555  
    MHAOvlFilter::fspacing\_t, 1017

bandsXchannels  
    gtfb\_simd\_cfg\_t, 555

bandw\_correction  
    speechnoise.cpp, 1702

bark2hz\_t  
    MHAOvlFilter::barkscale::bark2hz\_t, 999

BARKSCALE\_ENTRIES  
    mha\_fftfb.cpp, 1591

bartlett  
    MHAWindow, 154

bartlett\_t  
    MHAWindow::bartlett\_t, 1287

base\_t  
    MHAMultiSrc::base\_t, 992  
    MHAParser::base\_t, 1030, 1031  
    MHAWindow::base\_t, 1288

basename  
    save\_spec\_t, 1419  
    save\_wave\_t, 1421  
    shadowfilter\_begin::shadowfilter\_begin\_t,  
        1424  
    shadowfilter\_end::shadowfilter\_end\_t,  
        1428

bass  
    plingploing::plingploing\_t, 1342

bassmod  
    plingploing::if\_t, 1339

bassmod\_

bassperiod  
    plingploing::if\_t, 1340

bassperiod\_

bbcilib\_interface\_t, 333  
    ~bbcilib\_interface\_t, 334

bbcilib\_interface\_t, 334  
    calib\_in, 335  
    calib\_out, 335  
    plugloader, 335  
    prepare, 334  
    process, 334  
    release, 335

beam1  
    rohBeam::rohConfig, 1406

beamA  
    rohBeam::rohConfig, 1406

beamExport  
     rohBeam::rohBeam, 1401

beamW  
     rohBeam::rohConfig, 1406

beta  
     ADM::ADM< F >, 263  
     adm\_if\_t, 274

beta\_const  
     smooth\_cepstrum::smooth\_cepstrum\_if\_t, 1437  
     smooth\_cepstrum::smooth\_params, 1448

bf\_src  
     steerbf, 1472

bf\_src\_copy  
     steerbf\_config, 1474

bf\_vec  
     steerbf\_config, 1474

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, 1394  
     rohBeam::rohConfig, 1405

blinvert  
     coherence::cohflt\_t, 352

blackman  
     MHAWindow, 155

blackman\_t  
     MHAWindow::blackman\_t, 1290

blockprocessing\_polyphase\_resampling\_t  
     MHAFilter::blockprocessing\_polyphase\_resampling\_t, 865

blocks  
     droptect\_t, 438

blockSpec  
     rohBeam::rohConfig, 1406

blockXp  
     rohBeam::rohConfig, 1407

bookkeeping  
     MHAFilter::partitioned\_convolution\_t, 916  
     MHAParser::mhapluginloader\_t, 1090

bool\_mon\_t  
     MHAParser::bool\_mon\_t, 1042

bool\_t  
     MHAParser::bool\_t, 1044

bpm  
     plingploing::if\_t, 1339

bprofiling  
     mhachain::chain\_base\_t, 840

bracket\_balance  
     MHAParser::StrCnv, 130

brown  
     speechnoise\_t, 1468

browsemhaplugins.cpp, 1533  
     DEBUG, 1533  
     main, 1534

bt  
     plingploing::plingploing\_t, 1342

buf  
     ac2lsl::save\_var\_t< mha\_complex\_t >, 178  
     ac2lsl::save\_var\_t< T >, 175  
     lsl2ac::save\_var\_t, 686  
     mha\_fifo\_t< T >, 778  
     mha\_spec\_t, 791  
     mha\_wave\_t, 837

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, 806

buffered\_outgoing\_bytes  
     MHA\_TCP::Connection, 806

buffersize  
     lsl2ac::lsl2ac\_t, 681

bufSize  
     gsc\_adaptive\_stage::gsc\_adaptive\_stage, 536

bufsize  
     lsl2ac::save\_var\_t, 689

burn

DynComp::dc\_afterburn\_rt\_t, 447  
DynComp::dc\_afterburn\_t, 450  
multibandcompressor::interface\_t, 1302  
butter\_stop\_ord1  
    MHAFilter, 107  
bw  
    MHAOvIFilter::fscale\_bw\_t, 1012  
bw\_  
    MHAFilter::gamma\_flt\_t, 892  
bw\_generator  
    MHAFilter::thirdoctave\_analyzer\_t, 932  
bw\_hz  
    MHAOvIFilter::fscale\_bw\_t, 1012  
bw\_name  
    dc::dc\_vars\_t, 389  
bwv  
    MHAOvIFilter::fftfb\_ac\_info\_t, 1002  
    multibandcompressor::fftfb\_plug\_t, 1299  
bypass  
    adm\_if\_t, 274  
    db\_if\_t, 371  
    dc::dc\_t, 386  
    dc::dc\_vars\_t, 389  
    dc\_simple::dc\_vars\_t, 402  
    DynComp::dc\_afterburn\_vars\_t, 454

C

ADM, 82

c

    acPooling\_wave\_config, 217  
    adaptive\_feedback\_canceller, 242  
    doasvm\_classification\_config, 422  
    io\_tcp\_sound\_t::float\_union, 634  
    mha\_complex\_test\_array\_t, 743  
    nlms\_t, 1307

c1\_a  
    MHAFilter::o1\_ar\_filter\_t, 905

c1\_r  
    MHAFilter::o1\_ar\_filter\_t, 905

c2\_a  
    MHAFilter::o1\_ar\_filter\_t, 905

c2\_r  
    MHAFilter::o1\_ar\_filter\_t, 905

c\_ifc\_parser\_t  
    MHAParser::c\_ifc\_parser\_t, 1047

c\_min  
    coherence::cohflt\_t, 351

c\_parse\_cmd  
    MHAParser::c\_ifc\_parser\_t, 1048

c\_parse\_cmd\_t  
    MHAParser, 126

c\_parse\_err  
    MHAParser::c\_ifc\_parser\_t, 1048  
c\_parse\_err\_t  
    MHAParser, 128

c\_scale  
    coherence::cohflt\_t, 351

calc\_in  
    wave2spec\_t, 1497

calc\_out  
    overlapadd::overlapadd\_t, 1333  
    spec2wave\_t, 1465

calc\_pre\_wnd  
    wave2spec\_t, 1496

calc\_sine  
    cpupload::cpupload\_cfg\_t, 365

calib\_in  
    bbcalib\_interface\_t, 335

calib\_out  
    bbcalib\_interface\_t, 335

calibrator\_runtime\_layer\_t, 335  
    b\_is\_input, 337  
    b\_use\_clipping, 337  
    b\_use\_fir, 337  
    calibrator\_runtime\_layer\_t, 336  
    fir, 337  
    firfir2ffflen, 336  
    firflen, 336  
    gain, 337  
    pmode, 338  
    process, 336  
    quant, 337  
    softclip, 337  
    spechnoise, 337

calibrator\_t, 338  
    b\_is\_input, 340  
    calibrator\_t, 339  
    patchbay, 340  
    prepare, 339  
    prepared, 340  
    process, 339  
    read\_levels, 340  
    release, 339  
    update, 340  
    update\_tau\_level, 340  
    vars, 340

calibrator\_variables\_t, 341  
    calibrator\_variables\_t, 341  
    config\_parser, 343  
    do\_clipping, 343  
    fir, 341  
    fragsize, 342  
    nbits, 342

num\_channels, 343  
 peaklevel, 341  
 rmslevel, 342  
 softclip, 343  
 spnoise\_channels, 342  
 spnoise\_level, 342  
 spnoise\_mode, 342  
 spnoise\_parser, 342  
 srate, 342  
 tau\_level, 342  
 callback  
     matlab\_wrapper::callback, 690  
     matlab\_wrapper::matlab\_wrapper\_t, 698  
 callbacks  
     matlab\_wrapper::matlab\_wrapper\_t, 698  
 can\_read  
     MHAFilter::blockprocessing\_polyphase\_resampling\_t,  
         867  
 can\_read\_bytes  
     MHA\_TCP::Connection, 804  
 can\_read\_line  
     MHA\_TCP::Connection, 803  
 can\_sysread  
     MHA\_TCP::Connection, 802  
 can\_syswrite  
     MHA\_TCP::Connection, 802  
 can\_update  
     addsndfile::level\_adapt\_t, 256  
     audiometerbackend::level\_adapt\_t, 319  
     fader\_wave::level\_adapt\_t, 489  
 catch\_condition  
     MHAParser::posix\_threads\_t, 1171  
 catch\_thread  
     MHAParser::dummy\_threads\_t,  
         1167  
     MHAParser::posix\_threads\_t, 1169  
     MHAParser::thread\_platform\_t,  
         1186  
 categories  
     plugindescription\_t, 1355  
 cb\_patchbay  
     matlab\_wrapper::matlab\_wrapper\_t, 698  
 cdata  
     mha\_audio\_t, 740  
     MHASignal::matrix\_t, 1232  
 center\_frequencies  
     dc::dc\_vars\_t, 390  
     dc\_simple::dc\_if\_t, 395  
 cf  
     mha\_audio\_descriptor\_t, 738  
     MHAFilter::thirdoctave\_analyzer\_t, 932  
     MHAOvlFilter::band\_descriptor\_t, 998  
     MHAOvlFilter::ffftb\_vars\_t, 1009  
     plingploing::plingploing\_t, 1342  
 cf2bands  
     MHAOvlFilter::fspacing\_t, 1016  
 cf\_  
     MHAFilter::gamma\_flt\_t, 892  
     wavwriter\_t, 1503  
 cf\_generator  
     MHAFilter::thirdoctave\_analyzer\_t, 932  
 cf\_h  
     MHAOvlFilter::band\_descriptor\_t, 998  
 cf\_in  
     overlapadd::overlapadd\_if\_t, 1330  
     smoothgains\_bridge::overlapadd\_if\_t,  
         1452  
     MHAPlugin::mhaplugloader\_t, 1090  
 cf\_input  
     PluginLoader::mhaplugloader\_t, 1371  
 cf\_l  
     MHAOvlFilter::band\_descriptor\_t, 997  
 cf\_name  
     dc::dc\_vars\_t, 389  
 cf\_out  
     overlapadd::overlapadd\_if\_t, 1330  
     smoothgains\_bridge::overlapadd\_if\_t,  
         1452  
 cf\_out\_  
     MHAParser::mhaplugloader\_t, 1090  
 cf\_output  
     PluginLoader::mhaplugloader\_t, 1372  
 cfac  
     route::interface\_t, 1411  
 cfg  
     MHAParser::config\_t< runtime\_cfg\_t >,  
         1146  
 cfg\_  
     MHAFilter::thirdoctave\_analyzer\_t, 932  
 cfg\_dump  
     MHAParser, 126  
 cfg\_dump\_short  
     MHAParser, 126  
 cfg\_node\_current  
     MHAParser::config\_t< runtime\_cfg\_t >,  
         1147  
 cfg\_node\_t  
     MHAParser::cfg\_node\_t< runtime\_cfg\_t  
         >, 1140  
 cfg\_root  
     MHAParser::config\_t< runtime\_cfg\_t >,

1147  
cfg\_t, 343  
    ac2lsl::cfg\_t, 167  
    acsave::cfg\_t, 222  
    alpha, 346  
    attack, 346  
    cfg\_t, 344  
    channel, 345  
    decay, 346  
    equalize::cfg\_t, 459  
    frozen\_noise\_, 346  
    gain\_spec\_, 345  
    gain\_wave\_, 345  
    lsl2ac::cfg\_t, 676  
    matrixmixer::cfg\_t, 709  
    pos, 346  
    process, 345  
    rand\_dist, 346  
    replace\_, 345  
    rng, 346  
    scale, 345  
    shadowfilter\_begin::cfg\_t, 1422  
    shadowfilter\_end::cfg\_t, 1425  
    start\_lin, 346  
    use\_frozen\_, 345  
    windnoise::cfg\_t, 1505  
cfin  
    altplugs\_t, 305  
    fw\_t, 526  
    mhachain::chain\_base\_t, 841  
    route::interface\_t, 1411  
cfout  
    altplugs\_t, 305  
    fw\_t, 526  
    mhachain::chain\_base\_t, 841  
    route::interface\_t, 1411  
cfv  
    MHAOvlFilter::fftfb\_ac\_info\_t, 1001  
    multibandcompressor::fftfb\_plug\_t, 1299  
cg  
    coherence::cohflt\_t, 351  
ch  
    MHASignal::doublebuffer\_t, 1207  
chain\_base\_t  
    mhachain::chain\_base\_t, 839  
chains  
    MHAPlugin\_Split::split\_t, 1178  
chance  
    dropgen\_t, 435  
change\_mode  
    addsndfile::addsndfile\_if\_t, 252  
    audiometerbackend::audiometer\_if\_t, 316  
channel  
    cfg\_t, 345  
    example5\_t, 478  
    MHAMultiSrc::channel\_t, 993  
channel\_gain\_name  
    combc\_if\_t, 357  
channel\_gains\_  
    combc\_t, 359  
channel\_info  
    mha\_spec\_t, 791  
    mha\_wave\_t, 837  
channel\_no  
    example6\_t, 480  
channel\_pair  
    level\_matching::channel\_pair, 644, 645  
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, 496  
    gtfb\_analyzer::gtfb\_analyzer\_cfg\_t, 547  
    gtfb\_simd\_cfg\_t, 555  
    level\_matching::level\_matching\_t, 653  
    mhaconfig\_t, 848  
    MHAFilter::fftfilter\_t, 877  
    MHAFilter::filter\_t, 888  
    MHAParser::mhaconfig\_mon\_t, 1085  
    MHAPlugin\_Split::split\_t, 1177  
    MHASignal::delay\_t, 1201  
    rt\_nlms\_t, 1415  
    sine\_cfg\_t, 1430  
    sine\_t, 1433  
    testplugin::config\_parser\_t, 1479  
    Vector and matrix processing toolbox, 38  
channels\_t  
    MHAMultiSrc::channels\_t, 993  
char\_data  
    testplugin::ac\_parser\_t, 1477  
chdir  
    mha\_audio\_descriptor\_t, 739  
check\_alignment  
    gtfb\_simd.cpp, 1558  
CHECK\_EXPR  
    mha\_defs.h, 1583  
check\_index  
    adm\_rtconfig\_t, 277

**check\_low**  
 MHParse::range\_var\_t, 1108  
**check\_parenthesis\_complex**  
 mha\_parser.cpp, 1611  
**check\_range**  
 MHParse::range\_var\_t, 1108  
**check\_sign\_complex**  
 mha\_parser.cpp, 1611  
**check\_sound\_data\_type**  
 io\_tcp\_sound\_t, 631  
**check\_up**  
 MHParse::range\_var\_t, 1108  
**CHECK\_VAR**  
 mha\_defs.h, 1583  
**check\_vars**  
 ac2lsl::cfg\_t, 167  
**chname**  
 dc::dc\_vars\_t, 389  
**chunkbytes\_in**  
 io\_asterisk\_sound\_t, 594  
 io\_tcp\_sound\_t, 631  
**chunksize**  
 lsl2ac::lsl2ac\_t, 681  
 lsl2ac::save\_var\_t, 688  
**ci**  
 matrixmixer::matmix\_t, 712  
**class\_signal\_type**  
 matlab\_wrapper::types< MHA\_SPECTRUMcmd\_release  
     >, 708  
 matlab\_wrapper::types< MHA\_WAVEFORMcmd\_start  
     >, 708  
**cleanup\_plugins**  
 mhachain::plugins\_t, 845  
**cleanup\_unused\_cfg**  
 MHAPlugIn::config\_t< runtime\_cfg\_t >, 1146  
**clear**  
 mha\_fifo\_t< T >, 777  
 MHATableLookup::linear\_table\_t, 1278  
 MHATableLookup::table\_t, 1280  
 MHATableLookup::xy\_table\_t, 1284  
 Vector and matrix processing toolbox, 44  
**clear\_chains**  
 MHAPlugIn\_Split::split\_t, 1175  
**clear\_plugins**  
 pluginbrowser\_t, 1353  
**Client**  
 MHA\_TCP::Client, 798  
**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, 390  
 dc\_simple::dc\_if\_t, 395  
**clientname**  
 MHAIOJack::io\_jack\_t, 941  
 MHAIOJackdb::io\_jack\_t, 949  
**clipmeter**  
 softclipper\_t, 1457  
**clipped**  
 softclipper\_variables\_t, 1460  
**close\_session**  
 plugins::hoertech::acrec::acwriter\_t, 1382  
 wavwriter\_t, 1503  
**closed**  
 MHA\_TCP::Connection, 807  
**closesocket**  
 mha\_tcp.cpp, 1640  
**cLTASS**  
 gtfb\_simple\_rt\_t, 565  
 gtfb\_simple\_t, 570  
 MHAOvlFilter::ffftfb\_ac\_info\_t, 1002  
 MHAOvlFilter::ffftfb\_vars\_t, 1010  
**cmd\_prepare**  
 MHAIOPortAudio::io\_portaudio\_t, 957  
 MHAIOPortAudio::io\_portaudio\_t, 958  
**cmd\_start**  
 MHAIOPortAudio::io\_portaudio\_t, 957  
**cmd\_stop**  
 MHAIOPortAudio::io\_portaudio\_t, 957  
**co**  
 matrixmixer::matmix\_t, 712  
**coeff**  
 gtfb\_analyzer::gtfb\_analyzer\_cfg\_t, 548  
 gtfb\_analyzer::gtfb\_analyzer\_t, 552  
 gtfb\_simd\_t, 560  
**coeff\_decomb**  
 adm\_if\_t, 274  
**coeff\_lp**  
 adm\_if\_t, 274  
**coh\_c**  
 coherence::cohflt\_t, 352  
**coh\_rlp**  
 coherence::cohflt\_t, 352  
**coherence**, 84  
 getcipd, 84  
**coherence.cpp**, 1534  
**coherence::cohflt\_if\_t**, 347

algo, 349  
cohfilt\_if\_t, 348  
patchbay, 348  
prepare, 348  
process, 348  
release, 348  
update, 348  
vars, 349  
coherence::cohfilt\_t, 349  
alpha, 351  
avg\_ipd, 351  
b\_ltg, 352  
blInvert, 352  
c\_min, 351  
c\_scale, 351  
cg, 351  
coh\_c, 352  
coh\_rlp, 352  
cohfilt\_t, 350  
g, 351  
gain, 352  
gain\_delay, 352  
insert, 350  
limit, 351  
lp1i, 352  
lp1ltg, 352  
lp1r, 351  
nbands, 351  
process, 350  
s\_out, 352  
staticgain, 353  
coherence::vars\_t, 353  
alpha, 354  
average, 354  
delay, 355  
invert, 354  
limit, 354  
ltgcomp, 355  
ltgtau, 355  
mapping, 354  
staticgain, 355  
tau, 354  
tau\_unit, 354  
vars\_t, 354  
cohfilt\_if\_t  
    coherence::cohfilt\_if\_t, 348  
cohfilt\_t  
    coherence::cohfilt\_t, 350  
collect\_result  
    MHAPlugin\_Split::split\_t, 1176  
    MHAPlugin\_Split::splitted\_part\_t, 1183  
colored\_intensity  
    Vector and matrix processing toolbox, 54  
cols  
    acsave::mat4head\_t, 224  
combc\_if\_t, 355  
    channel\_gain\_name, 357  
    combc\_if\_t, 356  
    element\_gain\_name, 357  
    interleaved, 357  
    outchannels, 357  
    prepare, 356  
    process, 356  
    combc\_t, 357  
        ac\_, 358  
        channel\_gains\_, 359  
        combc\_t, 358  
        element\_gain\_name\_, 359  
        interleaved\_, 359  
        nbands, 359  
        process, 358  
        s\_out, 359  
        w\_out, 359  
combinechannels.cpp, 1534  
comm\_var\_t, 359  
    data, 361  
    data\_type, 360  
    num\_entries, 360  
    stride, 361  
commentate  
    MHAParser, 126  
commit  
    DynComp::dc\_afterburn\_vars\_t, 454  
commit\_pending  
    DynComp::dc\_afterburn\_t, 451  
commit\_t  
    MHAParser::commit\_t< receiver\_t >, 1050  
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, 659  
    lpc\_config, 674  
comp\_iter  
    lpc\_config, 674  
compensation  
    windnoise::cfg\_t, 1507  
COMPILER\_ID

compiler\_id.hh, 1536  
 compiler\_id.cpp, 1535  
 compiler\_id.hh, 1535  
   COMPILER\_ID, 1536  
   COMPILER\_ID\_MAJOR, 1535  
   COMPILER\_ID\_MINOR, 1535  
   COMPILER\_ID\_PATCH, 1535  
   COMPILER\_ID\_VENDOR, 1535  
   COMPILER\_ID\_VERSION, 1536  
   COMPILER\_ID\_VERSION\_HELPER1, 1536  
   COMPILER\_ID\_VERSION\_HELPER2, 1535  
 COMPILER\_ID\_MAJOR  
   compiler\_id.hh, 1535  
 COMPILER\_ID\_MINOR  
   compiler\_id.hh, 1535  
 COMPILER\_ID\_PATCH  
   compiler\_id.hh, 1535  
 COMPILER\_ID\_VENDOR  
   compiler\_id.hh, 1535  
 COMPILER\_ID\_VERSION  
   compiler\_id.hh, 1536  
 COMPILER\_ID\_VERSION\_HELPER1  
   compiler\_id.hh, 1536  
 COMPILER\_ID\_VERSION\_HELPER2  
   compiler\_id.hh, 1535  
 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, 869  
 complex\_data  
   testplugin::ac\_parser\_t, 1477  
 complex\_filter.cpp, 1536  
 complex\_filter.h, 1536  
 complex\_mon\_t  
   MHAParser::complex\_mon\_t, 1052  
 complex\_ofs  
   MHASignal::matrix\_t, 1232  
 complex\_scale\_channel.cpp, 1537  
 complex\_scale\_channel\_t, 361  
   complex\_scale\_channel\_t, 362  
   factor, 363  
   patchbay, 363  
   prepare, 362  
   process, 362  
   scale\_ch, 363  
   update\_cfg, 363  
 complex\_t  
   MHAParser::complex\_t, 1054  
 compression  
   dc\_simple::dc\_t, 398  
 compute\_beamW  
   rohBeam::rohBeam, 1398  
 compute\_delaycomp\_vec  
   rohBeam::rohBeam, 1398  
 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, 1398  
 compute\_wng  
   rohBeam::rohBeam, 1399  
 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, 1359  
 config\_in  
   testplugin::if\_t, 1483

config\_out  
    testplugin::if\_t, 1483

config\_parser  
    calibrator\_variables\_t, 343

config\_parser\_t  
    testplugin::config\_parser\_t, 1479

config\_t  
    MHAParser::config\_t< runtime\_cfg\_t >, 1144

configfile  
    PluginLoader::config\_file\_splitter\_t, 1360

configname  
    PluginLoader::config\_file\_splitter\_t, 1360

configs  
    altconfig\_t, 299

configuration, 4

configuration variable, 4

conflux  
    DynComp::dc\_afterburn\_rt\_t, 448  
    DynComp::dc\_afterburn\_vars\_t, 453

conjugate  
    Complex arithmetics in the openMHA, 67  
    Vector and matrix processing toolbox, 57

connect  
    MHAEvents::emitter\_t, 856  
    MHAEvents::patchbay\_t< receiver\_t >, 858, 859

connect\_input  
    MHAJack::client\_t, 974

connect\_output  
    MHAJack::client\_t, 974

connect\_to  
    MHAJack::port\_t, 983

connected  
    io\_asterisk\_parser\_t, 591  
    io\_tcp\_parser\_t, 628

Connection  
    MHA\_TCP::Connection, 801

connection\_loop  
    io\_asterisk\_t, 599  
    io\_tcp\_t, 637

connections  
    mha\_tcp::server\_t, 820  
    MHAEvents::emitter\_t, 857

connections\_in  
    MHAIOJack::io\_jack\_t, 941  
    MHAIOJackdb::io\_jack\_t, 949

connections\_out  
    MHAIOJack::io\_jack\_t, 942  
    MHAIOJackdb::io\_jack\_t, 949

connector  
    MHAFilter::adapt\_filter\_t, 864  
    MHAFilter::iir\_filter\_t, 898  
    MHAParser::mhaplugindoc.cpp, 1090

connector\_base\_t  
    MHAEvents::connector\_base\_t, 850

connector\_t  
    MHAEvents::connector\_t< receiver\_t >, 853

cons  
    MHAEvents::patchbay\_t< receiver\_t >, 859

consecutive\_dropouts  
    droptect\_t, 438

CONST\_C  
    rohBeam, 160

contained\_frames  
    MHASignal::ringbuffer\_t, 1238

conv2latex  
    generatemhaplugindoc.cpp, 1552

convert\_empty2normal  
    AuditoryProfile::profile\_t::ear\_t, 333

convert\_f2logf  
    gaintable.cpp, 1550

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, 1655  
    MHAIOTCP.cpp, 1681

copy\_from\_at  
    MHASignal::waveform\_t, 1268

copy\_output\_spec  
    MHAParser::split\_t, 1175

copy\_output\_wave  
    MHAParser::split\_t, 1175

copy\_permuted  
    MHASignal, 149

copyfixedfboutput  
    rohBeam::rohConfig, 1405

corr\_out  
    lpc\_config, 675

corrLL  
    rohBeam::rohConfig, 1407

corrRR  
    rohBeam::rohConfig, 1407

corrXpXp

rohBeam::rohConfig, 1407  
 corrXpYf  
 rohBeam::rohConfig, 1407  
 corrZZ  
 rohBeam::rohConfig, 1407  
 cpupload, 85  
 cpupload.cpp, 1537  
 cpupload::cpupload\_cfg\_t, 363  
     calc\_sine, 365  
     cpupload\_cfg\_t, 364  
     factor, 365  
     phase, 365  
     process, 364  
     result, 365  
     table, 365  
     use\_sine, 365  
     write\_to\_table, 365  
 cpupload::cpupload\_if\_t, 366  
     cpupload\_if\_t, 367  
     factor, 368  
     patchbay, 368  
     prepare, 367  
     process, 367  
     table\_size, 368  
     update, 367  
     use\_sine, 368  
 cpupload\_cfg\_t  
     cpupload::cpupload\_cfg\_t, 364  
 cpupload\_if\_t  
     cpupload::cpupload\_if\_t, 367  
 create\_datafile  
     plugins::hoertech::acrec::acwriter\_t, 1381  
 create\_latex\_doc  
     generatemhaplugindoc.cpp, 1553  
 create\_or\_replace\_var  
     ac2lsl::cfg\_t, 167  
 create\_soundfile  
     wavwriter\_t, 1502  
 creator  
     speechnoise\_t, 1469  
 creator\_A  
     MHAFilter::complex\_bandpass\_t, 869  
 creator\_B  
     MHAFilter::complex\_bandpass\_t, 870  
 cstr\_strerror  
     mha\_errno.c, 1585  
 current  
     altplugs\_t, 304  
     mha\_rt\_fifo\_t< T >, 789  
 current\_input\_signal\_buffer\_half\_index  
     MHAFilter::partitioned\_convolution\_t, 916  
 current\_message  
     mha\_tcp::buffered\_socket\_t, 797  
 current\_output\_partition\_index  
     MHAFilter::partitioned\_convolution\_t, 917  
 current\_powspec  
     droptect\_t, 439  
 current\_thread\_priority  
     MHAPlugin\_Split::posix\_threads\_t, 1170  
 current\_thread\_scheduler  
     MHAPlugin\_Split::posix\_threads\_t, 1170  
 cv  
     lsl2ac::save\_var\_t, 687  
     plugins::hoertech::acrec::acrec\_t, 1377  
 cvalue  
     gtfb\_analyzer::gtfb\_analyzer\_cfg\_t, 548  
 d  
     gsc\_adaptive\_stage::gsc\_adaptive\_stage, 539  
 data  
     acsave::save\_var\_t, 227  
     comm\_var\_t, 361  
     DynComp::gaintable\_t, 458  
     MHA\_AC::acspace2matrix\_t, 725  
     MHA\_AC::double\_t, 726  
     MHA\_AC::float\_t, 728  
     MHA\_AC::int\_t, 730  
     mha\_rt\_fifo\_element\_t< T >, 786  
     MHAParser::bool\_mon\_t, 1043  
     MHAParser::bool\_t, 1045  
     MHAParser::complex\_mon\_t, 1052  
     MHAParser::complex\_t, 1055  
     MHAParser::float\_mon\_t, 1059  
     MHAParser::float\_t, 1062  
     MHAParser::int\_mon\_t, 1064  
     MHAParser::int\_t, 1067  
     MHAParser::kw\_t, 1074  
     MHAParser::mcomplex\_mon\_t, 1076  
     MHAParser::mcomplex\_t, 1079  
     MHAParser::mfloat\_mon\_t, 1081  
     MHAParser::mfloat\_t, 1083  
     MHAParser::mint\_mon\_t, 1093  
     MHAParser::mint\_t, 1095  
     MHAParser::string\_mon\_t, 1111  
     MHAParser::string\_t, 1113  
     MHAParser::vcomplex\_mon\_t, 1118  
     MHAParser::vcomplex\_t, 1120  
     MHAParser::vfloat\_mon\_t, 1122  
     MHAParser::vfloat\_t, 1125  
     MHAParser::vint\_mon\_t, 1127  
     MHAParser::vint\_t, 1130  
     MHAParser::vstring\_mon\_t, 1132

MHAParser::vstring\_t, 1134  
MHAParser::cfg\_node\_t< runtime\_cfg\_t >, 1141  
MHASignal::uint\_vector\_t, 1258  
wavwriter\_t, 1504  
data\_is\_initialized  
    MHAParser::base\_t, 1039  
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, 360  
    testplugin::ac\_parser\_t, 1477  
data\_type\_  
    ac2lsl::save\_var\_t< T >, 175  
data\_type\_t  
    testplugin::ac\_parser\_t, 1476  
db.cpp, 1537  
db2lin  
    MHASignal, 140  
db2sq  
    MHASignal, 141  
db\_if\_t, 369  
    ~db\_if\_t, 370  
    algo, 371  
    bypass, 371  
    db\_if\_t, 370  
    dbasync\_native::db\_if\_t, 374  
    fragsize, 371  
    patchbay, 370  
    plugloader, 371  
    prepare, 370  
    process, 370  
    release, 370  
db\_t, 371  
    db\_t, 372  
    inner\_process, 372  
    plugloader, 372  
dbasync.cpp, 1537  
dbasync\_native, 85  
    gcd, 86  
    INVALID\_THREAD\_PRIORITY, 86  
    thread\_start, 86  
dbasync\_native::db\_if\_t, 373  
    ~db\_if\_t, 374  
    algo, 376  
    db\_if\_t, 374  
    delay, 375  
    fragsize, 375  
    framework\_thread\_priority, 376  
framework\_thread\_scheduler, 376  
plugloader, 375  
prepare, 374  
process, 374  
release, 375  
sub\_ac, 375  
worker\_thread\_priority, 375  
worker\_thread\_scheduler, 375  
dbasync\_native::dbasync\_t, 376  
    ~dbasync\_t, 377  
    attr, 378  
    dbasync\_t, 377  
    inner\_input, 378  
    outer\_output, 378  
    outer\_process, 377  
    plugloader, 378  
    priority, 378  
    scheduler, 378  
    svc, 378  
    thread, 378  
dbasync\_native::delay\_check\_t, 379  
    delay\_check\_t, 379  
dbasync\_t  
    dbasync\_native::dbasync\_t, 377  
DBG  
    MHAIoalsa.cpp, 1647  
dbspl2pa  
    MHASignal, 142  
dbspl2pa2  
    MHASignal, 143  
DC  
    dc\_simple, 87  
dc, 86  
dc.cpp, 1538  
    DUPVEC, 1538  
    get\_audiochannels, 1538  
dc.hh, 1539  
dc::dc\_if\_t, 380  
    algo, 382  
    dc\_if\_t, 381  
    patchbay, 382  
    prepare, 381  
    process, 381  
    update, 381  
    update\_monitors, 381  
dc::dc\_t, 382  
    attack, 385  
    bypass, 386  
    dc\_t, 383  
    decay, 386  
    explicit\_insert, 384

fftlen, 386  
 get\_attack\_filter\_state, 385  
 get\_decay\_filter\_state, 385  
 get\_level\_in\_db, 384  
 get\_level\_in\_db\_adjusted, 385  
 get\_nbands, 384  
 get\_nch, 384  
 get\_rmslevel\_filter\_state, 385  
 gt, 385  
 level\_in\_db, 386  
 level\_in\_db\_adjusted, 386  
 log\_interp, 386  
 naudiochannels, 386  
 nbands, 386  
 nch, 386  
 offset, 385  
 process, 384  
 rmslevel, 385  
 dc::dc\_vars\_t, 387  
     band\_weights, 390  
     bw\_name, 389  
     bypass, 389  
     center\_frequencies, 390  
     cf\_name, 389  
     chname, 389  
     clientid, 390  
     dc\_vars\_t, 388  
     edge\_frequencies, 390  
     ef\_name, 389  
     filterbank, 389  
     filtered\_level, 390  
     gainrule, 390  
     gtdata, 388  
     gtmin, 388  
     gtstep, 388  
     input\_level, 390  
     log\_interp, 389  
     modified, 390  
     offset, 389  
     preset, 390  
     tauattack, 388  
     taudecay, 389  
     taurmslevel, 388  
 dc::dc\_vars\_validator\_t, 391  
     dc\_vars\_validator\_t, 391  
 dc\_afterburn.cpp, 1539  
     mylogf, 1539  
 dc\_afterburn.h, 1539  
 dc\_afterburn\_rt\_t  
     DynComp::dc\_afterburn\_rt\_t, 447  
 dc\_afterburn\_t  
     DynComp::dc\_afterburn\_t, 450  
 dc\_afterburn\_vars\_t  
     DynComp::dc\_afterburn\_vars\_t, 453  
 dc\_if\_t  
     dc::dc\_if\_t, 381  
     dc\_simple::dc\_if\_t, 393  
 dc\_simple, 86  
     DC, 87  
     force\_resize, 88  
     LEVEL, 87  
     not\_zero, 88  
     test\_fail, 87  
 dc\_simple.cpp, 1540  
 dc\_simple.hh, 1540  
 dc\_simple::dc\_if\_t, 392  
     center\_frequencies, 395  
     clientid, 395  
     dc\_if\_t, 393  
     edge\_frequencies, 395  
     filterbank, 395  
     gainrule, 395  
     has Been Modified, 394  
     modified, 395  
     mon\_g, 395  
     mon\_l, 395  
     patchbay, 396  
     prepare, 393  
     prepared, 396  
     preset, 395  
     process, 394  
     read\_modified, 394  
     release, 393  
     update\_dc, 394  
     update\_gain\_mon, 394  
     update\_level, 394  
     update\_level\_mon, 394  
 dc\_simple::dc\_t, 396  
     compression, 398  
     dc\_t, 397  
     expansion, 398  
     expansion\_threshold, 398  
     limiter, 398  
     limiter\_threshold, 398  
     maxgain, 398  
     mon\_g, 399  
     mon\_l, 399  
     nbands, 398  
     process, 397, 398  
 dc\_simple::dc\_t::line\_t, 399  
     line\_t, 399  
     m, 400

operator(), 400  
y0, 400  
dc\_simple::dc\_vars\_t, 400  
bypass, 402  
dc\_vars\_t, 401  
expansion\_slope, 402  
expansion\_threshold, 402  
g50, 401  
g80, 401  
limiter\_threshold, 402  
maxgain, 401  
tauattack, 402  
taudecay, 402  
dc\_simple::dc\_vars\_validator\_t, 403  
dc\_vars\_validator\_t, 403  
dc\_simple::level\_smoothen\_t, 404  
attack, 405  
decay, 406  
ffflen, 406  
level\_smoothen\_t, 405  
level\_spec, 406  
level\_wave, 406  
nbands, 406  
process, 405  
dc\_t  
dc::dc\_t, 383  
dc\_simple::dc\_t, 397  
dc\_vars\_t  
dc::dc\_vars\_t, 388  
dc\_simple::dc\_vars\_t, 401  
dc\_vars\_validator\_t  
dc::dc\_vars\_validator\_t, 391  
dc\_simple::dc\_vars\_validator\_t, 403  
deallocate\_domains  
MHAPlugin\_Split::domain\_handler\_t, 1162  
DEBUG  
addsndfile.cpp, 1527  
audiometerbackend.cpp, 1532  
browseshapugins.cpp, 1533  
fader\_wave.cpp, 1548  
MHAIOFile.cpp, 1657  
debug  
io\_asterisk\_parser\_t, 591  
io\_tcp\_parser\_t, 627  
debug\_file  
io\_asterisk\_parser\_t, 592  
io\_tcp\_parser\_t, 628  
debug\_filename  
io\_asterisk\_parser\_t, 592  
io\_tcp\_parser\_t, 628  
decay  
cfg\_t, 346  
dc::dc\_t, 386  
dc\_simple::level\_smoothen\_t, 406  
softclip\_t, 1456  
softclipper\_t, 1457  
decomb\_coeffs  
adm\_rtconfig\_t, 279  
decomb\_order  
adm\_if\_t, 274  
decrease\_condition  
mha\_fifo\_posix\_threads\_t, 772  
decrement  
mha\_fifo\_posix\_threads\_t, 771  
mha\_fifo\_thread\_platform\_t, 783  
DEFAULT\_RET\_SIZE  
mha\_parser.hh, 1617  
defaultHighInputLatency  
MHAIOPortAudio::device\_info\_t, 954  
defaultHighOutputLatency  
MHAIOPortAudio::device\_info\_t, 954  
defaultLowInputLatency  
MHAIOPortAudio::device\_info\_t, 954  
defaultLowOutputLatency  
MHAIOPortAudio::device\_info\_t, 954  
defaultSampleRate  
MHAIOPortAudio::device\_info\_t, 954  
Delay  
ADM::Delay< F >, 266  
delay, 89  
coherence::vars\_t, 355  
dbasync\_native::db\_if\_t, 375  
delaysum::delaysum\_wave\_if\_t, 411  
mha\_dbbuf\_t< FIFO >, 749  
MHAFilter::gamma\_flt\_t, 892  
MHAFilter::partitioned\_convolution\_t::index\_t, 919  
MHAPlugin\_Split::split\_t, 1178  
MHASignal::delay\_spec\_t, 1199  
MHASignal::delay\_wave\_t, 1203  
delay.cpp, 1541  
delay.hh, 1541  
delay::interface\_t, 407  
delays, 408  
interface\_t, 408  
patchbay, 408  
prepare, 408  
process, 408  
update, 408  
delay\_ac  
ac2wave\_if\_t, 188

ac2wave\_t, 190  
 delay\_check\_t  
     dbasync\_native::delay\_check\_t, 379  
 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, 1406  
 delays  
     delay::interface\_t, 408  
     MHASignal::delay\_t, 1201  
 delays\_in  
     MHAIOJack::io\_jack\_t, 941  
 delays\_out  
     MHAIOJack::io\_jack\_t, 942  
 delaysum, 89  
 delaysum::delaysum\_wave\_if\_t, 409  
     delay, 411  
     delaysum\_wave\_if\_t, 410  
     patchbay, 411  
     prepare, 410  
     process, 410  
     release, 411  
     update\_cfg, 411  
     weights, 411  
 delaysum::delaysum\_wave\_t, 412  
     delaysum\_wave\_t, 413  
     out, 413  
     process, 413  
     weights, 413  
 delaysum\_spec, 89  
 delaysum\_spec.cpp, 1541  
 delaysum\_spec::delaysum\_spec\_if\_t, 414  
     delaysum\_spec\_if\_t, 415  
     gain, 416  
     groupdelay, 415  
     patchbay, 416  
     prepare, 415  
     process, 415  
     update\_cfg, 415  
     output, 417  
     process, 417  
     scale, 417  
 delaysum\_spec\_if\_t  
     delaysum\_spec::delaysum\_spec\_if\_t, 415  
 delaysum\_t  
     delaysum\_spec::delaysum\_t, 416  
 delaysum\_wave.cpp, 1542  
 delaysum\_wave\_if\_t  
     delaysum::delaysum\_wave\_if\_t, 410  
 delaysum\_wave\_t  
     delaysum::delaysum\_wave\_t, 413  
 delete\_plug  
     altplices\_t, 304  
 DELT  
     gsc\_adaptive\_stage, 96  
 delta\_phi  
     fshift::fshift\_config\_t, 507  
     fshift\_hilbert::hilbert\_shifter\_t, 518  
 delta\_phi\_total  
     fshift::fshift\_config\_t, 507  
     fshift\_hilbert::hilbert\_shifter\_t, 518  
 delta\_pitch  
     smooth\_cepstrum::smooth\_cepstrum\_if\_t,  
         1437  
     smooth\_cepstrum::smooth\_params, 1447  
 descriptor  
     mha\_audio\_t, 739  
 desired\_chan  
     gsc\_adaptive\_stage::gsc\_adaptive\_stage,  
         537  
 desired\_delay  
     gtfb\_simple\_t, 569  
 desired\_fill\_count  
     mha\_drifter\_fifo\_t< T >, 758  
 detected  
     windnoise::if\_t, 1512  
 detected\_acname  
     windnoise::if\_t, 1512  
 dev  
     audiometerbackend::Inn3rdoct\_t, 322  
 dev\_in  
     io\_alsa\_t, 579  
 dev\_out  
     io\_alsa\_t, 579  
 device  
     alsa\_dev\_par\_parser\_t, 290  
 device\_index\_in

MHAIOPortAudio::io\_portaudio\_t, 960  
device\_index\_in\_updated  
    MHAIOPortAudio::io\_portaudio\_t, 957  
device\_index\_out  
    MHAIOPortAudio::io\_portaudio\_t, 960  
device\_index\_out\_updated  
    MHAIOPortAudio::io\_portaudio\_t, 957  
device\_info  
    MHAIOPortAudio::io\_portaudio\_t, 958  
device\_info\_t  
    MHAIOPortAudio::device\_info\_t, 952  
device\_name\_in  
    MHAIOPortAudio::io\_portaudio\_t, 960  
device\_name\_in\_updated  
    MHAIOPortAudio::io\_portaudio\_t, 957  
device\_name\_out  
    MHAIOPortAudio::io\_portaudio\_t, 960  
device\_name\_out\_updated  
    MHAIOPortAudio::io\_portaudio\_t, 957  
df  
    fshift::fshift\_config\_t, 507  
    fshift::fshift\_t, 510  
    fshift\_hilbert::frequency\_translator\_t, 513  
    fshift\_hilbert::hilbert\_shifter\_t, 517  
diag\_loading\_mu  
    rohBeam::rohBeam, 1400  
diff\_coeffs  
    mha\_filter.cpp, 1594  
diff\_t  
    MHAFilter::diff\_t, 873  
digits  
    mha\_error\_helpers, 99  
dimension  
    MHASignal::matrix\_t, 1226  
dimensions  
    acmon::acmon\_t, 209  
dimstr  
    acmon::ac\_monitor\_t, 205  
dir  
    mha\_channel\_info\_t, 741  
dir\_t  
    MHAJack::port\_t, 981  
dir\_type  
    MHAJack::port\_t, 983  
dis  
    dropgen\_t, 435  
Discard  
    Isl2ac, 97  
discard  
    MHASignal::ringbuffer\_t, 1239  
disconnect  
    MHAEvents::emitter\_t, 856  
disk\_write\_threshold\_min\_num\_samples  
    plugins::hoertech::acrec::acwriter\_t, 1382  
diskbuffer  
    plugins::hoertech::acrec::acwriter\_t, 1383  
dispmode  
    acmon::acmon\_t, 209  
dist  
    plingploing::plingploing\_t, 1343  
dist1  
    plingploing::plingploing\_t, 1343  
distance  
    mha\_direction\_t, 752  
distances  
    adm\_if\_t, 273  
dm  
    lpc\_burglattice\_config, 671  
do\_clipping  
    calibrator\_variables\_t, 343  
do\_get\_var  
    testplugin::ac\_parser\_t, 1476  
do\_insert\_var  
    testplugin::ac\_parser\_t, 1476  
DO\_RESAMPLE  
    addsndfile, 80  
doagcc  
    doasvm\_feature\_extraction\_config, 427  
doasvm  
    doasvm\_classification\_config, 422  
doasvm\_classification, 417  
    ~doasvm\_classification, 418  
    angles, 420  
    b, 420  
    doasvm\_classification, 418  
    max\_p\_ind\_name, 420  
    p\_name, 420  
    patchbay, 420  
    prepare, 419  
    process, 419  
    release, 419  
    update\_cfg, 419  
    vGCC\_name, 420  
    w, 420  
    x, 420  
    y, 420  
doasvm\_classification.cpp, 1542  
    INSERT\_PATCH, 1542  
    PATCH\_VAR, 1542  
doasvm\_classification.h, 1543  
doasvm\_classification\_config, 421  
    ~doasvm\_classification\_config, 421

ac, 422  
 c, 422  
 doasvm, 422  
 doasvm\_classification\_config, 421  
 p, 422  
 p\_max, 422  
 process, 422  
 doasvm\_feature\_extraction, 423  
   ~doasvm\_feature\_extraction, 424  
   doasvm\_feature\_extraction, 424  
 fftlen, 425  
 max\_lag, 425  
 nupsample, 425  
 patchbay, 425  
 prepare, 424  
 process, 424  
 release, 425  
 update\_cfg, 425  
 vGCC\_name, 425  
 doasvm\_feature\_extraction.cpp, 1543  
   INSERT\_PATCH, 1543  
   PATCH\_VAR, 1543  
 doasvm\_feature\_extraction.h, 1543  
 doasvm\_feature\_extraction\_config, 426  
   ~doasvm\_feature\_extraction\_config, 427  
 doagcc, 427  
 doasvm\_feature\_extraction\_config, 426  
 fft, 428  
 fftlen, 427  
 G, 429  
 G\_length, 427  
 GCC\_end, 428  
 GCC\_start, 427  
 hifftwin, 428  
 hifftwin\_sum, 428  
 hwin, 428  
 ifft, 428  
 in\_spec, 429  
 proc\_wave, 428  
 process, 427  
 vGCC, 428  
 vGCC\_ac, 428  
 wndlen, 427  
 doc\_appendix.h, 1544  
 doc\_examples.h, 1544  
 doc\_frameworks.h, 1544  
 doc\_general.h, 1544  
 doc\_kernel.h, 1544  
 doc\_matlab.h, 1544  
 doc\_mhamain.h, 1544  
 doc\_parser.h, 1544  
 doc\_plugins.h, 1544  
 doc\_system.h, 1544  
 doc\_toolbox.h, 1544  
 doCircularComp  
   gsc\_adaptive\_stage::gsc\_adaptive\_stage,  
     537  
   gsc\_adaptive\_stage::gsc\_adaptive\_stage\_if,  
     544  
 document\_io\_plugin  
   analysemhaplugin.cpp, 1530  
 document\_plugin  
   analysemhaplugin.cpp, 1530  
 documentation  
   plugindescription\_t, 1355  
 domain  
   mhaconfig\_t, 848  
   MHAParser::mhaconfig\_mon\_t, 1085  
   MHAParser\_Split::splitted\_part\_t, 1183  
   rmslevel::rmslevel\_t, 1393  
   testplugin::config\_parser\_t, 1480  
 domain\_handler\_t  
   MHAParser\_Split::domain\_handler\_t,  
     1160  
 DONT\_RESAMPLE\_PERMISSIVE  
   addsndfile, 80  
 DONT\_RESAMPLE\_STRICT  
   addsndfile, 80  
 double2acvar, 90  
 double2acvar.cpp, 1544  
 double2acvar::double2acvar\_t, 429  
   ~double2acvar\_t, 431  
   ac\_double, 432  
   double2acvar\_t, 430  
   is\_prepared, 432  
   on\_configuration\_update, 432  
   patchbay, 432  
   poll\_latest\_value\_and\_reinsert, 431  
   prepare\_, 431  
   process, 431  
   release\_, 432  
 double2acvar\_t  
   double2acvar::double2acvar\_t, 430  
 double\_t  
   MHA\_AC::double\_t, 726  
 doublebuffer\_t  
   MHASignal::doublebuffer\_t, 1204  
 down  
   MHASignal::schroeder\_t, 1242  
 downsample.cpp, 1544  
 downsampling\_factor  
   MHAFilter::polyphase\_resampling\_t, 923

downscale  
    MHASignal::quantizer\_t, 1236

drain  
    DynComp::dc\_afterburn\_vars\_t, 453

drain\_inv  
    DynComp::dc\_afterburn\_rt\_t, 448

drand  
    plingloing, 156

dropgen.cpp, 1545

dropgen\_t, 433  
    chance, 435  
    dis, 435  
    dropgen\_t, 434  
    max\_sleep\_time, 435  
    min\_sleep\_time, 435  
    patchbay, 435  
    prepare, 434  
    process, 434  
    r, 435  
    random\_engine, 435  
    release, 434

dropouts  
    droptect\_t, 438

droptect.cpp, 1545

droptect\_t, 436  
    blocks, 438  
    consecutive\_dropouts, 438  
    current\_powspec, 439  
    dropouts, 438  
    droptect\_t, 437  
    filter\_activated, 439  
    filtered\_powspec, 439  
    filtered\_powspec\_mon, 439  
    level\_mon, 439  
    period, 439  
    prepare, 437  
    process, 438  
    release, 438  
    reset, 438  
    tau, 439  
    threshold, 439

ds\_t, 440  
    antialias, 441  
    ds\_t, 441  
    prepare, 441  
    process, 441  
    ratio, 441  
    release, 441

dt  
    mha\_audio\_descriptor\_t, 738

dtme

MHA\_TCP, 103

dummy\_interface\_test  
    MHAIOalsa.cpp, 1649

MHAIOAsterisk.cpp, 1654

MHAIOFile.cpp, 1659

MHAIOJack.cpp, 1663

MHAIOJackdb.cpp, 1667

MHAIOParser.cpp, 1671

MHAIOPortAudio.cpp, 1675

MHAIOTCP.cpp, 1681

dummy\_jack\_proc\_cb  
    mhajack.cpp, 1684

dummy\_threads\_t  
    MHAPlugin\_Split::dummy\_threads\_t, 1166

dump\_mha  
    fw\_t, 526

dup  
    MHAFilter::thirdoctave\_analyzer\_t, 932

duplicate\_vector  
    gtfb\_simple\_rt\_t, 563

DUPVEC  
    dc.cpp, 1538

dupvec  
    Vector and matrix processing toolbox, 41

dupvec\_chk  
    Vector and matrix processing toolbox, 41

dur\_  
    plingloing::plingloing\_t, 1342

dynamiclib\_t, 442  
    ~dynamiclib\_t, 444  
    dynamiclib\_t, 443, 444  
    fullname, 446  
    getmodulename, 445  
    getname, 445  
    h, 446  
    load\_lib, 445  
    modulename, 446  
    resolve, 444  
    resolve\_checked, 444

DynComp, 90  
    interp1, 90  
    interp2, 91

DynComp::dc\_afterburn\_rt\_t, 446  
    burn, 447  
    conflux, 448  
    dc\_afterburn\_rt\_t, 447  
    drain\_inv, 448  
    lp, 448  
    maxgain, 448  
    mpo\_inv, 448

DynComp::dc\_afterburn\_t, 449  
  \_cf, 451  
  \_channels, 451  
  \_srate, 451  
  burn, 450  
  commit\_pending, 451  
  dc\_afterburn\_t, 450  
  fb\_pars\_configured, 451  
  patchbay, 451  
  set\_fb\_pars, 450  
  unset\_fb\_pars, 450  
  update, 450  
  update\_burner, 450

DynComp::dc\_afterburn\_vars\_t, 452  
  bypass, 454  
  commit, 454  
  conflux, 453  
  dc\_afterburn\_vars\_t, 453  
  drain, 453  
  f, 453  
  maxgain, 453  
  mpo, 453  
  taugain, 453

DynComp::gaintable\_t, 454  
  ~gaintable\_t, 455  
  data, 458  
  gaintable\_t, 455  
  get\_gain, 456, 457  
  get\_iofun, 457  
  get\_vF, 457  
  get\_vL, 457  
  nbands, 457  
  nchannels, 457  
  num\_channels, 458  
  num\_F, 458  
  num\_L, 458  
  update, 456  
  vF, 458  
  vFlog, 458  
  vL, 458

E  
  gsc\_adaptive\_stage::gsc\_adaptive\_stage, 539

e  
  gsc\_adaptive\_stage::gsc\_adaptive\_stage, 539

E2  
  gsc\_adaptive\_stage::gsc\_adaptive\_stage, 539

e\_out

gsc\_adaptive\_stage::gsc\_adaptive\_stage, 540  
  ear\_t  
    AuditoryProfile::parser\_t::ear\_t, 328  
  edge\_frequencies  
    dc::dc\_vars\_t, 390  
    dc\_simple::dc\_if\_t, 395  
  ef  
    MHAOvlFilter::fftfb\_vars\_t, 1010  
  ef2bands  
    MHAOvlFilter::fspacing\_t, 1016  
  ef\_h  
    MHAOvlFilter::band\_descriptor\_t, 998  
  ef\_l  
    MHAOvlFilter::band\_descriptor\_t, 998  
  ef\_name  
    dc::dc\_vars\_t, 389  
  efv  
    MHAOvlFilter::fftfb\_ac\_info\_t, 1002  
    multibandcompressor::fftfb\_plug\_t, 1299  
  element\_gain\_name  
    combc\_if\_t, 357  
    gtfb\_simple\_t, 569  
  element\_gain\_name\_  
    combc\_t, 359  
    gtfb\_simple\_rt\_t, 565  
  elevation  
    mha\_direction\_t, 752  
  emit\_event  
    MHAEvents::connector\_base\_t, 850, 851  
    MHAEvents::connector\_t< receiver\_t >, 853, 854  
  emitter  
    MHAEvents::connector\_t< receiver\_t >, 854  
  emitter\_die  
    MHAEvents::connector\_base\_t, 851  
  emitter\_is\_alive  
    MHAEvents::connector\_base\_t, 851  
  empty\_string  
    MHAParser::keyword\_list\_t, 1071  
  enable\_adaptive\_beam  
    rohBeam::configOptions, 1394  
    rohBeam::rohBeam, 1400  
    rohBeam::rohConfig, 1405  
  enable\_export  
    rohBeam::rohBeam, 1401  
  enable\_wng\_optimization  
    rohBeam::rohBeam, 1401  
  end\_time  
    MHA\_TCP::Timeout\_Event, 830

entries  
  MHAParser::keyword\_list\_t, 1071  
  MHAParser::parser\_t, 1104

entry  
  MHAParser::entry\_t, 1056

entry\_map\_t  
  MHAParser, 126

entry\_t  
  MHAParser::entry\_t, 1055

envelope\_delay  
  MHAFilter::gamma\_flt\_t, 892

envreplace  
  MHAParser, 127

eof  
  MHA\_TCP::Connection, 803

EPrew  
  adaptive\_feedback\_canceller\_config, 249

EPSILON  
  lpc\_bl\_predictor.h, 1567  
  lpc\_burg-lattice.h, 1568

epsilon  
  smoothgains\_bridge::overlapadd\_if\_t, 1451

equal\_dim  
  Vector and matrix processing toolbox, 42

equalize, 91

equalize.cpp, 1545

equalize::cfg\_t, 459  
  ~cfg\_t, 459  
  cfg\_t, 459  
  fftgains, 460  
  nchannels, 460  
  num\_bins, 460  
  operator=, 459

equalize::freqgains\_t, 460  
  fftgains, 462  
  freqgains\_t, 461  
  id, 462  
  patchbay, 462  
  prepare, 461  
  process, 461  
  update\_gains, 462  
  update\_id, 462

equidist2bands  
  MHA毓Filter::fspacing\_t, 1016

erb\_hz\_f\_hz  
  speechnoise.cpp, 1701

ERR\_IHANDLE  
  MHAIOalsa.cpp, 1648  
  MHAIOAsterisk.cpp, 1652  
  MHAIOFile.cpp, 1658

MHAIOJack.cpp, 1662  
MHAIOJackdb.cpp, 1666  
MHAIOParser.cpp, 1670  
MHAIOPortAudio.cpp, 1674  
MHAIOTCP.cpp, 1679

err\_in  
  MHAFilter::adapt\_filter\_param\_t, 860  
  MHAFilter::adapt\_filter\_t, 864

ERR\_SUCCESS  
  MHAIOalsa.cpp, 1648  
  MHAIOAsterisk.cpp, 1652  
  MHAIOFile.cpp, 1657  
  MHAIOJack.cpp, 1662  
  MHAIOJackdb.cpp, 1666  
  MHAIOParser.cpp, 1670  
  MHAIOPortAudio.cpp, 1674  
  MHAIOTCP.cpp, 1679

ERR\_USER  
  MHAIOalsa.cpp, 1648  
  MHAIOAsterisk.cpp, 1652  
  MHAIOFile.cpp, 1658  
  MHAIOJack.cpp, 1662  
  MHAIOJackdb.cpp, 1666  
  MHAIOParser.cpp, 1670  
  MHAIOPortAudio.cpp, 1674  
  MHAIOTCP.cpp, 1679

error  
  mha\_fifo\_lw\_t< T >, 769  
  MHA\_TCP::Thread, 828

Error handling in the openMHA, 28  
  MHA\_assert, 29  
  MHA\_assert\_equal, 29  
  mha\_debug, 29  
  MHA\_ErrorMsg, 28

error\_h  
  osc\_server\_t, 1321

errorlog  
  fw\_t, 525

ESTIM\_CUR  
  nlms\_wave.cpp, 1691

ESTIM\_PREV  
  nlms\_wave.cpp, 1691

estimateDebug  
  noise\_psd\_estimator::noise\_psd\_estimator\_t, 1313

ESTIMATION\_TYPES  
  nlms\_wave.cpp, 1691

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, 220

event\_stop\_and\_flush  
    acsave::acsave\_t, 220

eventhandler  
    MHAEvents::connector\_t< receiver\_t >, 854

eventhandler\_s  
    MHAEvents::connector\_t< receiver\_t >, 854

eventhandler\_suu  
    MHAEvents::connector\_t< receiver\_t >, 855

Events  
    MHA\_TCP::Event\_Watcher, 808

events  
    MHA\_TCP::Event\_Watcher, 809

example1.cpp, 1545

example1\_t, 463  
    example1\_t, 464  
    prepare, 464  
    process, 464  
    release, 464

example2.cpp, 1545

example2\_t, 465  
    example2\_t, 466  
    factor, 468  
    prepare, 467  
    process, 468  
    release, 468  
    scale\_ch, 468

example3.cpp, 1546

example3\_t, 469  
    example3\_t, 470  
    factor, 472  
    on\_prereadaccess, 471  
    on\_scale\_ch\_readaccess, 471  
    on\_scale\_ch\_valuechanged, 471  
    on\_scale\_ch\_writeaccess, 471  
    patchbay, 473  
    prepare, 471  
    prepared, 472  
    process, 472

    release, 471  
    scale\_ch, 472  
    example4\_t, 473  
    example4\_t, 474  
    factor, 476  
    on\_prereadaccess, 475  
    on\_scale\_ch\_readaccess, 475  
    on\_scale\_ch\_valuechanged, 475  
    on\_scale\_ch\_writeaccess, 475  
    patchbay, 477  
    prepare, 475  
    prepared, 477  
    process, 476  
    release, 476  
    scale\_ch, 476  
    example5.cpp, 1546  
    \_\_declspec, 1546

example5\_t, 477  
    channel, 478  
    example5\_t, 477  
    process, 478  
    scale, 478  
    example6.cpp, 1546  
    \_\_declspec, 1547

example6\_t, 478  
    channel\_no, 480  
    example6\_t, 479  
    patchbay, 480  
    prepare, 479  
    process, 479  
    rmsdb, 480  
    update\_cfg, 480

example7.cpp, 1547

example7.hh, 1547

example7\_t, 481  
    example7\_t, 481  
    prepare, 482  
    process, 482  
    release, 482

exec\_fw\_command  
    fw\_t, 523

existed\_before  
    mha\_stash\_environment\_variable\_t, 793

exit\_on\_stop  
    fw\_t, 525

exit\_request  
    fw\_t, 522  
    plugins::hoertech::acrec::acwriter\_t, 1381  
    wavwriter\_t, 1502

expansion

dc\_simple::dc\_t, 398  
expansion\_slope  
  dc\_simple::dc\_vars\_t, 402  
expansion\_threshold  
  dc\_simple::dc\_t, 398  
  dc\_simple::dc\_vars\_t, 402  
expfilt  
  MHAOvlFilter::ShapeFun, 122  
expi  
  Complex arithmetics in the openMHA, 61,  
    64  
explicit\_insert  
  dc::dc\_t, 384  
export\_beam\_design  
  rohBeam::rohBeam, 1399  
export\_to  
  MHASignal::spectrum\_t, 1248  
  MHASignal::waveform\_t, 1269  
expression\_t, 482  
  MHAParser::expression\_t, 1056, 1057  
extern\_connector  
  MHAParser::commit\_t< receiver\_t >,  
    1050

F

  adaptive\_feedback\_canceller\_config, 246  
  rt\_nlms\_t, 1415

f

  AuditoryProfile::parser\_t::fmap\_t, 330  
  DynComp::dc\_afterburn\_vars\_t, 453  
  io\_tcp\_sound\_t::float\_union, 634  
  MHAOvlFilter::ffftb\_vars\_t, 1009  
  MHAOvlFilter::fscale\_t, 1014

f0\_high

  smooth\_cepstrum::smooth\_cepstrum\_if\_t,  
    1437  
  smooth\_cepstrum::smooth\_params, 1447

f0\_low

  smooth\_cepstrum::smooth\_cepstrum\_if\_t,  
    1437  
  smooth\_cepstrum::smooth\_params, 1447

f\_est

  lpc\_bl\_predictor\_config, 665

f\_hz

  MHAOvlFilter::fscale\_t, 1014

F\_Uflt

  adaptive\_feedback\_canceller\_config, 248

factor

  complex\_scale\_channel\_t, 363  
  cpupload::cpupload\_cfg\_t, 365  
  cpupload::cpupload\_if\_t, 368  
  example2\_t, 468

  example3\_t, 472  
  example4\_t, 476  
  plugin\_interface\_t, 1352

fader\_if\_t, 483

  actgains, 485  
  fader\_if\_t, 484  
  newgains, 485  
  patchbay, 484  
  prepare, 484  
  process, 484  
  tau, 484  
  update\_cfg, 484

fader\_spec.cpp, 1547

fader\_wave, 92

  level\_adaptor, 92

fader\_wave.cpp, 1547

  DEBUG, 1548

fader\_wave::fader\_wave\_if\_t, 485

  fader\_wave\_if\_t, 486

  gain, 487  
  patchbay, 487  
  prepare, 486  
  prepared, 487  
  process, 486  
  ramplen, 487  
  release, 486  
  set\_level, 487

fader\_wave::level\_adapt\_t, 488

  can\_update, 489  
  get\_level, 489  
  ilen, 489  
  l\_new, 489  
  l\_old, 490  
  level\_adapt\_t, 488  
  pos, 489  
  update\_frame, 489  
  wnd, 489

fader\_wave\_if\_t

  fader\_wave::fader\_wave\_if\_t, 486

fail\_on\_async\_jackerr

  MHAIOJackdb::io\_jack\_t, 949

fail\_on\_async\_jackerror

  MHAIOJackdb::io\_jack\_t, 946  
  MHAJack::client\_t, 980

fail\_on\_nonmonotonic

  MHAOvlFilter::ffftb\_vars\_t, 1009

fail\_on\_nonmonotonic\_cf

  MHAOvlFilter::fspacing\_t, 1016

fail\_on\_unique\_bins

  MHAOvlFilter::ffftb\_vars\_t, 1009

fail\_on\_unique\_fftbins

MHAOvlFilter::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, 526  
 fb  
     MHAFilter::thirdoctave\_analyzer\_t, 932  
 fb\_acinfo  
     fftfilterbank::fftfb\_plug\_t, 505  
 fb\_pars\_configured  
     DynComp::dc\_afterburn\_t, 451  
 fbpow  
     fftfbpow::fftfbpow\_t, 494  
 fcn\_init  
     matlab\_wrapper::matlab\_wrapper\_t::wrapped\_pffitfilt\_t, 491  
         name, 492  
         patchbay, 493  
 fcn\_prepare  
     matlab\_wrapper::matlab\_wrapper\_t::wrapped\_pffitfilt\_t, 491  
         process, 492  
         update\_cfg, 492  
 fcn\_process\_ss  
     matlab\_wrapper::matlab\_wrapper\_t::wrapped\_pffitfilt\_t, 491  
         fbpow, 494  
         fftfbpow\_t, 494  
 fcn\_process\_sw  
     matlab\_wrapper::matlab\_wrapper\_t::wrapped\_pffitfilt\_t, 491  
         fbpow, 494  
         fftfbpow\_t, 494  
 fcn\_process\_ws  
     matlab\_wrapper::matlab\_wrapper\_t::wrapped\_pffitfilt\_t, 491  
         fbpow, 494  
         fftfilt\_t  
             fftfilter::fftfilter\_t, 497  
             filteropt\_t,  
                 irs\_length, 93  
                 irs\_validator, 93  
 fcn\_release  
     matlab\_wrapper::matlab\_wrapper\_t::wrapped\_pffitfilt\_t, 491  
         filteropt\_t, 1548  
         fftfilter::fftfilter\_t, 495  
             channels, 496  
 fcn\_terminate  
     matlab\_wrapper::matlab\_wrapper\_t::wrapped\_pffitfilt\_t, 497  
         fftfilter\_t, 495

ffflen, 496  
fragsize, 496  
irslen, 496  
process, 496  
fftfilter::interface\_t, 497  
    ffflen, 499  
    ffflen\_final, 499  
    interface\_t, 498  
    irs, 499  
    patchbay, 499  
    prepare, 498  
    process, 498  
    update, 499  
fftfilter\_t  
    fftfilter::fftfilter\_t, 495  
    MHAFilter::fftfilter\_t, 874  
fftfilterbank, 94  
fftfilterbank.cpp, 1549  
fftfilterbank::fftfb\_interface\_t, 500  
    algo, 502  
    fftfb\_interface\_t, 500  
    nbands, 502  
    nchannels, 502  
    patchbay, 502  
    prepare, 501  
    prepared, 502  
    process, 501, 502  
    release, 501  
    return\_imag, 502  
    update\_cfg, 502  
fftfilterbank::fftfb\_plug\_t, 503  
    fb\_acinfo, 505  
    fftfb\_plug\_t, 504  
    imag, 505  
    insert, 504  
    process, 504  
    return\_imag, 505  
    s\_out, 505  
fftfilterbank\_t  
    MHAFilter::fftfilterbank\_t, 879  
fftgains  
    equalize::cfg\_t, 460  
    equalize::freqgains\_t, 462  
ffflen  
    dc::dc\_t, 386  
    dc\_simple::level\_smoothen\_t, 406  
    doasvm\_feature\_extraction, 425  
    doasvm\_feature\_extraction\_config, 427  
    fftfilter::fftfilter\_t, 496  
    fftfilter::interface\_t, 499  
    level\_matching::level\_matching\_config\_t, 649  
        mhaconfig\_t, 849  
        MHAFilter::fftfilter\_t, 877  
        MHAFilter::fftfilterbank\_t, 882  
        MHAFilter::smoothspec\_t, 929  
        MHAOvIFilter::fftfb\_t, 1006  
        MHAOvIFilter::overlap\_save\_filterbank\_t::vars\_t, 1023  
        MHAParser::mhaconfig\_mon\_t, 1086  
        rmslevel::rmslevel\_t, 1393  
        smooth\_cepstrum::smooth\_cepstrum\_t, 1441  
        testplugin::config\_parser\_t, 1480  
ffflen\_final  
    fftfilter::interface\_t, 499  
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  
    spechnoise.cpp, 1701  
fifo  
    plugins::hoertech::acrec::acwriter\_t, 1382  
    wavwriter\_t, 1503  
fifo\_size  
    mha\_dblbuf\_t< FIFO >, 750  
fifolen  
    analysispath\_if\_t, 312  
    plugins::hoertech::acrec::acrec\_t, 1377  
    wavrec\_t, 1500  
fileformat  
    acsave::acsave\_t, 221  
filename  
    addsndfile::addsndfile\_if\_t, 252  
filename\_input  
    io\_file\_t, 605  
filename\_output  
    io\_file\_t, 605  
fill  
    rmslevel::rmslevel\_t, 1392  
fill\_info  
    MHAIOPortAudio::device\_info\_t, 953  
    MHAIOPortAudio::stream\_info\_t, 962  
filled  
    MHASignal::async\_rmslevel\_t, 1198  
filter  
    MHAFilter::adapt\_filter\_state\_t, 861

MHAFilter::adapt\_filter\_t, 863  
 MHAFilter::complex\_bandpass\_t, 870, 871  
 MHAFilter::fftfilter\_t, 875, 876  
 MHAFilter::fftfilterbank\_t, 880, 881  
 MHAFilter::filter\_t, 886, 887  
 MHAFilter::iir\_filter\_t, 896  
 filter\_activated  
 droptect\_t, 439  
 filter\_analytic  
 MHAOvIFilter::overlap\_save\_filterbank\_analytic, 1019  
 filter\_complex  
 gtfb\_analyzer.cpp, 1555  
 filter\_partitions  
 MHAFilter::partitioned\_convolution\_t, 915  
 filter\_real  
 gtfb\_analyzer.cpp, 1556  
 filter\_simd  
 gtfb\_simd.cpp, 1560  
 filter\_sisd\_complex  
 gtfb\_simd.cpp, 1559  
 filter\_sisd\_real  
 gtfb\_simd.cpp, 1560  
 filter\_t  
 MHAFilter::filter\_t, 885  
 filterbank  
 dc::dc\_vars\_t, 389  
 dc\_simple::dc\_if\_t, 395  
 filtered\_level  
 dc::dc\_vars\_t, 390  
 filtered\_powspec  
 droptect\_t, 439  
 filtered\_powspec\_mon  
 droptect\_t, 439  
 filtershapefun  
 mha\_fftfb.cpp, 1591  
 FINISHED  
 MHA\_TCP::Thread, 826  
 fir  
 calibrator\_runtime\_layer\_t, 337  
 calibrator\_variables\_t, 341  
 fir\_lp  
 MHAFilter, 107  
 firchannels  
 MHAFilter::fftfilterbank\_t, 882  
 firfir2ffflen  
 calibrator\_runtime\_layer\_t, 336  
 firfirlen  
 calibrator\_runtime\_layer\_t, 336  
 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, 1477  
 float\_mon\_t  
 MHAParser::float\_mon\_t, 1058  
 float\_t  
 MHA\_AC::float\_t, 728  
 MHAParser::float\_t, 1061  
 flush\_data  
 acsave::cfg\_t, 223  
 fmap\_t  
 AuditoryProfile::parser\_t::fmap\_t, 329  
 fmax  
 fshift::fshift\_t, 510  
 fshift\_hilbert::frequency\_translator\_t, 513  
 fmin  
 fshift::fshift\_t, 510  
 fshift\_hilbert::frequency\_translator\_t, 513  
 FMTsz  
 mha\_os.h, 1606  
 fname  
 acsave::acsave\_t, 221  
 for\_each  
 MHASignal, 139  
 force\_remove\_item  
 MHAParser::parser\_t, 1101  
 force\_resize  
 dc\_simple, 88  
 format  
 ac2lsl::type\_info, 179  
 io\_alsa\_t, 580  
 format\_name  
 wavwriter\_t, 1504  
 forward  
 lpc\_bl\_predictor\_config, 665  
 lpc\_burglattice\_config, 671  
 MHASignal::fft\_t, 1209  
 forward\_scale  
 MHASignal::fft\_t, 1209  
 fr  
 spec\_fader\_t, 1466  
 frac\_old  
 gsc\_adaptive\_stage::gsc\_adaptive\_stage, 536  
 frag\_out

MHAJack::client\_avg\_t, 966  
MHAJack::client\_noncont\_t, 970  
fragsize  
  alsa\_t< T >, 294  
  analysispath\_if\_t, 312  
  calibrator\_variables\_t, 342  
  db\_if\_t, 371  
  dbasync\_native::db\_if\_t, 375  
  fftfilter::fftfilter\_t, 496  
  io\_asterisk\_sound\_t, 595  
  io\_file\_t, 604  
  io\_parser\_t, 615  
  io\_tcp\_sound\_t, 633  
  mconv::MConv, 716  
  mhaconfig\_t, 848  
  MHAFilter::fftfilter\_t, 876  
  MHAFilter::fftfilterbank\_t, 882  
  MHAFilter::partitioned\_convolution\_t, 915  
  MHAFilter::resampling\_filter\_t, 926  
  MHAIOPortAudio::io\_portaudio\_t, 959  
  MHAJack::client\_t, 977  
  MHAParser::mhaconfig\_mon\_t, 1085  
  MHAPlugin\_Resampling::resampling\_if\_t, 1155  
  testplugin::config\_parser\_t, 1480  
fragsize\_in  
  MHAFilter::blockprocessing\_polyphase\_resampling\_t, 867  
fragsize\_out  
  MHAFilter::blockprocessing\_polyphase\_resampling\_t, 867  
fragsize\_ratio  
  MHAIOJackdb::io\_jack\_t, 948  
fragsize\_validator  
  MHAFilter::resampling\_filter\_t, 926  
frame  
  MHA\_AC::acspace2matrix\_t, 724  
frame\_data  
  alsa\_t< T >, 294  
framecnt  
  acsave::save\_var\_t, 227  
  adm\_if\_t, 275  
frameno  
  MHA\_AC::acspace2matrix\_t, 725  
  noise\_psd\_estimator::noise\_psd\_estimator\_t, 1314  
framerate  
  ac2osc\_t, 184  
frames  
  ac2wave\_t, 189  
  adaptive\_feedback\_canceller\_config, 246  
          gtfb\_analyzer::gtfb\_analyzer\_cfg\_t, 547  
          rt\_nlms\_t, 1415  
frameshift  
  fshift\_hilbert::hilbert\_shifter\_t, 518  
framework\_thread\_priority  
  dbasync\_native::db\_if\_t, 376  
  MHAPlugin\_Split::split\_t, 1178  
framework\_thread\_scheduler  
  dbasync\_native::db\_if\_t, 376  
  MHAPlugin\_Split::split\_t, 1177  
freq  
  audiometerbackend::audiometer\_if\_t, 317  
  plingploing::plingploing\_t, 1343  
freq2bin  
  Vector and matrix processing toolbox, 39  
freq\_offsets  
  rmslevel::rmslevel\_t, 1393  
freqgains\_t  
  equalize::freqgains\_t, 461  
freqResp  
  rohBeam::rohConfig, 1407  
frequency  
  sine\_t, 1433  
frequency\_response  
  MHAFilter::partitioned\_convolution\_t, 916  
frequency\_translator\_t  
  fshift\_hilbert::frequency\_translator\_t, 512  
FrequencyBinLowPass  
  windnoise::cfg\_t, 1507  
framing\_channel  
  adm\_rtconfig\_t, 278  
front\_channels  
  adm\_if\_t, 273  
  adm\_rtconfig\_t, 278  
frozen\_noise\_cfg\_t, 346  
frozennoise\_length  
  noise\_t, 1316  
fs  
  MHAFilter::o1\_ar\_filter\_t, 905  
fs\_  
  MHAOvIFilter::fspacing\_t, 1017  
fscale  
  gtfb\_simple\_t, 569  
  MHAOvIFilter::fftfb\_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, 1549  
 fshift.hh, 1549  
 fshift::fshift\_config\_t, 505  
 ~fshift\_config\_t, 506  
 delta\_phi, 507  
 delta\_phi\_total, 507  
 df, 507  
 fshift\_config\_t, 506  
 kmax, 507  
 kmin, 507  
 process, 506  
 fshift::fshift\_t, 508  
 ~fshift\_t, 509  
 df, 510  
 fmax, 510  
 fmin, 510  
 fshift\_t, 509  
 m\_df, 511  
 m\_fmax, 511  
 m\_fmin, 510  
 patchbay, 510  
 prepare, 509  
 process, 509  
 release, 510  
 update\_cfg, 510  
 fshift\_config\_t  
     fshift::fshift\_config\_t, 506  
 fshift\_hilbert, 95  
 fshift\_hilbert.cpp, 1550  
 fshift\_hilbert::frequency\_translator\_t, 511  
     df, 513  
     fmax, 513  
     fmin, 513  
     frequency\_translator\_t, 512  
     irslen, 514  
     patchbay, 513  
     phasemode, 514  
     prepare, 513  
     process, 512  
     release, 513  
     update, 513  
 fshift\_hilbert::hilbert\_shifter\_t, 514  
     ~hilbert\_shifter\_t, 516  
     analytic, 516  
     delta\_phi, 518  
     delta\_phi\_total, 518  
     df, 517  
     frameshift, 518  
     fullspec, 516  
     hilbert\_shifter\_t, 515  
     kmax, 518  
     kmin, 517  
     mhafft, 517  
     mixw\_ref, 517  
     mixw\_shift, 517  
     plan\_spec2analytic, 517  
     process, 516  
     shifted, 516  
 fshift\_t  
     fshift::fshift\_t, 509  
 fspacing\_t  
     MHAOvIFilter::fspacing\_t, 1015  
 ft  
     spec2wave\_t, 1464  
     wave2spec\_t, 1496  
 ftype  
     MHAOvIFilter::ffffb\_vars\_t, 1008  
 fu  
     rt\_nlms\_t, 1416  
 fu\_previous  
     rt\_nlms\_t, 1416  
 fuflt  
     rt\_nlms\_t, 1416  
 fullname  
     dynamiclib\_t, 446  
     MHAParser::base\_t, 1037  
     plugindescription\_t, 1355  
 fullspec  
     fshift\_hilbert::hilbert\_shifter\_t, 516  
 fun1  
     plingploing::plingploing\_t, 1343  
 fun1\_key  
     plingploing::if\_t, 1338  
     plingploing::plingploing\_t, 1343  
 fun1\_range  
     plingploing::if\_t, 1339  
     plingploing::plingploing\_t, 1343  
 fun2  
     plingploing::plingploing\_t, 1343  
 fun2\_key  
     plingploing::if\_t, 1339  
     plingploing::plingploing\_t, 1343  
 fun2\_range  
     plingploing::if\_t, 1339  
     plingploing::plingploing\_t, 1343  
 fun\_t  
     MHAWindow::fun\_t, 1291  
 funs  
     MHAOvIFilter::scale\_var\_t, 1026  
 fw\_cmd  
     fw\_t, 525  
 fw\_exiting

fw\_t, 521  
fw\_fragsize  
    io\_alsa\_t, 578  
    MHAIOJack::io\_jack\_t, 941  
fw\_running  
    fw\_t, 521  
fw\_samplerate  
    io\_alsa\_t, 578  
    MHAIOJack::io\_jack\_t, 941  
fw\_sleep  
    fw\_t, 525  
fw\_sleep\_cmd  
    fw\_t, 524  
fw\_starting  
    fw\_t, 521  
fw\_stopped  
    fw\_t, 521  
fw\_stopping  
    fw\_t, 521  
fw\_t, 519  
    ~fw\_t, 521  
    ac, 526  
    async\_poll\_msg, 524  
    async\_read, 524  
    b\_exit\_request, 527  
    cfin, 526  
    cfout, 526  
    dump\_mha, 526  
    errorlog, 525  
    exec\_fw\_command, 523  
    exit\_on\_stop, 525  
    exit\_request, 522  
    fatallog, 526  
    fw\_cmd, 525  
    fw\_exiting, 521  
    fw\_running, 521  
    fw\_sleep, 525  
    fw\_sleep\_cmd, 524  
    fw\_starting, 521  
    fw\_stopped, 521  
    fw\_stopping, 521  
    fw\_t, 521  
    fw\_unprepared, 521  
    fw\_until, 525  
    fw\_until\_cmd, 524  
    get\_input\_signal\_dimension, 524  
    get\_parserstate, 524  
    inst\_name, 526  
    io\_error, 527  
    io\_lib, 527  
    io\_name, 525  
    load\_io\_lib, 524  
    load\_proc\_lib, 523  
    nchannels\_out, 525  
    parserstate, 525  
    patchbay, 527  
    plugin\_paths, 526  
    plugins, 526  
    prepare, 522  
    prepare\_vars, 524  
    proc\_error, 527  
    proc\_error\_string, 527  
    proc\_lib, 526  
    proc\_name, 525  
    process, 523  
    quit, 522  
    release, 522  
    start, 522  
    started, 523  
    state, 527  
    state\_t, 521  
    stop, 522  
    stopped, 522, 523  
fw\_unprepared  
    fw\_t, 521  
fw\_until  
    fw\_t, 525  
fw\_until\_cmd  
    fw\_t, 524  
fw\_vars\_t, 528  
    fw\_vars\_t, 528  
    lock\_channels, 528  
    lock\_srate\_fragsize, 528  
    pfragmentsize, 529  
    pinchannels, 529  
    psrate, 529  
    unlock\_channels, 528  
    unlock\_srate\_fragsize, 528  
fwcb  
    io\_asterisk\_t, 600  
    io\_tcp\_t, 638

G

    doasvm\_feature\_extraction\_config, 429

g

    coherence::cohflt\_t, 351

g50

    dc\_simple::dc\_vars\_t, 401

g80

    dc\_simple::dc\_vars\_t, 401

G\_ERRNO

    MHA\_TCP, 102

G\_length

doasvm\_feature\_extraction\_config, 427  
 gain, 95  
     alsa\_t< T >, 294  
     calibrator\_runtime\_layer\_t, 337  
     coherence::cohflt\_t, 352  
     delaysum\_spec::delaysum\_spec\_if\_t, 416  
     fader\_wave::fader\_wave\_if\_t, 487  
     multibandcompressor::plugin\_signals\_t,  
         1304  
 gain.cpp, 1550  
 gain::gain\_if\_t, 529  
     gain\_if\_t, 530  
     gains, 531  
     patchbay, 531  
     prepare, 531  
     process, 530  
     release, 531  
     update\_gain, 531  
     update\_minmax, 531  
     vmax, 532  
     vmin, 531  
 gain::scaler\_t, 532  
     scaler\_t, 532  
 gain\_ac  
     ac2wave\_if\_t, 187  
     ac2wave\_t, 190  
 gain\_delay  
     coherence::cohflt\_t, 352  
 gain\_if\_t  
     gain::gain\_if\_t, 530  
 gain\_in  
     ac2wave\_if\_t, 187  
     ac2wave\_t, 190  
 gain\_min  
     smooth\_cepstrum::smooth\_cepstrum\_t,  
         1442  
 gain\_min\_db  
     smooth\_cepstrum::smooth\_cepstrum\_if\_t,  
         1438  
     smooth\_cepstrum::smooth\_params, 1448  
 gain\_spec\_  
     cfg\_t, 345  
 gain\_wave\_  
     cfg\_t, 345  
 gain\_wiener  
     smooth\_cepstrum::smooth\_cepstrum\_t,  
         1444  
 gainrule  
     dc::dc\_vars\_t, 390  
     dc\_simple::dc\_if\_t, 395  
 gains  
     adaptive\_feedback\_canceller, 243  
     gain::gain\_if\_t, 531  
     shadowfilter\_end::cfg\_t, 1427  
     spec\_fader\_t, 1466  
 gaintable.cpp, 1550  
     convert\_f2logf, 1550  
     isempty, 1551  
 gaintable.h, 1551  
 gaintable\_t  
     DynComp::gaintable\_t, 455  
 gamma\_flt\_t  
     MHAFilter::gamma\_flt\_t, 889  
 gamma\_post  
     smooth\_cepstrum::smooth\_cepstrum\_t,  
         1443  
 gauss  
     MHAOvIFilter::ShapeFun, 122  
 GCC\_end  
     doasvm\_feature\_extraction\_config, 428  
 GCC\_start  
     doasvm\_feature\_extraction\_config, 427  
 gcd  
     audiometerbackend, 83  
     dbasync\_native, 86  
     MHAFilter, 108  
 generatemhaplugindoc.cpp, 1551  
     conv2latex, 1552  
     create\_latex\_doc, 1553  
     main, 1553  
     print\_plugin\_references, 1552  
 generator  
     audiometerbackend, 83  
 get  
     testplugin::config\_parser\_t, 1479  
 get\_A  
     MHAFilter::gamma\_flt\_t, 891  
 get\_a  
     MHAParser::base\_t::replace\_t, 1040  
 get\_ac  
     latex\_doc\_t, 643  
     plug\_t, 1346  
 get\_accept\_event  
     MHA\_TCP::Server, 813  
 get\_adaptation\_ratio  
     adm\_rtconfig\_t, 278  
 get\_address  
     mha\_tcp::server\_t, 817  
 get\_all\_input\_ports  
     MHAIOJack::io\_jack\_t, 940  
     MHAIOJackdb::io\_jack\_t, 947  
 get\_all\_names\_from\_ac\_space

ac2lsl::ac2lsl\_t, 164  
get\_all\_output\_ports  
    MHAIOJack::io\_jack\_t, 940  
    MHAIOJackdb::io\_jack\_t, 947  
get\_all\_stream\_names  
    lsl2ac::lsl2ac\_t, 680  
get\_arg\_type\_and\_dimension  
    ac\_mul\_t, 193, 194  
get\_attack\_filter\_state  
    dc::dc\_t, 385  
get\_audiochannels  
    dc.cpp, 1538  
get\_available\_space  
    mha\_drifter\_fifo\_t< T >, 756  
    mha\_fifo\_lf\_t< T >, 765  
    mha\_fifo\_t< T >, 776  
get\_b  
    MHAParser::base\_t::replace\_t, 1040  
get\_bands  
    gtfb\_simd\_cfg\_t, 554  
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, 796  
get\_bw\_hz  
    MHAOvlFilter::fscale\_bw\_t, 1011  
get\_c1  
    MHAFilter::o1flt\_lowpass\_t, 908  
get\_c\_handle  
    MHAKernel::algo\_comm\_class\_t, 986  
get\_categories  
    io\_lib\_t, 610  
    io\_wrapper, 640  
    latex\_doc\_t, 642  
    plug\_wrapper, 1347  
    plug\_wrapperl, 1349  
    PluginLoader::mhapluginloader\_t, 1369  
get\_cdata  
    MHASignal::matrix\_t, 1232  
get\_cf\_fftbin  
    MHAOvlFilter::fspacing\_t, 1015  
get\_cf\_hz  
    MHAFilter::thirdoctave\_analyzer\_t, 931  
    MHAOvlFilter::fspacing\_t, 1016  
get\_cfin  
    MHAParser::mhapluginloader\_t, 1089  
get\_cfout  
    MHAParser::mhapluginloader\_t, 1089  
get\_channelconfig  
    MHAOvlFilter::overlap\_save\_filterbank\_t, 1021  
get\_channels  
    gtfb\_simd\_cfg\_t, 554  
get\_comm\_var  
    MHASignal::matrix\_t, 1226  
get\_configfile  
    PluginLoader::config\_file\_splitter\_t, 1360  
get\_configname  
    PluginLoader::config\_file\_splitter\_t, 1359  
get\_connected  
    io\_asterisk\_parser\_t, 589  
    io\_tcp\_parser\_t, 625  
get\_context  
    mha\_tcp::server\_t, 819  
get\_cpu\_load  
    MHAJack::client\_t, 976  
get\_current\_profile  
    AuditoryProfile::parser\_t, 326  
get\_decay\_filter\_state  
    dc::dc\_t, 385  
get\_delay  
    mha\_dblbuf\_t< FIFO >, 746  
get\_delays\_in  
    MHAIOJack::io\_jack\_t, 940  
get\_delays\_out  
    MHAIOJack::io\_jack\_t, 940  
get\_des\_fill\_count  
    mha\_drifter\_fifo\_t< T >, 756  
get\_documentation  
    io\_lib\_t, 610  
    io\_wrapper, 640  
    plug\_wrapper, 1347  
    plug\_wrapperl, 1349  
    PluginLoader::mhapluginloader\_t, 1369  
get\_ear  
    AuditoryProfile::parser\_t::ear\_t, 328  
    AuditoryProfile::profile\_t, 331  
get\_ef\_hz  
    MHAOvlFilter::fspacing\_t, 1016  
get\_endpoint  
    mha\_tcp::server\_t, 817  
get\_entries  
    algo\_comm\_t, 285  
    MHAKernel::algo\_comm\_class\_t, 988  
    MHAParser::keyword\_list\_t, 1070  
get\_error  
    algo\_comm\_t, 285  
    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, 803  
 get\_ffflen  
     MHAOvlFilter::fftfb\_t, 1005  
 get\_fifo\_size  
     mha\_dblbuf\_t< FIFO >, 746  
 get\_fill\_count  
     mha\_drifter\_fifo\_t< T >, 756  
     mha\_fifo\_lf\_t< T >, 765  
     mha\_fifo\_t< T >, 776, 778  
 get\_fmap  
     AuditoryProfile::parser\_t::fmap\_t, 330  
 get\_fragsize  
     MHAJack::client\_t, 974  
 get\_frames  
     gtfb\_simd\_cfg\_t, 554  
 get\_frequencies  
     AuditoryProfile::fmap\_t, 325  
 get\_fun  
     MHAOvlFilter::scale\_var\_t, 1025  
 get\_gain  
     DynComp::gaintable\_t, 456, 457  
 get\_gf  
     gtfb\_simple\_rt\_t, 563  
 get\_handle  
     plug\_t, 1346  
 get\_idx  
     level\_matching::channel\_pair, 646  
 get\_index  
     MHAParser::keyword\_list\_t, 1070  
     MHASignal::matrix\_t, 1231  
 get\_inner\_error  
     mha\_dblbuf\_t< FIFO >, 747  
 get\_inner\_size  
     mha\_dblbuf\_t< FIFO >, 746  
 get\_input\_channels  
     mha\_dblbuf\_t< FIFO >, 746  
 get\_input\_fifo\_fill\_count  
     mha\_dblbuf\_t< FIFO >, 747  
 get\_input\_fifo\_space  
     mha\_dblbuf\_t< FIFO >, 747  
 get\_input\_signal\_dimension  
     fw\_t, 524  
 get\_interface  
     MHA\_TCP::Server, 813  
 get\_iofun  
     DynComp::gaintable\_t, 457  
 get\_irs  
     MHAFilter::fftfilterbank\_t, 881  
 get\_last\_name  
     MHAParser::mhaplugloader\_t, 1089  
 get\_last\_output  
     MHAFilter::o1flt\_lowpass\_t, 908  
 get\_latex\_doc  
     latex\_doc\_t, 642  
 get\_len\_A  
     MHAFilter::filter\_t, 887  
 get\_len\_B  
     MHAFilter::filter\_t, 887  
 get\_length  
     MHASignal::uint\_vector\_t, 1257  
 get\_level  
     addsndfile::level\_adapt\_t, 256  
     audiometerbackend::level\_adapt\_t, 319  
     fader\_wave::level\_adapt\_t, 489  
 get\_level\_in\_db  
     dc::dc\_t, 384  
 get\_level\_in\_db\_adjusted  
     dc::dc\_t, 385  
 get\_libname  
     PluginLoader::config\_file\_splitter\_t, 1359  
 get\_local\_address  
     io\_asterisk\_parser\_t, 587  
     io\_tcp\_parser\_t, 624  
 get\_local\_port  
     io\_asterisk\_parser\_t, 587  
     io\_tcp\_parser\_t, 624  
 get\_longmsg  
     MHA\_Error, 762  
 get\_ltass\_gain\_db  
     MHAOvlFilter::fftfb\_t, 1004  
 get\_main\_category  
     latex\_doc\_t, 642  
 get\_mapping  
     MHASignal::loop\_wavefragment\_t, 1219  
 get\_max\_fill\_count  
     mha\_fifo\_t< T >, 777  
 get\_min\_fill\_count  
     mha\_drifter\_fifo\_t< T >, 757  
 get\_mismatch  
     level\_matching::channel\_pair, 646  
 get\_msg  
     MHA\_Error, 762  
 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, 384

get\_nch  
    dc::dc\_t, 384

get\_nelements  
    MHASignal::matrix\_t, 1227

get\_noise\_model\_func  
    rohBeam::rohBeam, 1399

get\_nreals  
    MHASignal::matrix\_t, 1231

get\_num\_accepted\_connections  
    mha\_tcp::server\_t, 818

get\_origname  
    PluginLoader::config\_file\_splitter\_t, 1359

get\_os\_event  
    MHA\_TCP::Timeout\_Event, 829  
    MHA\_TCP::Wakeup\_Event, 834

get\_outer\_size  
    mha\_dblbuf\_t< FIFO >, 746

get\_output\_channels  
    mha\_dblbuf\_t< FIFO >, 747

get\_output\_fifo\_fill\_count  
    mha\_dblbuf\_t< FIFO >, 747

get\_output\_fifo\_space  
    mha\_dblbuf\_t< FIFO >, 747

get\_parser\_tab  
    latex\_doc\_t, 643

get\_parser\_var  
    latex\_doc\_t, 643

get\_parserstate  
    fw\_t, 524

get\_paths  
    pluginbrowser\_t, 1353

get\_peer\_address  
    MHA\_TCP::Connection, 803

get\_peer\_port  
    MHA\_TCP::Connection, 803

get\_physical\_input\_ports  
    MHAIOJack::io\_jack\_t, 940  
    MHAIOJackdb::io\_jack\_t, 947

get\_physical\_output\_ports  
    MHAIOJack::io\_jack\_t, 940  
    MHAIOJackdb::io\_jack\_t, 947

get\_plugins  
    pluginbrowser\_t, 1354

get\_port  
    MHA\_TCP::Server, 813  
    mha\_tcp::server\_t, 817

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, 1346

get\_process\_wave  
    plug\_t, 1345

get\_rdata  
    MHASignal::matrix\_t, 1231

get\_read\_event  
    MHA\_TCP::Connection, 803

get\_read\_ptr  
    mha\_fifo\_t< T >, 778

get\_resynthesis\_gain  
    MHAFilter::gamma\_flt\_t, 891

get\_rmslevel\_filter\_state  
    dc::dc\_t, 385

get\_server\_port\_open  
    io\_asterisk\_parser\_t, 588  
    io\_tcp\_parser\_t, 625

get\_short\_name  
    MHAJack::port\_t, 983

get\_signal  
    MHAParser\_Split::domain\_handler\_t,  
        1163

get\_size  
    MHASignal::waveform\_t, 1271

get\_srate  
    MHAJack::client\_t, 974

get\_time\_correction  
    lsl2ac::save\_var\_t, 686

get\_type  
    MHAParser::window\_t, 1137

get\_value  
    MHAParser::keyword\_list\_t, 1069

get\_values  
    AuditoryProfile::fmap\_t, 325

get\_var  
    algo\_comm\_t, 283  
    MHAKernel::algo\_comm\_class\_t, 987  
    testplugin::ac\_parser\_t, 1477

get\_var\_double  
    algo\_comm\_t, 285

**MHAKernel::algo\_comm\_class\_t**, 988  
**get\_var\_float**  
  **algo\_comm\_t**, 284  
    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**, 1381  
**get\_vF**  
  **DynComp::gaintable\_t**, 457  
**get\_vL**  
  **DynComp::gaintable\_t**, 457  
**get\_weights**  
  **MHAFilter::complex\_bandpass\_t**, 870  
  **MHAFilter::gamma\_flt\_t**, 891  
**get\_window**  
  **MHAParser::window\_t**, 1136, 1137  
**get\_window\_data**  
  **windowselector\_t**, 1515  
**get\_write\_event**  
  **MHA\_TCP::Connection**, 803  
**get\_write\_ptr**  
  **mha\_fifo\_t< T >**, 777  
**get\_xlimits**  
  **MHATableLookup::xy\_table\_t**, 1285  
**get\_xruns**  
  **MHAJack::client\_t**, 974  
**get\_xruns\_reset**  
  **MHAJack::client\_t**, 975  
**get\_zeropadding**  
  **wave2spec\_t**, 1495  
**getcipd**  
  coherence, 84  
**getdata**  
  **MHASignal::uint\_vector\_t**, 1258  
**getfullname**  
  **PluginLoader::mhaplugloader\_t**, 1368  
**getmodulename**  
  **dynamiclib\_t**, 445  
**Getmsg**  
  **mha\_error.hh**, 1589  
**getname**  
**dynamiclib\_t**, 445  
**MHA\_AC::ac2matrix\_t**, 720  
**getusername**  
  **MHA\_AC::ac2matrix\_t**, 720  
**getvar**  
  **acmon::ac\_monitor\_t**, 205  
  **MHA\_AC::ac2matrix\_helper\_t**, 718  
**GF**  
  **MHAFilter::gamma\_flt\_t**, 892  
**gf**  
  **gtfb\_simple\_rt\_t**, 565  
**gf\_internals**  
  **gtfb\_simple\_t**, 570  
**GITCOMMITHASH**  
  **mha\_git\_commit\_hash.cpp**, 1597  
**GLR**  
  **smooth\_cepstrum::smooth\_cepstrum\_t**,  
    1445  
**GLRDebug**  
  **noise\_psd\_estimator::noise\_psd\_estimator\_t**,  
    1313  
**GLRExp**  
  **noise\_psd\_estimator::noise\_psd\_estimator\_t**,  
    1314  
  **smooth\_cepstrum::smooth\_cepstrum\_t**,  
    1445  
**Grad**  
  **gsc\_adaptive\_stage::gsc\_adaptive\_stage**,  
    539  
**grad**  
  **gsc\_adaptive\_stage::gsc\_adaptive\_stage**,  
    539  
**GREETING\_TEXT**  
  **mhamain.cpp**, 1687  
**groupdelay**  
  **delaysum\_spec::delaysum\_spec\_if\_t**, 415  
**groupdelay\_t**  
  **MHASignal::schroeder\_t**, 1241  
**gsc\_adaptive\_stage**, 95  
  **DELT**, 96  
  **gsc\_adaptive\_stage::gsc\_adaptive\_stage**,  
    534  
**gsc\_adaptive\_stage.cpp**, 1554  
**gsc\_adaptive\_stage.hh**, 1554  
**gsc\_adaptive\_stage::gsc\_adaptive\_stage**,  
  533  
  **~gsc\_adaptive\_stage**, 535  
  **ac**, 536  
  **alp**, 538  
  **bufSize**, 536  
  **d**, 539

desired\_chan, 537  
doCircularComp, 537  
E, 539  
e, 539  
E2, 539  
e\_out, 540  
frac\_old, 536  
Grad, 539  
grad, 539  
gsc\_adaptive\_stage, 534  
insert, 536  
lenNewSamps, 536  
lenOldSamps, 536  
mha\_fft, 537  
mu, 537  
nchan, 537  
nfreq, 537  
P, 540  
process, 535  
Psum, 540  
useVAD, 538  
vadName, 538  
W, 538  
X, 538  
x, 538  
Y, 538  
y, 539  
gsc\_adaptive\_stage::gsc\_adaptive\_stage\_if, 541  
~gsc\_adaptive\_stage\_if, 542  
alp, 545  
doCircularComp, 544  
gsc\_adaptive\_stage\_if, 542  
lenOldSamps, 544  
mu, 544  
on\_model\_param\_valuechanged, 544  
patchbay, 544  
prepare, 543  
process, 543  
release, 543  
update\_cfg, 544  
useVAD, 545  
vadName, 545  
gsc\_adaptive\_stage\_if  
  gsc\_adaptive\_stage::gsc\_adaptive\_stage\_if, 542  
gsc\_adaptive\_stage\_if.cpp, 1554  
gsc\_adaptive\_stage\_if.hh, 1554  
gt  
  dc::dc\_t, 385  
gtdata  
  dc::dc\_vars\_t, 388  
gtfb\_analyzer, 96  
gtfb\_analyzer.cpp, 1554  
  filter\_complex, 1555  
  filter\_real, 1556  
gtfb\_analyzer::gtfb\_analyzer\_cfg\_t, 545  
  ~gtfb\_analyzer\_cfg\_t, 547  
  bands, 547  
  channels, 547  
  coeff, 548  
  cvalue, 548  
  frames, 547  
  gtfb\_analyzer\_cfg\_t, 546  
  norm\_phase, 548  
  order, 548  
  s\_out, 548  
  state, 549  
  states, 547  
gtfb\_analyzer::gtfb\_analyzer\_t, 549  
  coeff, 552  
  gtfb\_analyzer\_t, 551  
  norm\_phase, 552  
  order, 552  
  patchbay, 551  
  prepare, 551  
  prepared, 552  
  process, 551  
  update\_cfg, 551  
gtfb\_analyzer\_cfg\_t  
  gtfb\_analyzer::gtfb\_analyzer\_cfg\_t, 546  
gtfb\_analyzer\_t  
  gtfb\_analyzer::gtfb\_analyzer\_t, 551  
gtfb\_simd.cpp, 1556  
  add4f, 1558  
  check\_alignment, 1558  
  filter\_simd, 1560  
  filter\_sisd\_complex, 1559  
  filter\_sisd\_real, 1560  
  mul4f, 1558  
  MXCSR\_DAZ, 1558  
  MXCSR\_FTZ, 1558  
  sub4f, 1558  
gtfb\_simd\_cfg\_t, 552  
  ~gtfb\_simd\_cfg\_t, 554  
  bands, 555  
  bandsXchannels, 555  
  channels, 555  
  get\_bands, 554  
  get\_channels, 554  
  get\_frames, 554  
  gtfb\_simd\_cfg\_t, 553, 554

icoefficients, 556  
 iinputs, 556  
 istates, 556  
 large\_array, 557  
 norm\_phase, 556  
 operator=, 555  
 order, 555  
 process, 555  
 rcoefficients, 556  
 rinputs, 556  
 rstates, 556  
 s\_out, 557  
 sout\_buf, 557  
**gtfb\_simd\_t**, 558  
 coeff, 560  
**gtfb\_simd\_t**, 559  
 norm\_phase, 560  
 order, 560  
 patchbay, 559  
 prepare, 559  
 prepared, 559  
 process, 559  
 update\_cfg, 559  
**gtfb\_simple\_bridge.cpp**, 1561  
**gtfb\_simple\_rt\_t**, 560  
 \_ac, 565  
 \_order, 563  
 \_pre\_stages, 564  
 ac\_resynthesis\_gain, 565  
 acbw, 564  
 accf, 564  
 cLTASS, 565  
 duplicate\_vector, 563  
 element\_gain\_name\_, 565  
 get\_gf, 563  
 gf, 565  
**gtfb\_simple\_rt\_t**, 561  
 imag, 564  
 input, 564  
 nbands, 564  
 output, 564  
 post\_plugin, 562  
 pre\_plugin, 562  
**gtfb\_simple\_t**, 566  
 cLTASS, 570  
 desired\_delay, 569  
 element\_gain\_name, 569  
 fscale, 569  
 gf\_internals, 570  
**gtfb\_simple\_t**, 567  
 name\_, 570  
 order, 569  
 plug, 569  
 prepare, 568  
 prestages, 569  
 process, 568  
 release, 568  
 resynthesis\_gain, 570  
 setlock, 568  
**gtmin**  
 dc::dc\_vars\_t, 388  
**gtstep**  
 dc::dc\_vars\_t, 388  
**h**  
 dynamiclib\_t, 446  
 MHASignal::hilbert\_t, 1215  
**H\_ERRNO**  
 MHA\_TCP, 102  
**hamming**  
 MHAWindow, 155  
**hamming\_t**  
 MHAWindow::hamming\_t, 1293  
**handle**  
 algo\_comm\_t, 280  
**handler**  
 osc\_variable\_t, 1324, 1325  
**hann**  
 hann.cpp, 1562  
 hann.h, 1563  
 MHAOvlFilter::ShapeFun, 121  
**hann.cpp**, 1562  
 hann, 1562  
 hannf, 1562  
 PI, 1562  
**hann.h**, 1562  
 hann, 1563  
 hannf, 1563  
**hann1**  
 plingploing::plingploing\_t, 1344  
**hann2**  
 plingploing::plingploing\_t, 1344  
**hannf**  
 hann.cpp, 1562  
 hann.h, 1563  
**hanning**  
 MHAWindow, 155  
**hanning\_ramps\_t**, 570  
 ~hanning\_ramps\_t, 571  
 hanning\_ramps\_t, 571  
 len\_a, 571  
 len\_b, 571  
 operator(), 571

ramp\_a, 571  
ramp\_b, 572  
hanning\_t  
    MHAWindow::hanning\_t, 1294  
hardlimit  
    softclipper\_t, 1458  
    softclipper\_variables\_t, 1460  
has Been\_modified  
    dc\_simple::dc\_if\_t, 394  
has Entry  
    MHAParser::parser\_t, 1104  
has Inner\_error  
    analysepath\_t, 308  
has Key  
    MHAKernel::comm\_var\_map\_t, 990  
has Parser  
    io\_wrapper, 640  
    plug\_wrapper, 1347  
    plug\_wrapperl, 1349  
    PluginLoader::mhaplugloader\_t, 1367  
has Process  
    io\_wrapper, 640  
    plug\_wrapper, 1348  
    plug\_wrapperl, 1350  
    PluginLoader::mhaplugloader\_t, 1367  
head Model\_sphere\_radius\_cm  
    rohBeam::rohBeam, 1400  
header  
    io\_asterisk\_sound\_t, 595  
    io\_tcp\_sound\_t, 632  
headModel  
    rohBeam::rohConfig, 1405  
help  
    MHAParser::base\_t, 1039  
HELP\_TEXT  
    mhamain.cpp, 1687  
hhCorrXpXp  
    rohBeam::rohConfig, 1407  
hifftwin  
    doasvm\_feature\_extraction\_config, 428  
hifftwin\_sum  
    doasvm\_feature\_extraction\_config, 428  
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, 515  
hilbert\_t  
    MHASignal::hilbert\_t, 1215  
HINSTANCE  
    mha\_plugin.hh, 1619  
HL  
    rmslevel, 159  
host  
    ac2osc\_t, 183  
    osc2ac\_t, 1319  
host\_port\_to\_sock\_addr  
    mha\_tcp.cpp, 1641  
hostApi  
    MHAIOPortAudio::device\_info\_t, 953  
Hs  
    MHAFilter::fftfilterbank\_t, 882  
HSTRERROR  
    MHA\_TCP, 102  
HTL  
    AuditoryProfile::parser\_t::ear\_t, 328  
    AuditoryProfile::profile\_t::ear\_t, 333  
hton  
    io\_asterisk\_sound\_t, 595  
    io\_tcp\_sound\_t, 632  
hw  
    MHAFilter::fftfilterbank\_t, 882  
hwin  
    doasvm\_feature\_extraction\_config, 428  
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, 1701  
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, 634

icoefficients  
     gtfb\_simd\_cfg\_t, 556

id  
     equalize::freqgains\_t, 462  
     mha\_channel\_info\_t, 741

id\_str  
     MHParse::base\_t, 1039

id\_string  
     MHParse::parser\_t, 1104

identity  
     MHASignal::schroeder\_t, 1243

identity.cpp, 1563

identity\_t, 572  
     identity\_t, 573  
     prepare, 573  
     process, 573  
     release, 573

idstr  
     mha\_channel\_info\_t, 741

idx  
     level\_matching::channel\_pair, 646

if\_t  
     plingploing::if\_t, 1337  
     testplugin::if\_t, 1482  
     windnoise::if\_t, 1510

iface  
     MHA\_TCP::Server, 814

ifft  
     doasvm\_feature\_extraction\_config, 428

ifftshift  
     ifftshift.cpp, 1563  
     ifftshift.h, 1564

ifftshift.cpp, 1563  
     ifftshift, 1563

ifftshift.h, 1563  
     ifftshift, 1564

Ignore  
     lsl2ac, 97

ignore  
     MHA\_TCP::Event\_Watcher, 809

ignored\_by  
     MHA\_TCP::Wakeup\_Event, 833

iinputs  
     gtfb\_simd\_cfg\_t, 556

iir\_filter\_state\_t  
     MHAFilter::iir\_filter\_state\_t, 893

iir\_filter\_t  
     MHAFilter::iir\_filter\_t, 895

iir\_ord1\_real\_t  
     MHAFilter::iir\_ord1\_real\_t, 899

iirfilter.cpp, 1564

iirfilter\_t, 574  
     iirfilter\_t, 574  
     prepare\_, 575  
     process, 575  
     release\_, 575

ilen  
     addsndfile::level\_adapt\_t, 256  
     audiometerbackend::level\_adapt\_t, 319  
     fader\_wave::level\_adapt\_t, 489

im  
     mha\_complex\_t, 742

imag  
     acsave::mat4head\_t, 225  
     fftfilterbank::fftfb\_plug\_t, 505  
     gtfb\_simple\_rt\_t, 564  
     MHASignal::matrix\_t, 1228–1230

imagfb  
     MHAOvFilter::overlap\_save\_filterbank\_analytic\_t,  
         1019

impulse\_response  
     MHAFilter::polyphase\_resampling\_t, 923  
     MHAFilter::transfer\_function\_t, 936

in\_buf  
     wave2spec\_t, 1497

in\_cfg  
     rohBeam::rohConfig, 1405  
     smooth\_cepstrum::smooth\_params, 1447

in\_spec  
     doasvm\_feature\_extraction\_config, 429  
     shadowfilter\_end::cfg\_t, 1426

in\_spec\_copy  
     shadowfilter\_begin::cfg\_t, 1422

inbuf  
     MHA\_TCP::Connection, 806

inch  
     mconv::MConv, 716  
     MHAJack::client\_t, 979

increase\_condition  
     mha\_fifo\_posix\_threads\_t, 772

increment  
     mha\_fifo\_posix\_threads\_t, 771  
     mha\_fifo\_thread\_platform\_t, 783

index  
     MHParse::keyword\_list\_t, 1070

index\_t  
     MHAFilter::partitioned\_convolution\_t::index\_t,  
         918

info  
     ac2lsl::save\_var\_base\_t, 170  
     ac2lsl::save\_var\_t< mha\_complex\_t >,  
         177

ac2lsl::save\_var\_t< T >, 174  
lsl2ac::save\_var\_t, 685  
init\_dynamic  
    rohBeam::rohConfig, 1404  
Init\_mha\_ruby  
    mha\_ruby.cpp, 1624  
init\_peer\_data  
    MHA\_TCP::Connection, 801  
initialize  
    MHA\_TCP::Server, 813  
inner2outer\_resampling  
    MHAPlugin\_Resampling::resampling\_t,  
        1158  
inner\_ac\_copy  
    analysepath\_t, 308  
inner\_error  
    analysepath\_t, 308  
    mha\_dbdbuf\_t< FIFO >, 750  
inner\_fragsize  
    MHAPlugin\_Resampling::resampling\_t,  
        1157  
inner\_in  
    MHASignal::doublebuffer\_t, 1206  
inner\_input  
    analysepath\_t, 307  
    dbasync\_native::dbasync\_t, 378  
inner\_out  
    MHASignal::doublebuffer\_t, 1206  
inner\_out\_domain  
    analysepath\_t, 308  
inner\_process  
    db\_t, 372  
    MHASignal::doublebuffer\_t, 1205  
inner\_process\_wave2spec  
    analysepath\_t, 307  
inner\_process\_wave2wave  
    analysepath\_t, 307  
inner\_signal  
    MHAPlugin\_Resampling::resampling\_t,  
        1158  
inner\_size  
    mha\_dbdbuf\_t< FIFO >, 749  
inner\_srate  
    MHAPlugin\_Resampling::resampling\_t,  
        1157  
input  
    ac\_proc::interface\_t, 198  
    gtfb\_simple\_rt\_t, 564  
    io\_parser\_t, 616  
    mha\_dbdbuf\_t< FIFO >, 748  
    MHAJack::port\_t, 981  
    MHASignal::loop\_wavefragment\_t, 1218  
    input\_cfg  
        MHAPlugin::plugin\_t< runtime\_cfg\_t >,  
            1151  
    input\_cfg\_  
        MHAPlugin::plugin\_t< runtime\_cfg\_t >,  
            1152  
    input\_channels  
        adm\_if\_t, 274  
        mha\_dbdbuf\_t< FIFO >, 750  
    input\_domain  
        PluginLoader::mhapluginloader\_t, 1367  
    input\_fifo  
        mha\_dbdbuf\_t< FIFO >, 750  
    input\_level  
        dc::dc\_vars\_t, 390  
    input\_portnames  
        MHAJack::client\_t, 980  
    input\_signal\_spec  
        MHAFilter::partitioned\_convolution\_t, 916  
    input\_signal\_wave  
        MHAFilter::partitioned\_convolution\_t, 916  
    input\_spec  
        testplugin::signal\_parser\_t, 1485  
    input\_to\_process  
        analysepath\_t, 309  
    input\_wave  
        testplugin::signal\_parser\_t, 1485  
    inputchannels  
        MHAFilter::fftfilterbank\_t, 882  
    inputPow  
        noise\_psd\_estimator::noise\_psd\_estimator\_t,  
            1312  
    inputSpec  
        noise\_psd\_estimator::noise\_psd\_estimator\_t,  
            1313  
    insert  
        acPooling\_wave\_config, 216  
        acSteer\_config, 232  
        adaptive\_feedback\_canceller\_config, 246  
        coherence::cohflt\_t, 350  
        fftfilterbank::fftfb\_plug\_t, 504  
        gsc\_adaptive\_stage::gsc\_adaptive\_stage,  
            536  
        ipc\_config, 673  
        MHA\_AC::ac2matrix\_t, 721  
        MHA\_AC::acspace2matrix\_t, 724  
        MHA\_AC::double\_t, 726  
        MHA\_AC::float\_t, 728  
        MHA\_AC::int\_t, 730  
        MHA\_AC::spectrum\_t, 733

MHA\_AC::stat\_t, 734  
 MHA\_AC::waveform\_t, 737  
 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, 1415  
 windnoise::if\_t, 1511  
 insert\_config\_vars  
 matlab\_wrapper::matlab\_wrapper\_t, 697  
 insert\_item  
 MHAParser::parser\_t, 1100  
 insert\_items  
 windowselector\_t, 1515  
 insert\_member  
 mha\_parser.hh, 1617  
 insert\_monitors  
 matlab\_wrapper::matlab\_wrapper\_t, 697  
**INSERT\_PATCH**  
 acConcat\_wave.cpp, 1520  
 acPooling\_wave.cpp, 1521  
 acSteer.cpp, 1524  
 acTransform\_wave.cpp, 1525  
 adaptive\_feedback\_canceller.cpp, 1525  
 doasvm\_classification.cpp, 1542  
 doasvm\_feature\_extraction.cpp, 1543  
 level\_matching.cpp, 1564  
 lpc.cpp, 1566  
 lpc\_bl\_predictor.cpp, 1567  
 lpc\_burg-lattice.cpp, 1568  
 smooth\_cepstrum.cpp, 1698  
 steerbf.cpp, 1705  
**INSERT\_VAR**  
 smooth\_cepstrum.cpp, 1698  
 insert\_var  
 algo\_comm\_t, 281  
 MHAKernel::algo\_comm\_class\_t, 986  
 testplugin::ac\_parser\_t, 1477  
 insert\_var\_double  
 algo\_comm\_t, 282  
 MHAKernel::algo\_comm\_class\_t, 987  
 insert\_var\_float  
 algo\_comm\_t, 281  
 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, 1321  
 insert\_vars  
 lsl2ac::save\_var\_t, 686  
 inspect  
 MHAFilter::complex\_bandpass\_t, 871  
 MHAFilter::gamma\_flt\_t, 891  
 MHASignal::delay\_t, 1201  
 inst\_name  
 fw\_t, 526  
 int\_data  
 testplugin::ac\_parser\_t, 1477  
 int\_mon\_t  
 MHAParser::int\_mon\_t, 1063  
 int\_t  
 MHA\_AC::int\_t, 730  
 MHAParser::int\_t, 1066  
 integrate  
 Vector and matrix processing toolbox, 43  
 intensity  
 mha\_signal.cpp, 1627  
 interface\_t  
 ac\_proc::interface\_t, 197  
 delay::interface\_t, 408  
 fftfilter::interface\_t, 498  
 multibandcompressor::interface\_t, 1301  
 route::interface\_t, 1409  
 interleaved  
 combc\_if\_t, 357  
 interleaved\_  
 combc\_t, 359  
 intermic\_distance\_cm  
 rohBeam::rohBeam, 1400  
 intern\_level  
 MHASignal::loop\_wavefragment\_t, 1221  
 internal\_fir  
 MHAFilter::smoothspec\_t, 929  
 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, 1640  
INVALID\_THREAD\_PRIORITY  
  dbasync\_native, 86  
  MHAPlugin\_Split, 136  
invalidate\_window\_data  
  windowselector\_t, 1516  
invert  
  coherence::vars\_t, 354  
invgain  
  alsa\_t< T >, 294  
inwave  
  lpc\_config, 674  
io  
  MHAJack, 112  
  MHAJack::client\_avg\_t, 964  
  MHAJack::client\_noncont\_t, 968  
io\_alsa\_t, 575  
  alsa\_start\_counter, 580  
  b\_process, 578  
  dev\_in, 579  
  dev\_out, 579  
  format, 580  
  fw fragsize, 578  
  fw\_samplerate, 578  
  io\_alsa\_t, 577  
  p\_in, 580  
  p\_out, 580  
  patchbay, 580  
  pcmlink, 580  
  prepare, 577, 578  
  priority, 580  
  proc\_event, 579  
  proc\_handle, 579  
  proc\_thread, 579  
  process, 578  
  release, 577  
  start, 577  
  start\_event, 579  
  start\_handle, 579  
  stop, 578  
  stop\_event, 579  
  stop\_handle, 579  
  thread\_start, 578  
io\_asterisk\_fwcb\_t, 580  
  ~io\_asterisk\_fwcb\_t, 582  
  io\_asterisk\_fwcb\_t, 581  
  io\_err, 585  
  proc\_err, 584  
  proc\_event, 584  
  proc\_handle, 584  
  process, 582  
          set\_errnos, 583  
          start, 582  
          start\_event, 584  
          start\_handle, 584  
          stop, 583  
          stop\_event, 584  
          stop\_handle, 584  
io\_asterisk\_parser\_t, 585  
  ~io\_asterisk\_parser\_t, 587  
  connected, 591  
  debug, 591  
  debug\_file, 592  
  debug\_filename, 592  
  get\_connected, 589  
  get\_local\_address, 587  
  get\_local\_port, 587  
  get\_server\_port\_open, 588  
  io\_asterisk\_parser\_t, 587  
  local\_address, 591  
  local\_port, 591  
  peer\_address, 592  
  peer\_port, 592  
  server\_port\_open, 591  
  set\_connected, 589  
  set\_local\_port, 588  
  set\_new\_peer, 590  
  set\_server\_port\_open, 588  
io\_asterisk\_sound\_t, 592  
  ~io\_asterisk\_sound\_t, 594  
  chunkbytes\_in, 594  
  fragsize, 595  
  header, 595  
  hton, 595  
  io\_asterisk\_sound\_t, 593  
  ntoh, 595  
  num\_inchannels, 596  
  num\_outchannels, 596  
  output\_data, 596  
  prepare, 594  
  release, 594  
  s\_in, 596  
  samplerate, 595  
io\_asterisk\_t, 596  
  ~io\_asterisk\_t, 598  
  accept\_loop, 599  
  connection\_loop, 599  
  fwcb, 600  
  io\_asterisk\_t, 597  
  notify\_release, 600  
  notify\_start, 600  
  notify\_stop, 600

parse, 599  
 parser, 599  
 prepare, 598  
 release, 598  
 server, 600  
 sound, 599  
 start, 598  
 stop, 598  
 thread, 600  
**io\_context**  
 mha\_tcp::server\_t, 820  
**io\_err**  
 io\_asterisk\_fwcbs\_t, 585  
 io\_tcp\_fwcbs\_t, 621  
**io\_error**  
 fw\_t, 527  
**IO\_ERROR\_JACK**  
 mhajack.h, 1686  
**IO\_ERROR\_MHAJACKLIB**  
 mhajack.h, 1686  
**io\_file\_t**, 601  
 ~io\_file\_t, 603  
 b\_prepared, 606  
 filename\_input, 605  
 filename\_output, 605  
 fragsize, 604  
**io\_file\_t**, 602  
 length, 606  
 nchannels\_file\_in, 604  
 nchannels\_in, 604  
 nchannels\_out, 604  
 output\_sample\_format, 605  
 prepare, 603  
 proc\_event, 605  
 proc\_handle, 605  
 release, 603  
 s\_file\_in, 606  
 s\_in, 606  
 s\_out, 606  
 samplerate, 604  
 setlock, 604  
 sf\_in, 606  
 sf\_out, 607  
 sfinf\_in, 607  
 sfinf\_out, 607  
 start, 603  
 start\_event, 605  
 start\_handle, 605  
 startsample, 606  
 stop, 603  
 stop\_event, 605  
 stop\_handle, 605  
 stopped, 603  
 strict\_channel\_match, 606  
 strict\_srate\_match, 606  
 total\_read, 607  
**io\_jack\_t**  
 MHAIOJack::io\_jack\_t, 939  
 MHAIOJackdb::io\_jack\_t, 946  
**io\_lib**  
 fw\_t, 527  
**io\_lib\_t**, 607  
 ~io\_lib\_t, 609  
 get\_categories, 610  
 get\_documentation, 610  
 io\_lib\_t, 609  
 IODestroy\_cb, 612  
 IOInit\_cb, 611  
 IOPrepare\_cb, 611  
 IORelease\_cb, 611  
 IOSetVar\_cb, 611  
 IOStart\_cb, 611  
 IOStop\_cb, 611  
 IOStrError\_cb, 611  
 lib\_data, 611  
 lib\_err, 610  
 lib\_handle, 611  
 lib\_str\_error, 610  
 plugin\_categories, 612  
 plugin\_documentation, 612  
 prepare, 609  
 release, 610  
 start, 610  
 stop, 610  
 test\_error, 610  
**io\_name**  
 fw\_t, 525  
**io\_parser\_t**, 612  
 ~io\_parser\_t, 614  
 b\_fw\_started, 616  
 b\_prepared, 617  
 b\_starting, 617  
 b\_stopped, 617  
 fragsize, 615  
 input, 616  
**io\_parser\_t**, 614  
 nchannels\_in, 615  
 nchannels\_out, 615  
 output, 616  
 patchbay, 617  
 prepare, 614  
 proc\_event, 615

proc\_handle, 615  
process\_frame, 615  
release, 614  
s\_in, 616  
s\_out, 616  
start, 614  
start\_event, 616  
start\_handle, 616  
started, 615  
stop, 614  
stop\_event, 616  
stop\_handle, 616  
stopped, 615  
io\_portaudio\_t  
    MHAIOPortAudio::io\_portaudio\_t, 956  
io\_tcp\_fwcb\_t, 617  
    ~io\_tcp\_fwcb\_t, 618  
    io\_err, 621  
    io\_tcp\_fwcb\_t, 618  
    proc\_err, 621  
    proc\_event, 620  
    proc\_handle, 621  
    process, 619  
    set\_errno, 619  
    start, 619  
    start\_event, 620  
    start\_handle, 621  
    stop, 620  
    stop\_event, 620  
    stop\_handle, 621  
io\_tcp\_parser\_t, 622  
    ~io\_tcp\_parser\_t, 623  
    connected, 628  
    debug, 627  
    debug\_file, 628  
    debug\_filename, 628  
    get\_connected, 625  
    get\_local\_address, 624  
    get\_local\_port, 624  
    get\_server\_port\_open, 625  
    io\_tcp\_parser\_t, 623  
    local\_address, 627  
    local\_port, 627  
    peer\_address, 628  
    peer\_port, 628  
    server\_port\_open, 628  
    set\_connected, 626  
    set\_local\_port, 624  
    set\_new\_peer, 626  
    set\_server\_port\_open, 625  
io\_tcp\_sound\_t, 629  
    ~io\_tcp\_sound\_t, 630  
    check\_sound\_data\_type, 631  
    chunkbytes\_in, 631  
    fragsize, 633  
    header, 632  
    hton, 632  
    io\_tcp\_sound\_t, 630  
    ntoh, 632  
    num\_inchannels, 633  
    num\_outchannels, 633  
    prepare, 631  
    release, 631  
    s\_in, 633  
    samplerate, 633  
    io\_tcp\_sound\_t::float\_union, 634  
        c, 634  
        f, 634  
        i, 634  
    io\_tcp\_t, 634  
        ~io\_tcp\_t, 636  
        accept\_loop, 637  
        connection\_loop, 637  
        fwcb, 638  
        io\_tcp\_t, 635  
        notify\_release, 638  
        notify\_start, 638  
        notify\_stop, 638  
        parse, 637  
        parser, 637  
        prepare, 636  
        release, 636  
        server, 638  
        sound, 637  
        start, 636  
        stop, 636  
        thread, 638  
    io\_wrapper, 639  
        ~io\_wrapper, 639  
        get\_categories, 640  
        get\_documentation, 640  
        has\_parser, 640  
        has\_process, 640  
        io\_wrapper, 639  
        parse, 640  
    iob  
        MHAJack::port\_t, 984  
    IODestroy  
        MHAIOalsa.cpp, 1649, 1651  
        MHAIOasterisk.cpp, 1654, 1656  
        MHAIOfile.cpp, 1659, 1660  
        MHAIOjack.cpp, 1663, 1664

MHAIOJackdb.cpp, 1667, 1668  
MHAIOParser.cpp, 1671, 1672  
MHAIOPortAudio.cpp, 1675, 1677  
MHAIOTCP.cpp, 1681, 1683  
**IODestroy\_cb**  
  io\_lib\_t, 612  
**IODestroy\_t**  
  mha\_io\_ifc.h, 1600  
**IOInit**  
  MHAIOalsa.cpp, 1648, 1649  
  MHAIOAsterisk.cpp, 1653, 1655  
  MHAIOFile.cpp, 1658, 1659  
  MHAIOJack.cpp, 1662, 1663  
  MHAIOJackdb.cpp, 1666, 1667  
  MHAIOParser.cpp, 1670, 1671  
  MHAIOPortAudio.cpp, 1674, 1676  
  MHAIOTCP.cpp, 1680, 1682  
**IOInit\_cb**  
  io\_lib\_t, 611  
**IOInit\_t**  
  mha\_io\_ifc.h, 1600  
**IOPrepare**  
  MHAIOalsa.cpp, 1648, 1650  
  MHAIOAsterisk.cpp, 1654, 1655  
  MHAIOFile.cpp, 1658, 1659  
  MHAIOJack.cpp, 1662, 1663  
  MHAIOJackdb.cpp, 1666, 1667  
  MHAIOParser.cpp, 1670, 1671  
  MHAIOPortAudio.cpp, 1675, 1676  
  MHAIOTCP.cpp, 1680, 1682  
**IOPrepare\_cb**  
  io\_lib\_t, 611  
**IOPrepare\_t**  
  mha\_io\_ifc.h, 1600  
**IOProcessEvent\_inner**  
  MHAIOJackdb::io\_jack\_t, 946  
**IOProcessEvent\_t**  
  mha\_io\_ifc.h, 1599  
**IORelease**  
  MHAIOalsa.cpp, 1648, 1650  
  MHAIOAsterisk.cpp, 1654, 1655  
  MHAIOFile.cpp, 1658, 1660  
  MHAIOJack.cpp, 1663, 1664  
  MHAIOJackdb.cpp, 1667, 1668  
  MHAIOParser.cpp, 1671, 1672  
  MHAIOPortAudio.cpp, 1675, 1676  
  MHAIOTCP.cpp, 1681, 1682  
**IORelease\_cb**  
  io\_lib\_t, 611  
**IORelease\_t**  
  mha\_io\_ifc.h, 1600  
**IOSetVar**  
  MHAIOalsa.cpp, 1649, 1650  
  MHAIOAsterisk.cpp, 1654, 1656  
  MHAIOFile.cpp, 1658, 1660  
  MHAIOJack.cpp, 1663, 1664  
  MHAIOJackdb.cpp, 1667, 1668  
  MHAIOParser.cpp, 1671, 1672  
  MHAIOPortAudio.cpp, 1675, 1677  
  MHAIOTCP.cpp, 1681, 1682  
**IOSetVar\_cb**  
  io\_lib\_t, 611  
**IOSetVar\_t**  
  mha\_io\_ifc.h, 1600  
**IOStart**  
  MHAIOalsa.cpp, 1648, 1650  
  MHAIOAsterisk.cpp, 1654, 1655  
  MHAIOFile.cpp, 1658, 1659  
  MHAIOJack.cpp, 1662, 1664  
  MHAIOJackdb.cpp, 1666, 1668  
  MHAIOParser.cpp, 1670, 1672  
  MHAIOPortAudio.cpp, 1675, 1676  
  MHAIOTCP.cpp, 1680, 1682  
**IOStart\_cb**  
  io\_lib\_t, 611  
**IOStart\_t**  
  mha\_io\_ifc.h, 1600  
**IOStartedEvent\_t**  
  mha\_io\_ifc.h, 1599  
**IOStop**  
  MHAIOalsa.cpp, 1648, 1650  
  MHAIOAsterisk.cpp, 1654, 1655  
  MHAIOFile.cpp, 1658, 1660  
  MHAIOJack.cpp, 1662, 1664  
  MHAIOJackdb.cpp, 1666, 1668  
  MHAIOParser.cpp, 1670, 1672  
  MHAIOPortAudio.cpp, 1675, 1676  
  MHAIOTCP.cpp, 1681, 1682  
**IOStop\_cb**  
  io\_lib\_t, 611  
**IOStop\_t**  
  mha\_io\_ifc.h, 1600  
**IOStoppedEvent**  
  MHAJack::client\_avg\_t, 965  
  MHAJack::client\_noncont\_t, 969  
**IOStoppedEvent\_t**  
  mha\_io\_ifc.h, 1599  
**IOStrError**  
  MHAIOalsa.cpp, 1649, 1650  
  MHAIOAsterisk.cpp, 1654, 1656  
  MHAIOFile.cpp, 1659, 1660  
  MHAIOJack.cpp, 1663, 1664

MHAIOJackdb.cpp, 1667, 1668  
MHAIOParser.cpp, 1671, 1672  
MHAIOPortAudio.cpp, 1675, 1677  
MHAIOTCP.cpp, 1681, 1683  
IOStrError\_cb  
    io\_lib\_t, 611  
IOStrError\_t  
    mha\_io\_ifc.h, 1600  
irs  
    fftfilter::interface\_t, 499  
    mconv::MConv, 716  
irs\_length  
    fftfilter, 93  
irs\_validator  
    fftfilter, 93  
irslen  
    fftfilter::fftfilter\_t, 496  
    fshift\_hilbert::frequency\_translator\_t, 514  
irslen\_inner2outer  
    MHAParser::Resampling::resampling\_if\_t, 1155  
irslen\_outer2inner  
    MHAParser::Resampling::resampling\_if\_t, 1155  
irswnd  
    MHAOvLFilter::overlap\_save\_filterbank\_t::vars\_t, 1023  
    smoothgains\_bridge::overlapadd\_if\_t, 1451  
is\_complex  
    MHA\_AC::ac2matrix\_helper\_t, 718  
    mha\_audio\_descriptor\_t, 738  
    plugins::hoertech::acrec::acwriter\_t, 1383  
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, 1383  
is\_playback\_active  
    MHASignal::loop\_wavefragment\_t, 1220  
is\_power\_of\_two  
    MHAUtils, 151  
is\_prepared  
    adm\_if\_t, 273  
    double2acvar::double2acvar\_t, 432  
MHAJack::client\_t, 976  
MHAParser::plugin\_t< runtime\_cfg\_t >, 1151  
PluginLoader::mhaplugloader\_t, 1369  
is\_prepared\_  
    MHAParser::plugin\_t< runtime\_cfg\_t >, 1152  
is\_running  
    osc\_server\_t, 1322  
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, 325  
    gaintable.cpp, 1551  
    MHAFilter::transfer\_function\_t, 935  
istates  
    gtfb\_simd\_cfg\_t, 556  
isval  
    MHAParser::kw\_t, 1073  
iter  
    adaptive\_feedback\_canceller\_config, 247  
    iterate\_lnn  
        audiometerbackend::lnn3rdoct\_t, 321  
iterator  
    MHA\_TCP::Event\_Watcher, 808  
j0  
    rohBeam, 159  
jack\_error\_handler  
    mhajack.cpp, 1683  
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, 671  
kappa\_block

lpc\_burglattice\_config, 671  
 kappa\_const  
     smooth\_cepstrum::smooth\_cepstrum\_if\_t, 1437  
     smooth\_cepstrum::smooth\_params, 1448  
 keyword\_list\_t  
     MHParse::keyword\_list\_t, 1069  
 kick\_condition  
     MHAParser\_Split::posix\_threads\_t, 1170  
 kick\_thread  
     MHAParser\_Split::dummy\_threads\_t, 1166  
     MHAParser\_Split::posix\_threads\_t, 1169  
     MHAParser\_Split::thread\_platform\_t, 1186  
 kicked  
     MHAParser\_Split::posix\_threads\_t, 1171  
 km  
     lpc\_bl\_predictor\_config, 666  
 kmax  
     fshift::fshift\_config\_t, 507  
     fshift\_hilbert::hilbert\_shifter\_t, 518  
 kmin  
     fshift::fshift\_config\_t, 507  
     fshift\_hilbert::hilbert\_shifter\_t, 517  
 kth\_smallest  
     MHASignal, 144  
 kw\_index2type  
     transducers.cpp, 1707  
 kw\_t  
     MHAParser::kw\_t, 1072, 1073

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, 489

l\_old  
     addsndfile::level\_adapt\_t, 257  
     audiometerbackend::level\_adapt\_t, 320  
     fader\_wave::level\_adapt\_t, 490

lambda

lpc\_burglattice, 669  
 lpc\_burglattice\_config, 671

lambda\_ceps  
     smooth\_cepstrum::smooth\_cepstrum\_t, 1444

lambda\_ceps\_prev  
     smooth\_cepstrum::smooth\_cepstrum\_t, 1444

lambda\_ml\_ceps  
     smooth\_cepstrum::smooth\_cepstrum\_t, 1443

lambda\_ml\_full  
     smooth\_cepstrum::smooth\_cepstrum\_t, 1443

lambda\_ml\_smooth  
     smooth\_cepstrum::smooth\_cepstrum\_t, 1443

lambda\_smoothing\_power  
     nlms\_t, 1307

lambda\_spec  
     smooth\_cepstrum::smooth\_cepstrum\_t, 1444

lambda\_thresh  
     smooth\_cepstrum::smooth\_cepstrum\_if\_t, 1437  
     smooth\_cepstrum::smooth\_params, 1447

large\_array  
     gtfb\_simd\_cfg\_t, 557

last\_complex\_bin  
     MHASignal::subsample\_delay\_t, 1254

last\_errormsg  
     MHAParser::parser\_t, 1104

last\_jack\_err  
     mhajack.cpp, 1684

last\_jack\_err\_msg  
     mhajack.cpp, 1684  
     mhajack.h, 1686

last\_name  
     MHAParser::mhapluginloader\_t, 1090

latex\_doc\_t, 641  
     ac, 643  
     get\_ac, 643  
     get\_categories, 642  
     get\_latex\_doc, 642  
     get\_main\_category, 642  
     get\_parser\_tab, 643  
     get\_parser\_var, 643  
     latex\_doc\_t, 641  
     latex\_pluginname, 643  
     loader, 644  
     parsername, 643  
     plugin\_macro, 644  
     pluginname, 643  
     strdom, 642

latex\_pluginname  
     latex\_doc\_t, 643

len

MHA\_AC::acspace2matrix\_t, 725  
 MHAFilter::filter\_t, 888

MHATableLookup::linear\_table\_t, 1278  
plingploing::plingploing\_t, 1342  
len\_A  
    MHAFilter::filter\_t, 888  
len\_a  
    hanning\_ramps\_t, 571  
len\_B  
    MHAFilter::filter\_t, 888  
len\_b  
    hanning\_ramps\_t, 571  
length  
    io\_file\_t, 606  
    MHASignal::uint\_vector\_t, 1258  
lenNewSamps  
    gsc\_adaptive\_stage::gsc\_adaptive\_stage,  
        536  
lenOldSamps  
    gsc\_adaptive\_stage::gsc\_adaptive\_stage,  
        536  
    gsc\_adaptive\_stage::gsc\_adaptive\_stage\_if,  
        544  
lev  
    noise\_t, 1316  
    sine\_t, 1433  
LEVEL  
    dc\_simple, 87  
level  
    addsndfile::addsndfile\_if\_t, 253  
    audiometerbackend::audiometer\_if\_t, 317  
    plingploing::if\_t, 1338  
    plingploing::plingploing\_t, 1344  
level\_adapt\_t  
    addsndfile::level\_adapt\_t, 255  
    audiometerbackend::level\_adapt\_t, 319  
    fader\_wave::level\_adapt\_t, 488  
level\_adaptor  
    addsndfile, 80  
    audiometerbackend, 83  
    fader\_wave, 92  
level\_in\_db  
    dc::dc\_t, 386  
level\_in\_db\_adjusted  
    dc::dc\_t, 386  
level\_matching, 96  
level\_matching.cpp, 1564  
    INSERT\_PATCH, 1564  
    PATCH\_VAR, 1564  
level\_matching.hh, 1565  
level\_matching::channel\_pair, 644  
    channel\_pair, 644, 645  
    get\_idx, 646  
get\_mismatch, 646  
idx, 646  
mismatch, 647  
update\_mismatch, 645, 646  
level\_matching::level\_matching\_config\_t, 647  
    ~level\_matching\_config\_t, 648  
ffflen, 649  
level\_matching\_config\_t, 648  
lp, 649  
pairings, 649  
process, 648, 649  
range, 649  
tmp, 649  
level\_matching::level\_matching\_t, 650  
    ~level\_matching\_t, 651  
channels, 653  
level\_matching\_t, 651  
lp\_level\_tau, 653  
lp\_signal\_fpass, 653  
lp\_signal\_fstop, 653  
lp\_signal\_order, 653  
patchbay, 653  
prepare, 652  
process, 651, 652  
range, 653  
release, 652  
update\_cfg, 652  
level\_matching\_config\_t  
    level\_matching::level\_matching\_config\_t,  
        648  
level\_matching\_t  
    level\_matching::level\_matching\_t, 651  
level\_mode\_t  
    MHASignal::loop\_wavefragment\_t, 1217  
level\_mon  
    droptect\_t, 439  
level\_smoothen\_t  
    dc\_simple::level\_smoothen\_t, 405  
level\_spec  
    dc\_simple::level\_smoothen\_t, 406  
level\_wave  
    dc\_simple::level\_smoothen\_t, 406  
levelmeter.cpp, 1565  
    PASCAL, 1565  
levelmeter\_t, 654  
    levelmeter\_t, 655  
    mode, 656  
    patchbay, 656  
    peak, 656  
    prepare, 655  
    process, 655

query\_peak, 656  
 query\_rms, 655  
 rms, 656  
 tau, 656  
 update\_tau, 655  
 levelmode  
     addsndfile::addsndfile\_if\_t, 253  
 Levinson2  
     lpc.cpp, 1566  
 lib\_data  
     io\_lib\_t, 611  
     PluginLoader::mhaplugloader\_t, 1370  
 lib\_err  
     io\_lib\_t, 610  
     PluginLoader::mhaplugloader\_t, 1370  
 lib\_handle  
     io\_lib\_t, 611  
     PluginLoader::mhaplugloader\_t, 1370  
 lib\_str\_error  
     io\_lib\_t, 610  
 libdata  
     analysepath\_t, 308  
     MHAParser::c\_ifc\_parser\_t, 1048  
 liberr  
     MHAParser::c\_ifc\_parser\_t, 1048  
 libname  
     analysispath\_if\_t, 312  
     PluginLoader::config\_file\_splitter\_t, 1360  
 library\_handle  
     matlab\_wrapper::matlab\_wrapper\_t::wrappedplugin\_t,  
         704  
 library\_name  
     matlab\_wrapper::matlab\_wrapper\_t, 698  
 library\_paths  
     pluginbrowser\_t, 1354  
 like\_ratio  
     acPooling\_wave\_config, 217  
 like\_ratio\_name  
     acPooling\_wave, 214  
 limit  
     coherence::cohfilt\_t, 351  
     coherence::vars\_t, 354  
     MHASignal, 144  
     MHASignal::quantizer\_t, 1236  
     MHASignal::waveform\_t, 1269  
 limiter  
     dc\_simple::dc\_t, 398  
 limiter\_threshold  
     dc\_simple::dc\_t, 398  
     dc\_simple::dc\_vars\_t, 402  
 lin2db  
     MHASignal, 139, 140  
 line\_t  
     dc\_simple::dc\_t::line\_t, 399  
 linear  
     MHAOvlFilter::ShapeFun, 121  
     softclipper\_t, 1458  
     softclipper\_variables\_t, 1460  
 linear\_table\_t  
     MHATableLookup::linear\_table\_t, 1276  
 Linearphase\_FIR  
     ADM::Linearphase\_FIR< F >, 269  
 list\_dir  
     mha\_os.cpp, 1603  
     mha\_os.h, 1608  
 lnn3rdoct\_t  
     audiometerbackend::lnn3rdoct\_t, 321  
 lo\_addr  
     ac2osc\_t, 185  
 load\_io\_lib  
     fw\_t, 524  
 load\_lib  
     dynamiclib\_t, 445  
     matlab\_wrapper::matlab\_wrapper\_t, 697  
 load\_plug  
     MHAParser::mhaplugloader\_t, 1089  
 load\_proc\_lib  
     fw\_t, 523  
 loader  
     latex\_doc\_t, 644  
     wrappedplugin\_t,  
         analysispath\_if\_t, 311  
 local\_address  
     io\_asterisk\_parser\_t, 591  
     io\_tcp\_parser\_t, 627  
 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, 591  
     io\_tcp\_parser\_t, 627  
 local\_remove\_ref  
     MHAKernel::algo\_comm\_class\_t, 988  
 local\_remove\_var  
     MHAKernel::algo\_comm\_class\_t, 988  
 locate  
     MHAIOJackdb::io\_jack\_t, 950

locate\_end  
    MHASignal::loop\_wavefragment\_t, 1220

lock\_channels  
    fw\_vars\_t, 528

lock\_srate\_fragsize  
    fw\_vars\_t, 528

locked  
    MHParse::variable\_t, 1116

log\_down  
    MHASignal::schroeder\_t, 1244

log\_interp  
    dc::dc\_t, 386  
    dc::dc\_vars\_t, 389

log\_lambda\_spec  
    smooth\_cepstrum::smooth\_cepstrum\_t,  
        1444

log\_up  
    MHASignal::schroeder\_t, 1243

logfile  
    mhaserver\_t, 1193

logGLRFact  
    noise\_psd\_estimator::noise\_psd\_estimator\_t,  
        1314

    smooth\_cepstrum::smooth\_cepstrum\_t,  
        1445

logstring  
    mhaserver\_t, 1192

longmsg  
    MHA\_Error, 762

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, 1322

low\_incl  
    MHParse::range\_var\_t, 1108

low\_limit  
    MHParse::range\_var\_t, 1108

low\_side\_flat  
    MHAoVlFilter::band\_descriptor\_t, 998

low\_thresh  
    acPooling\_wave\_config, 217

lower\_threshold  
    acPooling\_wave, 213

lowpass\_quotient  
    windnoise::if\_t, 1512

    windnoise::if\_t, 1513

LowPassCutOffFrequency  
    windnoise::if\_t, 1512

LowPassFraction  
    windnoise::cfg\_t, 1508  
    windnoise::if\_t, 1512

LowPassWindGain  
    windnoise::cfg\_t, 1508  
    windnoise::if\_t, 1512

lp  
    DynComp::dc\_afterburn\_rt\_t, 448  
    level\_matching::level\_matching\_config\_t,  
        649

lp1i  
    coherence::cohflt\_t, 352

lp1tg  
    coherence::cohflt\_t, 352

lp1r  
    coherence::cohflt\_t, 351

lp\_coeffs  
    adm\_rtconfig\_t, adm\_if\_t, 279

lp\_level\_tau  
    level\_matching::level\_matching\_t, 653

lp\_order  
    adm\_if\_t, 273

lp\_signal\_fpass  
    level\_matching::level\_matching\_t, 653

lp\_signal\_fstop  
    level\_matching::level\_matching\_t, 653

lp\_signal\_order  
    level\_matching::level\_matching\_t, 653

lpc, 657  
    ~lpc, 658  
    algo\_name, 659  
    comp\_each\_iter, 659  
    lpc, 658  
    lpc\_buffer\_size, 659  
    lpc\_order, 659  
    norm, 660  
    patchbay, 660  
    prepare, 658  
    process, 658  
    release, 659  
    shift, 659  
    update\_cfg, 659

lpc.cpp, 1565  
    INSERT\_PATCH, 1566  
    Levinson2, 1566  
    PATCH\_VAR, 1566

lpc.h, 1566

lpc\_bl\_predictor, 660  
   ~lpc\_bl\_predictor, 661  
   lpc\_bl\_predictor, 661  
 lpc\_order, 663  
 name\_b, 663  
 name\_f, 663  
 name\_kappa, 663  
 name\_lpc\_b, 663  
 name\_lpc\_f, 663  
 patchbay, 663  
 prepare, 662  
 process, 662  
 release, 662  
 update\_cfg, 662  
 lpc\_bl\_predictor.cpp, 1566  
   INSERT\_PATCH, 1567  
   PATCH\_VAR, 1567  
 lpc\_bl\_predictor.h, 1567  
   EPSILON, 1567  
 lpc\_bl\_predictor\_config, 664  
   ~lpc\_bl\_predictor\_config, 664  
   ac, 665  
   b\_est, 665  
   backward, 665  
   f\_est, 665  
   forward, 665  
   km, 666  
   lpc\_bl\_predictor\_config, 664  
   lpc\_order, 665  
   name\_b, 666  
   name\_f, 665  
   name\_km, 665  
   process, 664  
   s\_b, 666  
   s\_f, 666  
 lpc\_buffer\_size  
   lpc, 659  
   lpc\_config, 674  
 lpc\_burg-lattice.cpp, 1567  
   INSERT\_PATCH, 1568  
   PATCH\_VAR, 1568  
 lpc\_burg-lattice.h, 1568  
   EPSILON, 1568  
 lpc\_burglattice, 666  
   ~lpc\_burglattice, 667  
   lambda, 669  
   lpc\_burglattice, 667  
   lpc\_order, 669  
   name\_b, 669  
   name\_f, 669  
   name\_kappa, 669  
   patchbay, 669  
   prepare, 668  
   process, 668  
   release, 668  
   update\_cfg, 668  
 lpc\_burglattice\_config, 669  
   ~lpc\_burglattice\_config, 670  
   ac, 670  
   backward, 671  
   dm, 671  
   forward, 671  
   kappa, 671  
   kappa\_block, 671  
   lambda, 671  
   lpc\_burglattice\_config, 670  
   lpc\_order, 671  
   name\_b, 672  
   name\_f, 671  
   nm, 671  
   process, 670  
   s\_b, 672  
   s\_f, 672  
 lpc\_config, 672  
   ~lpc\_config, 673  
   A, 674  
   comp\_each\_iter, 674  
   comp\_iter, 674  
   corr\_out, 675  
   insert, 673  
   inwave, 674  
   lpc\_buffer\_size, 674  
   lpc\_config, 673  
   lpc\_out, 675  
   N, 674  
   norm, 673  
   order, 674  
   process, 673  
   R, 674  
   sample, 674  
   shift, 673  
 lpc\_order  
   adaptive\_feedback\_canceller, 243  
   lpc, 659  
   lpc\_bl\_predictor, 663  
   lpc\_bl\_predictor\_config, 665  
   lpc\_burglattice, 669  
   lpc\_burglattice\_config, 671  
 lpc\_out  
   lpc\_config, 675  
 Isl2ac, 96  
   Discard, 97

Ignore, 97  
overrun\_behavior, 97  
lsl2ac.cpp, 1568  
lsl2ac.hh, 1568  
lsl2ac::cfg\_t, 675  
  cfg\_t, 676  
  process, 676  
  varlist, 676  
lsl2ac::lsl2ac\_t, 677  
  activate, 680  
  available\_streams, 682  
  buffersize, 681  
  chunksize, 681  
  get\_all\_stream\_names, 680  
  lsl2ac\_t, 679  
  nchannels, 681  
  nsamples, 681  
  overrun\_behavior, 681  
  patchbay, 681  
  prepare, 679  
  process, 679  
  release, 679  
  setlock, 680  
  streams, 680  
  update, 680  
lsl2ac::save\_var\_t, 682  
  ~save\_var\_t, 684  
  ac, 687  
  buf, 686  
  bufsize, 689  
  chunksize, 688  
  cv, 687  
  get\_time\_correction, 686  
  info, 685  
  insert\_vars, 686  
  n\_new\_samples, 689  
  name, 688  
  nchannels, 688  
  new\_name, 687  
  nsamples, 688  
  ob, 688  
  operator=, 685  
  pull\_samples\_discard, 685  
  pull\_samples\_ignore, 685  
  receive\_frame, 685  
  save\_var\_t, 684  
  skip, 688  
  stream, 686  
  tc, 687  
  tc\_name, 687  
  tic, 688  
  ts, 687  
  ts\_buf, 686  
  ts\_name, 687  
lsl2ac\_t  
  lsl2ac::lsl2ac\_t, 679  
LTASS\_combined  
  speechnoise\_t, 1468  
LTASS\_female  
  speechnoise\_t, 1468  
LTASS\_male  
  speechnoise\_t, 1468  
ltgcomp  
  coherence::vars\_t, 355  
ltgtau  
  coherence::vars\_t, 355  
lval  
  MHAParser::expression\_t, 1057  
m  
  dc\_simple::dc\_t::line\_t, 400  
  matrixmixer::cfg\_t, 710  
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, 511  
m\_fmax  
  fshift::fshift\_t, 511  
m\_fmin  
  fshift::fshift\_t, 510  
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, 1636  
**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, 1408  
**main**  
 analysemhaplugin.cpp, 1531  
 browsemhaplugins.cpp, 1534  
 generatemhaplugindoc.cpp, 1553  
 mha.cpp, 1571  
 MHAPlugin\_Split::posix\_threads\_t, 1170  
 testalsadvice.c, 1706  
**make\_friendly\_number**  
 MHAFilter, 106  
 mhajack.cpp, 1684  
**make\_friendly\_number\_by\_limiting**  
 adaptive\_feedback\_canceller.cpp, 1526  
 nlms\_wave.cpp, 1691  
**mapping**  
 addsndfile::addsndfile\_if\_t, 253  
 coherence::vars\_t, 354  
**matlab\_wrapper**, 97  
**matlab\_wrapper.cpp**, 1569  
**matlab\_wrapper.hh**, 1569  
 MHAPLUGIN\_OVERLOAD\_OUTDOMAIN, 1569  
**matlab\_wrapper::callback**, 689  
 callback, 690  
 on\_writeaccess, 690  
 parent, 691  
 user\_config, 690  
 var, 691  
**matlab\_wrapper::matlab\_wrapper\_rt\_cfg\_t**, 691  
 ~matlab\_wrapper\_rt\_cfg\_t, 692  
**matlab\_wrapper\_rt\_cfg\_t**, 692  
 user\_config, 692  
**matlab\_wrapper::matlab\_wrapper\_t**, 693  
 callback, 698  
 callbacks, 698  
 cb\_patchbay, 698  
 insert\_config\_vars, 697  
 insert\_monitors, 697  
 library\_name, 698  
 load\_lib, 697  
 matlab\_wrapper\_t, 695  
 monitors, 699  
 patchbay, 698  
 plug, 698  
 prepare, 696  
 process, 695, 696  
 release, 697  
 update, 697  
 update\_monitors, 697  
 vars, 698  
**matlab\_wrapper::matlab\_wrapper\_t::wrapped\_plugin\_t**, 699  
 ~wrapped\_plugin\_t, 701  
 fcn\_init, 704  
 fcn\_prepare, 705  
 fcn\_process\_ss, 705  
 fcn\_process\_sw, 705  
 fcn\_process\_ws, 705  
 fcn\_process\_ww, 704  
 fcn\_release, 705  
 fcn\_terminate, 704  
 library\_handle, 704  
 mha\_spec\_out, 707  
 mha\_wave\_out, 706  
 prepare, 703  
 process\_ss, 702  
 process\_sw, 703  
 process\_ws, 702  
 process\_ww, 701  
 release, 703  
 signal\_dimensions, 706  
 spec\_in, 706  
 spec\_out, 706  
 state, 704  
 user\_config, 704  
 wave\_in, 706  
 wave\_out, 706  
 wrapped\_plugin\_t, 701  
**matlab\_wrapper::types< MHA\_SPECTRUM >**, 707  
 array\_type, 707  
 class\_signal\_type, 708  
 signal\_type, 707  
**matlab\_wrapper::types< MHA\_WAVEFORM >**

>, 708  
array, 708  
class\_signal\_type, 708  
signal\_type, 708  
matlab\_wrapper::types< T >, 707  
matlab\_wrapper\_rt\_cfg\_t  
matlab\_wrapper::matlab\_wrapper\_rt\_cfg\_t, MAX\_USER\_ERR  
692  
matlab\_wrapper\_t  
matlab\_wrapper::matlab\_wrapper\_t, 695  
matmix\_t  
matrixmixer::matmix\_t, 711  
matrix\_t  
MHASignal::matrix\_t, 1223, 1225  
matrixmixer, 98  
matrixmixer.cpp, 1570  
matrixmixer::cfg\_t, 709  
cfg\_t, 709  
m, 710  
process, 709  
sout, 710  
wout, 710  
matrixmixer::matmix\_t, 710  
ci, 712  
co, 712  
matmix\_t, 711  
mixer, 712  
patchbay, 712  
prepare, 711  
process, 711, 712  
update\_m, 712  
MAX  
mha\_defs.h, 1584  
max  
spec2wave.cpp, 1699  
Vector and matrix processing toolbox, 56  
max\_clipped  
softclipper\_variables\_t, 1460  
max\_frames  
acsave::cfg\_t, 224  
max\_lag  
doasvm\_feature\_extraction, 425  
max\_p\_ind\_name  
doasvm\_classification, 420  
max\_pool\_ind\_name  
acPooling\_wave, 214  
max\_q  
smooth\_cepstrum::smooth\_cepstrum\_t,  
1445  
max\_sleep\_time  
dropgen\_t, 435  
MAX\_TCP\_PORT  
MHAIOAsterisk.cpp, 1653  
MHAIOTCP.cpp, 1680  
MAX\_TCP\_PORT\_STR  
MHAIOAsterisk.cpp, 1653  
MHAIOTCP.cpp, 1680  
MAX\_USER\_ERR  
MHAIOSalsa.cpp, 1648  
MHAIOAsterisk.cpp, 1653  
MHAIOWfile.cpp, 1658  
MHAIOJack.cpp, 1662  
MHAIOJackdb.cpp, 1666  
MHAIOParser.cpp, 1670  
MHAIOPortAudio.cpp, 1674  
MHAIOTCP.cpp, 1679  
mhajack.h, 1686  
max\_val  
smooth\_cepstrum::smooth\_cepstrum\_t,  
1445  
maxabs  
Vector and matrix processing toolbox, 54,  
55  
maxframe  
acsave::save\_var\_t, 227  
maxgain  
dc\_simple::dc\_t, 398  
dc\_simple::dc\_vars\_t, 401  
DynComp::dc\_afterburn\_rt\_t, 448  
DynComp::dc\_afterburn\_vars\_t, 453  
maximum\_reader\_xruns\_in\_succession\_before\_stop  
mha\_drifter\_fifo\_t< T >, 759  
maximum\_writer\_xruns\_in\_succession\_before\_stop  
mha\_drifter\_fifo\_t< T >, 759  
maxInputChannels  
MHAIOPortAudio::device\_info\_t, 953  
maxlen  
plingploing::if\_t, 1339  
maxlen\_  
plingploing::plingploing\_t, 1342  
maxLim  
rohBeam::rohConfig, 1408  
maxOutputChannels  
MHAIOPortAudio::device\_info\_t, 953  
mcomplex\_mon\_t  
MHAParser::mcomplex\_mon\_t, 1076  
mcomplex\_t  
MHAParser::mcomplex\_t, 1078  
MConv  
mconv::MConv, 714  
mconv, 98  
mconv.cpp, 1570

mconv::MConv, 713  
     fragsize, 716  
     inch, 716  
     irs, 716  
     MConv, 714  
     nchannels\_in, 716  
     nchannels\_out, 716  
     outch, 716  
     patchbay, 716  
     prepare, 715  
     process, 715  
     release, 715  
     update, 715  
     update\_irs, 715

mean  
     MHA\_AC::stat\_t, 734  
     MHASignal, 147  
     MHASignal::stat\_t, 1251

mean\_std  
     MHASignal::stat\_t, 1251

median  
     MHASignal, 145

mfloat\_mon\_t  
     MHAParser::mfloat\_mon\_t, 1080

mfloat\_t  
     MHAParser::mfloat\_t, 1082

mha  
     mhaserver\_t::tcp\_server\_t, 1195  
     speechnoise\_t, 1468

mha.cpp, 1570  
     main, 1571  
     mhamain, 1570

mha.hh, 1571  
     algo\_comm\_t, 1577  
     MHA\_AC\_CHAR, 1576  
     MHA\_AC\_DOUBLE, 1576  
     MHA\_AC\_FLOAT, 1576  
     MHA\_AC\_INT, 1576  
     MHA\_AC\_MHACOMPLEX, 1576  
     MHA\_AC\_MHAREAL, 1576  
     MHA\_AC\_UNKNOWN, 1576  
     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, 1577  
     MHA\_DOMAIN\_UNKNOWN, 1575  
     MHA\_RELEASE\_VERSION\_STRING, 1575  
     MHA\_SPECTRUM, 1575

MHA\_STRF, 1574  
     MHA\_STRUCT\_SIZEMATCH, 1574  
     MHA\_VERSION, 1575  
     MHA\_VERSION\_BUILD, 1574  
     MHA\_VERSION\_MAJOR, 1574  
     MHA\_VERSION\_MINOR, 1574  
     MHA\_VERSION\_RELEASE, 1574  
     MHA\_VERSION\_STRING, 1575  
     MHA\_WAVEFORM, 1575  
     MHA\_XSTRF, 1574  
     MHADestroy\_t, 1577  
     MHAGetVersion\_t, 1577  
     MHAInit\_t, 1577  
     MHAPluginCategory\_t, 1578  
     MHAPluginDocumentation\_t, 1578  
     MHAPrepare\_t, 1577  
     MHAProc\_spec2spec\_t, 1578  
     MHAProc\_spec2wave\_t, 1578  
     MHAProc\_wave2spec\_t, 1578  
     MHAProc\_wave2wave\_t, 1578  
     MHARelease\_t, 1577  
     MHASet\_t, 1577  
     MHASetcpp\_t, 1577  
     MHAStrError\_t, 1578

MHA\_AC, 98  
     MHA\_AC::ac2matrix\_helper\_t, 717  
         ac, 718  
         ac2matrix\_helper\_t, 718  
         acvar, 719  
         getvar, 718  
         is\_complex, 718  
         name, 718  
         size, 718  
         username, 718

MHA\_AC::ac2matrix\_t, 719  
     ac2matrix\_t, 720  
     getname, 720  
     getusername, 720  
     insert, 721  
     update, 720

MHA\_AC::acspace2matrix\_t, 721  
     ~acspace2matrix\_t, 723  
     acspace2matrix\_t, 722  
     data, 725  
     frame, 724  
     frameno, 725  
     insert, 724  
     len, 725  
     operator=, 723  
     operator[], 723, 724  
     size, 724

update, 724  
MHA\_AC::double\_t, 725  
  ~double\_t, 726  
  ac, 727  
  data, 726  
  double\_t, 726  
  insert, 726  
  name, 727  
MHA\_AC::float\_t, 727  
  ~float\_t, 728  
  ac, 728  
  data, 728  
  float\_t, 728  
  insert, 728  
  name, 729  
MHA\_AC::int\_t, 729  
  ~int\_t, 730  
  ac, 730  
  data, 730  
  insert, 730  
  int\_t, 730  
  name, 730  
MHA\_AC::spectrum\_t, 731  
  ~spectrum\_t, 732  
  ac, 733  
  insert, 733  
  name, 733  
  spectrum\_t, 732  
MHA\_AC::stat\_t, 733  
  insert, 734  
  mean, 734  
  stat\_t, 734  
  std, 735  
  update, 734  
MHA\_AC::waveform\_t, 735  
  ~waveform\_t, 736  
  ac, 737  
  insert, 737  
  name, 737  
  waveform\_t, 736  
MHA\_AC\_CHAR  
  mha.hh, 1576  
MHA\_AC\_DOUBLE  
  mha.hh, 1576  
MHA\_AC\_FLOAT  
  mha.hh, 1576  
MHA\_AC\_INT  
  mha.hh, 1576  
MHA\_AC\_MHACOMPLEX  
  mha.hh, 1576  
MHA\_AC\_MHAREAL  
  mha.hh, 1576  
  mha.hh, 1576  
  MHA\_AC\_UNKNOWN  
    mha.hh, 1576  
  MHA\_AC\_USER  
    mha.hh, 1576  
  MHA\_AC\_VEC\_FLOAT  
    mha.hh, 1576  
  mha\_algo\_comm.cpp, 1579  
    AC\_DIM\_MISMATCH, 1580  
    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, 1580  
  mha\_algo\_comm.h, 1580  
  mha\_algo\_comm.hh, 1581  
    algo\_comm\_default, 1582  
    ALGO\_COMM\_ID\_STR, 1582  
mha\_alloc  
  mha\_ruby.cpp, 1624  
MHA\_assert  
  Error handling in the openMHA, 29  
MHA\_assert\_equal  
  Error handling in the openMHA, 29  
mha\_audio\_descriptor\_t, 737  
  cf, 738  
  chdir, 739  
  dt, 738  
  is\_complex, 738  
  n\_channels, 738  
  n\_freqs, 738  
  n\_samples, 738  
mha\_audio\_t, 739  
  cdata, 740  
  descriptor, 739  
  rdata, 740  
MHA\_CALLBACK\_TEST  
  mha.hh, 1573  
MHA\_CALLBACK\_TEST\_PREFIX  
  mha.hh, 1573  
mha\_channel\_info\_t, 740  
  dir, 741  
  id, 741  
  idstr, 741  
  peaklevel, 741  
  side, 741  
mha\_complex  
  Complex arithmetics in the openMHA, 60  
  mha\_complex\_t, 741

im, 742  
 re, 742  
**mha\_complex\_test\_array\_t**, 742  
 c, 743  
**mha\_dbdbuf\_t**  
 mha\_dbdbuf\_t< FIFO >, 745  
**mha\_dbdbuf\_t< FIFO >**, 743  
 ~mha\_dbdbuf\_t, 746  
 delay, 749  
 fifo\_size, 750  
 get\_delay, 746  
 get\_fifo\_size, 746  
 get\_inner\_error, 747  
 get\_inner\_size, 746  
 get\_input\_channels, 746  
 get\_input\_fifo\_fill\_count, 747  
 get\_input\_fifo\_space, 747  
 get\_outer\_size, 746  
 get\_output\_channels, 747  
 get\_output\_fifo\_fill\_count, 747  
 get\_output\_fifo\_space, 747  
 inner\_error, 750  
 inner\_size, 749  
 input, 748  
 input\_channels, 750  
 input\_fifo, 750  
 mha\_dbdbuf\_t, 745  
 outer\_error, 751  
 outer\_size, 749  
 output, 749  
 output\_channels, 750  
 output\_fifo, 750  
 process, 748  
 provoke\_inner\_error, 747  
 provoke\_outer\_error, 748  
 value\_type, 745  
**mha\_debug**  
 Error handling in the openMHA, 29  
 mha\_error.cpp, 1588  
**mha\_defs.h**, 1582  
 \_\_MHA\_FUN\_\_, 1583  
 \_\_declspec, 1583  
 CHECK\_EXPR, 1583  
 CHECK\_VAR, 1583  
 M\_PI, 1583  
 MAX, 1584  
 MHA\_EAR\_LEFT, 1584  
 MHA\_EAR\_MAX, 1584  
 MHA\_EAR\_RIGHT, 1584  
 MIN, 1583  
**mha\_dleenv**  
 mha\_os.cpp, 1603  
 mha\_os.h, 1608  
**mha\_direction\_t**, 751  
 azimuth, 751  
 distance, 752  
 elevation, 752  
**MHA\_DOMAIN\_MAX**  
 mha.hh, 1575  
**mha\_domain\_t**  
 mha.hh, 1577  
**MHA\_DOMAIN\_UNKNOWN**  
 mha.hh, 1575  
**mha\_drifter\_fifo\_t**  
 mha\_drifter\_fifo\_t< T >, 754  
**mha\_drifter\_fifo\_t< T >**, 752  
 desired\_fill\_count, 758  
 get\_available\_space, 756  
 get\_des\_fill\_count, 756  
 get\_fill\_count, 756  
 get\_min\_fill\_count, 757  
 maximum\_reader\_xruns\_in\_succession\_before\_stop, 759  
 maximum\_writer\_xruns\_in\_succession\_before\_stop, 759  
**mha\_drifter\_fifo\_t**, 754  
 minimum\_fill\_count, 757  
 null\_data, 759  
 read, 755  
 reader\_started, 758  
 reader\_xruns\_in\_succession, 759  
 reader\_xruns\_since\_start, 759  
 reader\_xruns\_total, 758  
 starting, 757  
 startup\_zeros, 760  
 stop, 757  
 write, 755  
 writer\_started, 758  
 writer\_xruns\_in\_succession, 759  
 writer\_xruns\_since\_start, 758  
 writer\_xruns\_total, 758  
**MHA\_EAR\_LEFT**  
 mha\_defs.h, 1584  
**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, 1586  
MHA\_ERR\_UNKNOWN  
    mha\_errno.h, 1586  
MHA\_ERR\_USER  
    mha\_errno.h, 1587  
MHA\_ERR\_VARFMT  
    mha\_errno.h, 1587  
MHA\_ERR\_VARRANGE  
    mha\_errno.h, 1586  
mha\_errno.c, 1584  
    cstr\_strerror, 1585  
    mha\_set\_user\_error, 1585  
    mha\_strerror, 1585  
    next\_except\_str, 1585  
    STRLEN, 1585  
mha\_errno.h, 1586  
    MHA\_ERR\_INVALID\_HANDLE, 1586  
    MHA\_ERR\_NULL, 1586  
    MHA\_ERR\_SUCCESS, 1586  
    MHA\_ERR\_UNKNOWN, 1586  
    MHA\_ERR\_USER, 1587  
    MHA\_ERR\_VARFMT, 1587  
    MHA\_ERR\_VARRANGE, 1586  
    mha\_set\_user\_error, 1587  
    mha\_strerror, 1587  
MHA\_Error, 760  
    ~MHA\_Error, 761  
    get\_longmsg, 762  
    get\_msg, 762  
    longmsg, 762  
    MHA\_Error, 761  
    msg, 762  
    operator=, 762  
    what, 762  
mha\_error.cpp, 1587  
    mha\_debug, 1588  
mha\_error.hh, 1588  
    Getmsg, 1589  
mha\_error\_helpers, 99  
    digits, 99  
    snprintf\_required\_length, 100  
MHA\_ErrorMsg  
    Error handling in the openMHA, 28  
MHA\_ErrorMsg2  
    MHAIOAsterisk.cpp, 1653  
    MHAIOTCP.cpp, 1679  
MHA\_ErrorMsg3  
    MHAIOAsterisk.cpp, 1653  
    MHAIOTCP.cpp, 1680  
mha\_event\_emitter.h, 1589  
mha\_events.cpp, 1589  
    mha\_events.h, 1589  
    mha\_exit\_request  
        mha\_ruby.cpp, 1624  
    mha\_fft  
        gsc\_adaptive\_stage::gsc\_adaptive\_stage,  
            537  
        smooth\_cepstrum::smooth\_cepstrum\_t,  
            1441  
    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, 1590  
    BARKSCALE\_ENTRIES, 1591  
    filtershapefun, 1591  
mha\_fftfb.hh, 1592  
mha\_fifo.cpp, 1593  
mha\_fifo.h, 1593  
    mha\_fifo\_thread\_platform\_implementation\_t,  
        1594  
    mha\_fifo\_lf\_t  
        mha\_fifo\_lf\_t< T >, 764  
    mha\_fifo\_lf\_t< T >, 763  
        atomic\_read\_ptr, 766  
        atomic\_write\_ptr, 766  
        get\_available\_space, 765  
        get\_fill\_count, 765  
        mha\_fifo\_lf\_t, 764  
        read, 765  
        write, 764  
    mha\_fifo\_lw\_t  
        mha\_fifo\_lw\_t< T >, 767  
        mha\_fifo\_lw\_t< T >, 766

~mha\_fifo\_lw\_t, 767  
 error, 769  
 mha\_fifo\_lw\_t, 767  
 read, 768  
 set\_error, 769  
 sync, 769  
 write, 768  
 mha\_fifo\_posix\_threads\_t, 770  
   ~mha\_fifo\_posix\_threads\_t, 770  
   aquire\_mutex, 771  
   decrease\_condition, 772  
   decrement, 771  
   increase\_condition, 772  
   increment, 771  
   mha\_fifo\_posix\_threads\_t, 770  
   mutex, 772  
   release\_mutex, 771  
   wait\_for\_decrease, 771  
   wait\_for\_increase, 771  
 mha\_fifo\_t  
   mha\_fifo\_t< T >, 774, 775  
 mha\_fifo\_t< T >, 772  
   ~mha\_fifo\_t, 775  
   buf, 778  
   clear, 777  
   get\_available\_space, 776  
   get\_fill\_count, 776, 778  
   get\_max\_fill\_count, 777  
   get\_read\_ptr, 778  
   get\_write\_ptr, 777  
   mha\_fifo\_t, 774, 775  
   operator=, 777  
   read, 776  
   read\_ptr, 779  
   value\_type, 774  
   write, 775  
   write\_ptr, 778  
 mha\_fifo\_thread\_guard\_t, 779  
   ~mha\_fifo\_thread\_guard\_t, 780  
   mha\_fifo\_thread\_guard\_t, 779  
   sync, 780  
 mha\_fifo\_thread\_platform\_implementation\_t  
   mha\_fifo.h, 1594  
 mha\_fifo\_thread\_platform\_t, 780  
   ~mha\_fifo\_thread\_platform\_t, 781  
   aquire\_mutex, 782  
   decrement, 783  
   increment, 783  
   mha\_fifo\_thread\_platform\_t, 781, 782  
   operator=, 783  
   release\_mutex, 782  
 wait\_for\_decrease, 782  
 wait\_for\_increase, 783  
 mha\_filter.cpp, 1594  
   diff\_coeffs, 1594  
 mha\_filter.hh, 1594  
 mha\_fragsize  
   MHAIOJackdb::io\_jack\_t, 948  
 mha\_free  
   mha\_ruby.cpp, 1623  
 mha\_freelib  
   mha\_os.h, 1605  
 mha\_freelib\_success  
   mha\_os.h, 1605  
 mha\_generic\_chain.cpp, 1596  
   mhaconfig\_compare, 1596  
 mha\_generic\_chain.h, 1596  
   MHAPLUGIN\_OVERLOAD\_OUTDOMAIN, 1597  
 mha\_getenv  
   mha\_os.cpp, 1602  
   mha\_os.h, 1607  
 mha\_getlibfun  
   mha\_os.h, 1605  
 mha\_getlibfun\_checked  
   mha\_os.h, 1605  
 mha\_git\_commit\_hash  
   mha\_git\_commit\_hash.cpp, 1597  
   mha\_git\_commit\_hash.hh, 1598  
 mha\_git\_commit\_hash.cpp, 1597  
   GITCOMMITHASH, 1597  
   mha\_git\_commit\_hash, 1597  
 mha\_git\_commit\_hash.hh, 1598  
   mha\_git\_commit\_hash, 1598  
 mha\_hasenv  
   mha\_os.cpp, 1602  
   mha\_os.h, 1607  
 mha\_hton  
   mha\_os.h, 1608, 1609  
 MHA\_ID\_MATRIX  
   mha\_signal.cpp, 1627  
 MHA\_ID\_UINT\_VECTOR  
   mha\_signal.cpp, 1626  
 mha\_io\_ifc.h, 1598  
   IODestroy\_t, 1600  
   IOInit\_t, 1600  
   IOPrepare\_t, 1600  
   IOProcessEvent\_t, 1599  
   IORelease\_t, 1600  
   IOSetVar\_t, 1600  
   IOStart\_t, 1600  
   IOStartedEvent\_t, 1599

IOStop\_t, 1600  
IOStoppedEvent\_t, 1599  
IOStrError\_t, 1600  
MHAIO\_DOCUMENTATION, 1599  
MHAIO\_DOCUMENTATION\_PREFIX,  
    1599  
mha\_io\_utils.cpp, 1601  
mha\_io\_utils.hh, 1601  
mha\_lib\_extension  
    mha\_os.h, 1606  
mha\_libhandle\_t  
    mha\_os.h, 1606  
mha\_library\_paths  
    mha\_os.cpp, 1603  
    mha\_os.h, 1608  
mha\_loadlib  
    mha\_os.h, 1605  
mha\_loadlib\_error  
    mha\_os.h, 1606  
mha\_min\_1  
    mha\_signal.hh, 1637  
mha\_msleep  
    mha\_os.h, 1606  
mha\_multisrc.cpp, 1601  
mha\_multisrc.h, 1601  
mha\_ntoh  
    mha\_os.h, 1608, 1609  
mha\_os.cpp, 1602  
    list\_dir, 1603  
    mha\_delenv, 1603  
    mha\_getenv, 1602  
    mha\_hasenv, 1602  
    mha\_library\_paths, 1603  
    mha\_setenv, 1603  
mha\_os.h, 1604  
    FMTsz, 1606  
    list\_dir, 1608  
    mha\_delenv, 1608  
    mha\_freelib, 1605  
    mha\_freelib\_success, 1605  
    mha\_getenv, 1607  
    mha\_getlibfun, 1605  
    mha\_getlibfun\_checked, 1605  
    mha\_hasenv, 1607  
    mha\_hton, 1608, 1609  
    mha\_lib\_extension, 1606  
    mha\_libhandle\_t, 1606  
    mha\_library\_paths, 1608  
    mha\_loadlib, 1605  
    mha\_loadlib\_error, 1606  
    mha\_msleep, 1606  
mha\_ntoh, 1608, 1609  
MHA\_RESOLVE, 1606  
MHA\_RESOLVE\_CHECKED, 1606  
mha\_setenv, 1607  
mha\_parse  
    mha\_ruby.cpp, 1624  
mha\_parser.cpp, 1609  
    check\_parenthesis\_complex, 1611  
    check\_sign\_complex, 1611  
    MHAPLATFORM, 1610  
    parse\_1\_complex, 1612  
    parse\_1\_float, 1610  
    write\_float, 1610  
mha\_parser.hh, 1613  
    DEFAULT\_RETURNSIZE, 1617  
    insert\_member, 1617  
mha\_platform\_tic  
    mha\_profiling.c, 1622  
    mha\_profiling.h, 1623  
mha\_platform\_tictoc\_t  
    mha\_profiling.h, 1622  
mha\_platform\_toc  
    mha\_profiling.c, 1622  
    mha\_profiling.h, 1623  
mha\_plugin.cpp, 1617  
mha\_plugin.hh, 1617  
    \_\_declspec, 1618  
HINSTANCE, 1619  
MHAPLUGIN\_CALLBACKS, 1620  
MHAPLUGIN\_CALLBACKS\_PREFIX,  
    1619  
MHAPLUGIN\_DOCUMENTATION, 1621  
MHAPLUGIN\_DOCUMENTATION\_PREFIX,  
    1620  
MHAPLUGIN\_INIT\_CALLBACKS, 1620  
MHAPLUGIN\_INIT\_CALLBACKS\_PREFIX,  
    1619  
MHAPLUGIN\_PROC\_CALLBACK, 1620  
MHAPLUGIN\_PROC\_CALLBACK\_PREFIX,  
    1619  
MHAPLUGIN\_SETCPP\_CALLBACK\_PREFIX,  
    1619  
WINAPI, 1618  
mha\_profiling.c, 1621  
    mha\_platform\_tic, 1622  
    mha\_platform\_toc, 1622  
    mha\_tic, 1621  
    mha\_toc, 1622  
mha\_profiling.h, 1622  
    mha\_platform\_tic, 1623  
    mha\_platform\_tictoc\_t, 1622

mha\_platform\_toc, 1623  
 mha\_real\_t  
     Vector and matrix processing toolbox, 37  
 mha\_real\_test\_array\_t, 784  
     r, 784  
 MHA\_RELEASE\_VERSION\_STRING  
     mha.hh, 1575  
 MHA\_RESOLVE  
     mha\_os.h, 1606  
 MHA\_RESOLVE\_CHECKED  
     mha\_os.h, 1606  
 mha\_round  
     mha\_signal.hh, 1636  
 mha\_rt\_fifo\_element\_t  
     mha\_rt\_fifo\_element\_t< T >, 785  
 mha\_rt\_fifo\_element\_t< T >, 784  
     ~mha\_rt\_fifo\_element\_t, 785  
     abandonned, 786  
     data, 786  
     mha\_rt\_fifo\_element\_t, 785  
     next, 785  
 mha\_rt\_fifo\_t  
     mha\_rt\_fifo\_t< T >, 787  
 mha\_rt\_fifo\_t< T >, 786  
     ~mha\_rt\_fifo\_t, 787  
     current, 789  
     mha\_rt\_fifo\_t, 787  
     poll, 788  
     poll\_1, 788  
     push, 788  
     remove\_abandonned, 789  
     remove\_all, 789  
     root, 789  
 mha\_ruby.cpp, 1623  
     Init\_mha\_ruby, 1624  
     mha\_alloc, 1624  
     mha\_exit\_request, 1624  
     mha\_free, 1623  
     mha\_parse, 1624  
     rb\_f\_t, 1623  
 mha\_samplerate  
     MHAIOJackdb::io\_jack\_t, 948  
 mha\_set\_user\_error  
     mha\_errno.c, 1585  
     mha\_errno.h, 1587  
 mha\_setenv  
     mha\_os.cpp, 1603  
     mha\_os.h, 1607  
 mha\_signal.cpp, 1624  
     ASSERT\_EQUAL\_DIM, 1627  
     ASSERT\_EQUAL\_DIM\_PTR, 1627  
     intensity, 1627  
     MHA\_ID\_MATRIX, 1627  
     MHA\_ID\_UINT\_VECTOR, 1626  
     safe\_div, 1627  
     set\_minabs, 1627  
 mha\_signal.hh, 1628  
     M\_PI, 1636  
     mha\_min\_1, 1637  
     mha\_round, 1636  
     operator<<, 1637  
     operator>>, 1638  
     safe\_div, 1637  
     set\_minabs, 1637  
     value, 1637  
 mha\_signal\_fft.h, 1638  
 mha\_spec\_out  
     matlab\_wrapper::matlab\_wrapper\_t::wrapped\_plugin\_t,  
         707  
 mha\_spec\_t, 790  
     buf, 791  
     channel\_info, 791  
     num\_channels, 791  
     num\_frames, 791  
 MHA\_SPECTRUM  
     mha.hh, 1575  
 mha\_stash\_environment\_variable\_t, 792  
     ~mha\_stash\_environment\_variable\_t,  
         793  
     existed\_before, 793  
     mha\_stash\_environment\_variable\_t, 792  
     original\_content, 793  
     variable\_name, 793  
 mha\_strerror  
     mha\_errno.c, 1585  
     mha\_errno.h, 1587  
 MHA\_STRF  
     mha.hh, 1574  
 MHA\_STRUCT\_SIZEMATCH  
     mha.hh, 1574  
 mha\_tablelookup.cpp, 1638  
 mha\_tablelookup.hh, 1638  
 MHA\_TCP, 100  
     dtime, 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, 1639  
    ASYNC\_CONNECT\_STARTED, 1640  
    closesocket, 1640  
    host\_port\_to\_sock\_addr, 1641  
    INVALID\_SOCKET, 1640  
    SOCKET, 1641  
    SOCKET\_ERROR, 1640  
    tcp\_connect\_to, 1641  
    tcp\_connect\_to\_with\_timeout, 1641  
    thread\_start\_func, 1641  
mha\_tcp.hh, 1642  
    Sleep, 1643  
MHA\_TCP::Async\_Notify, 794  
    ~Async\_Notify, 795  
    Async\_Notify, 794  
    pipe, 795  
    reset, 795  
    set, 795  
mha\_tcp::buffered\_socket\_t, 795  
    current\_message, 797  
    get\_buffer, 796  
    next\_message, 797  
    queue\_write, 796  
    streambuf, 797  
MHA\_TCP::Client, 797  
    Client, 798  
MHA\_TCP::Connection, 799  
    ~Connection, 801  
    buffered\_incoming\_bytes, 806  
    buffered\_outgoing\_bytes, 806  
    can\_read\_bytes, 804  
    can\_read\_line, 803  
    can\_sysread, 802  
    can\_syswrite, 802  
    closed, 807  
    Connection, 801  
    eof, 803  
    fd, 807  
    get\_fd, 803  
    get\_peer\_address, 803  
    get\_peer\_port, 803  
    get\_read\_event, 803  
    get\_write\_event, 803  
    inbuf, 806  
    init\_peer\_data, 801  
    needs\_write, 806  
    outbuf, 806  
    peer\_addr, 807  
    read\_bytes, 805  
    read\_event, 806  
        read\_line, 804  
        sysread, 802  
        syswrite, 802  
        try\_write, 805  
        write, 805  
        write\_event, 806  
MHA\_TCP::Event\_Watcher, 807  
    ~Event\_Watcher, 809  
    Events, 808  
    events, 809  
    ignore, 809  
    iterator, 808  
    observe, 809  
    wait, 809  
MHA\_TCP::OS\_EVENT\_TYPE, 810  
    fd, 811  
    mode, 811  
    R, 810  
    T, 810  
    timeout, 811  
    W, 810  
    X, 810  
MHA\_TCP::Server, 811  
    ~Server, 812  
    accept, 813  
    accept\_event, 814  
    get\_accept\_event, 813  
    get\_interface, 813  
    get\_port, 813  
    iface, 814  
    initialize, 813  
    port, 814  
    Server, 812  
    serversocket, 814  
    sock\_addr, 814  
    try\_accept, 813  
mha\_tcp::server\_t, 815  
    ~server\_t, 817  
    acceptor, 820  
    add\_connection, 820  
    async\_accept\_has\_been\_triggered, 820  
    connections, 820  
    get\_address, 817  
    get\_context, 819  
    get\_endpoint, 817  
    get\_num\_accepted\_connections, 818  
    get\_port, 817  
    io\_context, 820  
    num\_accepted\_connections, 820  
    on\_received\_line, 818  
    post\_trigger\_read\_line, 819

run, 818  
 server\_t, 816  
 shutdown, 819  
 trigger\_accept, 819  
 trigger\_read\_line, 819  
 MHA\_TCP::sock\_init\_t, 821  
 sock\_init\_t, 821  
 MHA\_TCP::Sockaccept\_Event, 821  
 Sockaccept\_Event, 822  
 MHA\_TCP::Sockread\_Event, 822  
 Sockread\_Event, 823  
 MHA\_TCP::Sockwrite\_Event, 823  
 Sockwrite\_Event, 824  
 MHA\_TCP::Thread, 824  
 ~Thread, 827  
 arg, 827  
 error, 828  
 FINISHED, 826  
 PREPARED, 826  
 return\_value, 828  
 run, 827  
 RUNNING, 826  
 state, 828  
 thr\_f, 825  
 Thread, 826  
 thread\_arg, 828  
 thread\_attr, 827  
 thread\_finish\_event, 828  
 thread\_func, 828  
 thread\_handle, 827  
 MHA\_TCP::Timeout\_Event, 829  
 end\_time, 830  
 get\_os\_event, 829  
 Timeout\_Event, 829  
 MHA\_TCP::Timeout\_Watcher, 830  
 ~Timeout\_Watcher, 831  
 timeout, 831  
 Timeout\_Watcher, 831  
 MHA\_TCP::Wakeup\_Event, 832  
 ~Wakeup\_Event, 833  
 get\_os\_event, 834  
 ignored\_by, 833  
 observed\_by, 833  
 observers, 834  
 os\_event, 834  
 os\_event\_valid, 835  
 reset, 834  
 status, 834  
 Wakeup\_Event, 833  
 mha\_tcp\_server.cpp, 1643  
 mha\_tcp\_server.hh, 1643  
 mha\_test\_struct\_size  
     PluginLoader::mhaplugloader\_t, 1369  
 mha\_tic  
     mha\_profiling.c, 1621  
 mha\_tictoc\_t, 835  
     t, 835  
     tv1, 835  
     tv2, 835  
     tz, 835  
 mha\_toc  
     mha\_profiling.c, 1622  
 mha\_toolbox.h, 1644  
 mha\_utils.cpp, 1644  
 mha\_utils.hh, 1644  
 MHA\_VERSION  
     mha.hh, 1575  
 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, 1575  
 mha\_wave\_out  
     matlab\_wrapper::matlab\_wrapper\_t::wrapped\_plugin\_t,  
         706  
 mha\_wave\_t, 836  
     buf, 837  
     channel\_info, 837  
     num\_channels, 837  
     num\_frames, 837  
 MHA\_WAVEFORM  
     mha.hh, 1575  
 mha\_windowparser.cpp, 1644  
     wnd\_funcs, 1645  
 mha\_windowparser.h, 1645  
 MHA\_XSTRF  
     mha.hh, 1574  
 mhachain, 104  
 mhachain.cpp, 1646  
 mhachain::chain\_base\_t, 838  
     algos, 840  
     b\_prepared, 841  
     bprofiling, 840  
     cfin, 841  
     cfout, 841  
     chain\_base\_t, 839  
     old\_algos, 840

patchbay, 841  
prepare, 840  
process, 839, 840  
release, 840  
update, 840  
mhachain::mhachain\_t, 842  
    mhachain\_t, 842  
mhachain::plugs\_t, 843  
    ~plugs\_t, 844  
    ac, 845  
    algos, 845  
    alloc\_plugs, 844  
    b\_prepared, 845  
    b\_use\_profiling, 847  
    cleanup\_plugs, 845  
    parser, 845  
    plugs\_t, 843  
    prepare, 844  
    prepared, 844  
    proc\_cnt, 846  
    process, 844  
    prof\_algos, 845  
    prof\_cfg, 846  
    prof\_init, 846  
    prof\_load\_con, 847  
    prof\_prepare, 846  
    prof\_process, 846  
    prof\_process\_load, 846  
    prof\_process\_tt, 846  
    prof\_release, 846  
    prof\_tt\_con, 847  
    profiling, 845  
    release, 844  
    tictoc, 847  
    update\_proc\_load, 845  
mhachain\_t  
    mhachain::mhachain\_t, 842  
mhachannels  
    addsndfile::addsndfile\_if\_t, 254  
mhaconfig\_compare  
    mha\_generic\_chain.cpp, 1596  
    PluginLoader, 157  
mhaconfig\_in  
    MHAPlugin::plugin\_t< runtime\_cfg\_t >, 1153  
mhaconfig\_mon\_t  
    MHAParser::mhaconfig\_mon\_t, 1085  
mhaconfig\_out  
    MHAPlugin::plugin\_t< runtime\_cfg\_t >, 1153  
mhaconfig\_t, 847  
channels, 848  
domain, 848  
ffflen, 849  
fragsize, 848  
srate, 849  
wndlen, 849  
MHADestroy\_cb  
    PluginLoader::mhapluginloader\_t, 1370  
MHADestroy\_t  
    mha.hh, 1577  
MHAEvents, 104  
MHAEvents::connector\_base\_t, 849  
    ~connector\_base\_t, 850  
    connector\_base\_t, 850  
    emit\_event, 850, 851  
    emitter\_die, 851  
    emitter\_is\_alive, 851  
MHAEvents::connector\_t< receiver\_t >, 852  
    ~connector\_t, 853  
    connector\_t, 853  
    emit\_event, 853, 854  
    emitter, 854  
    eventhandler, 854  
    eventhandler\_s, 854  
    eventhandler\_suu, 855  
    receiver, 854  
MHAEvents::emitter\_t, 855  
    ~emitter\_t, 856  
    connect, 856  
    connections, 857  
    disconnect, 856  
    operator(), 856  
MHAEvents::patchbay\_t< receiver\_t >, 857  
    ~patchbay\_t, 858  
    connect, 858, 859  
    cons, 859  
mhafft  
    fshift\_hilbert::hilbert\_shifter\_t, 517  
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, 859  
    adapt\_filter\_param\_t, 860  
    err\_in, 860  
    mu, 860

MHAFilter::adapt\_filter\_state\_t, 860  
 adapt\_filter\_state\_t, 861  
 filter, 861  
 nchannels, 861  
 ntaps, 861  
 od, 862  
 oy, 862  
 W, 861  
 X, 861

MHAFilter::adapt\_filter\_t, 862  
 adapt\_filter\_t, 863  
 connector, 864  
 err\_in, 864  
 filter, 863  
 mu, 864  
 nchannels, 864  
 ntaps, 864  
 set\_channelcnt, 863  
 update\_mu, 863  
 update\_ntaps, 864

MHAFilter::blockprocessing\_polyphase\_resampling\_get\_irs, 881  
 864  
 ~blockprocessing\_polyphase\_resampling\_t, 866  
 blockprocessing\_polyphase\_resampling\_t, 865  
 can\_read, 867  
 fragsize\_in, 867  
 fragsize\_out, 867  
 num\_channels, 868  
 read, 867  
 resampling, 867  
 write, 866

MHAFilter::complex\_bandpass\_t, 868  
 A\_, 871  
 B\_, 872  
 complex\_bandpass\_t, 869  
 creator\_A, 869  
 creator\_B, 870  
 filter, 870, 871  
 get\_weights, 870  
 inspect, 871  
 set\_state, 870  
 set\_weights, 870  
 Yn, 872

MHAFilter::diff\_t, 872  
 diff\_t, 873

MHAFilter::fftfilter\_t, 873  
 ~fftfilter\_t, 874  
 channels, 877  
 fft, 878  
 fftfilter\_t, 874  
 fftlen, 877  
 filter, 875, 876  
 fragsize, 876  
 sInput, 877  
 sWeights, 877  
 update\_coeffs, 875  
 wInput, 877  
 wInput\_fft, 877  
 wIRS\_fft, 877  
 wOutput, 877  
 wOutput\_fft, 877

MHAFilter::fftfilterbank\_t, 878  
 ~fftfilterbank\_t, 880  
 fft, 883  
 fftfilterbank\_t, 879  
 fftlen, 882  
 filter, 880, 881  
 firchannels, 882  
 fragsize, 882  
 Hs, 882  
 hw, 882  
 inputchannels, 882  
 outputchannels, 882  
 tail, 883  
 update\_coeffs, 880  
 Xs, 882  
 xw, 882  
 Ys, 883  
 yw, 883  
 yw\_temp, 883

MHAFilter::filter\_t, 883  
 ~filter\_t, 886  
 A, 887  
 B, 888  
 channels, 888  
 filter, 886, 887  
 filter\_t, 885  
 get\_len\_A, 887  
 get\_len\_B, 887  
 len, 888  
 len\_A, 888  
 len\_B, 888  
 operator=, 886  
 state, 888

MHAFilter::gamma\_flt\_t, 888  
 ~gamma\_flt\_t, 890  
 A, 892  
 bw\_, 892  
 cf\_, 892

delay, 892  
envelope\_delay, 892  
gamma\_flt\_t, 889  
get\_A, 891  
get\_resynthesis\_gain, 891  
get\_weights, 891  
GF, 892  
inspect, 891  
operator(), 890  
phase\_correction, 890  
reset\_state, 891  
resynthesis\_gain, 892  
set\_weights, 891  
srata\_, 892  
MHAFilter::iir\_filter\_state\_t, 893  
    iir\_filter\_state\_t, 893  
MHAFilter::iir\_filter\_t, 894  
    A, 898  
    B, 898  
    connector, 898  
    filter, 896  
    iir\_filter\_t, 895  
    nchannels, 898  
    resize, 896  
    update\_filter, 898  
MHAFilter::iir\_ord1\_real\_t, 898  
    A, 901  
    B, 901  
    iir\_ord1\_real\_t, 899  
    operator(), 900, 901  
    set\_state, 900  
    Yn, 901  
MHAFilter::o1\_ar\_filter\_t, 902  
    c1\_a, 905  
    c1\_r, 905  
    c2\_a, 905  
    c2\_r, 905  
    fs, 905  
    o1\_ar\_filter\_t, 903  
    operator(), 904, 905  
    set\_tau\_attack, 904  
    set\_tau\_release, 904  
MHAFilter::o1flt\_lowpass\_t, 906  
    get\_c1, 908  
    get\_last\_output, 908  
    o1flt\_lowpass\_t, 907  
    set\_tau, 908  
MHAFilter::o1flt\_maxtrack\_t, 909  
    o1flt\_maxtrack\_t, 910  
    set\_tau, 910  
MHAFilter::o1flt\_mintrack\_t, 911  
    o1flt\_mintrack\_t, 912  
    set\_tau, 912  
MHAFilter::partitioned\_convolution\_t, 913  
    ~partitioned\_convolution\_t, 914  
    bookkeeping, 916  
    current\_input\_signal\_buffer\_half\_index, 916  
    current\_output\_partition\_index, 917  
    fft, 917  
    filter\_partitions, 915  
    fragsize, 915  
    frequency\_response, 916  
    input\_signal\_spec, 916  
    input\_signal\_wave, 916  
    nchannels\_in, 915  
    nchannels\_out, 915  
    output\_partitions, 915  
    output\_signal\_spec, 916  
    output\_signal\_wave, 917  
    partitioned\_convolution\_t, 914  
    process, 915  
MHAFilter::partitioned\_convolution\_t::index\_t, 917  
    delay, 919  
    index\_t, 918  
    source\_channel\_index, 919  
    target\_channel\_index, 919  
MHAFilter::polyphase\_resampling\_t, 919  
    downsampling\_factor, 923  
    impulse\_response, 923  
    now\_index, 923  
    polyphase\_resampling\_t, 921  
    read, 922  
    readable\_frames, 922  
    ringbuffer, 924  
    underflow, 923  
    upsampling\_factor, 923  
    write, 922  
MHAFilter::resampling\_filter\_t, 924  
    fragsize, 926  
    fragsize\_validator, 926  
    resampling\_filter\_t, 925  
MHAFilter::smoothspec\_t, 926  
    ~smoothspec\_t, 928  
    fft, 930  
    ffflen, 929  
    internal\_fir, 929  
    minphase, 930  
    nchannels, 929  
    smoothspec, 928

smoothspec\_t, 927  
 spec2fir, 929  
 tmp\_spec, 930  
 tmp\_wave, 930  
 window, 929  
**MHAFilter::thirdoctave\_analyzer\_t**, 930  
 bw\_generator, 932  
 cf, 932  
 cf\_generator, 932  
 cfg\_, 932  
 dup, 932  
 fb, 932  
 get\_cf\_hz, 931  
 nbands, 931  
 nchannels, 931  
 out\_chunk, 932  
 out\_chunk\_im, 933  
 process, 931  
 thirdoctave\_analyzer\_t, 931  
**MHAFilter::transfer\_function\_t**, 933  
 impulse\_response, 936  
 isempty, 935  
 non\_empty\_partitions, 935  
 partitions, 934  
 source\_channel\_index, 936  
 target\_channel\_index, 936  
 transfer\_function\_t, 934  
**MHAFilter::transfer\_matrix\_t**, 936  
 non\_empty\_partitions, 937  
 partitions, 937  
 mhafw\_lib.cpp, 1646  
 mhafw\_lib.h, 1646  
**MHAGetVersion\_cb**  
 PluginLoader::mhapluginloader\_t, 1370  
**MHAGetVersion\_t**  
 mha.hh, 1577  
**MHAInit\_cb**  
 PluginLoader::mhapluginloader\_t, 1370  
**MHAInit\_t**  
 mha.hh, 1577  
**MHAIO\_DOCUMENTATION**  
 mha\_io\_ifc.h, 1599  
**MHAIO\_DOCUMENTATION\_PREFIX**  
 mha\_io\_ifc.h, 1599  
**MHAIOalsa.cpp**, 1646  
 DBG, 1647  
 dummy\_interface\_test, 1649  
 ERR\_IHANDLE, 1648  
 ERR\_SUCCESS, 1648  
 ERR\_USER, 1648  
 IODestroy, 1649, 1651  
 IOInit, 1648, 1649  
 IOPrepare, 1648, 1650  
 IORelease, 1648, 1650  
 IOSetVar, 1649, 1650  
 IOStart, 1648, 1650  
 IOStop, 1648, 1650  
 IOStrError, 1649, 1650  
 MAX\_USER\_ERR, 1648  
 user\_err\_msg, 1651  
**MHAIOAsterisk.cpp**, 1651  
 copy\_error, 1655  
 dummy\_interface\_test, 1654  
 ERR\_IHANDLE, 1652  
 ERR\_SUCCESS, 1652  
 ERR\_USER, 1652  
 IODestroy, 1654, 1656  
 IOInit, 1653, 1655  
 IOPrepare, 1654, 1655  
 IORelease, 1654, 1655  
 IOSetVar, 1654, 1656  
 IOStart, 1654, 1655  
 IOStop, 1654, 1655  
 IOStrError, 1654, 1656  
 MAX\_TCP\_PORT, 1653  
 MAX\_TCP\_PORT\_STR, 1653  
 MAX\_USER\_ERR, 1653  
 MHA\_ErrorMsg2, 1653  
 MHA\_ErrorMsg3, 1653  
 MIN\_TCP\_PORT, 1653  
 MIN\_TCP\_PORT\_STR, 1653  
 thread\_startup\_function, 1655  
 user\_err\_msg, 1656  
**MHAIOFile.cpp**, 1656  
 DEBUG, 1657  
 dummy\_interface\_test, 1659  
 ERR\_IHANDLE, 1658  
 ERR\_SUCCESS, 1657  
 ERR\_USER, 1658  
 IODestroy, 1659, 1660  
 IOInit, 1658, 1659  
 IOPrepare, 1658, 1659  
 IORelease, 1658, 1660  
 IOSetVar, 1658, 1660  
 IOStart, 1658, 1659  
 IOStop, 1658, 1660  
 IOStrError, 1659, 1660  
 MAX\_USER\_ERR, 1658  
 user\_err\_msg, 1660  
**MHAIOJack**, 109  
**MHAIOJack.cpp**, 1661  
 dummy\_interface\_test, 1663

ERR\_IHANDLE, 1662  
ERR\_SUCCESS, 1662  
ERR\_USER, 1662  
IODestroy, 1663, 1664  
IOInit, 1662, 1663  
IOPrepare, 1662, 1663  
IOResume, 1663, 1664  
IOSetVar, 1663, 1664  
IOStart, 1662, 1664  
IOStop, 1662, 1664  
IOStrError, 1663, 1664  
MAX\_USER\_ERR, 1662  
user\_err\_msg, 1665  
MHAIOJack::io\_jack\_t, 937  
    clientname, 941  
    connections\_in, 941  
    connections\_out, 942  
    delays\_in, 941  
    delays\_out, 942  
    fw fragsize, 941  
    fw samplerate, 941  
    get\_all\_input\_ports, 940  
    get\_all\_output\_ports, 940  
    get\_delays\_in, 940  
    get\_delays\_out, 940  
    get\_physical\_input\_ports, 940  
    get\_physical\_output\_ports, 940  
    io\_jack\_t, 939  
    patchbay, 943  
    portnames\_in, 942  
    portnames\_out, 942  
    ports\_in\_all, 942  
    ports\_in\_physical, 942  
    ports\_out\_all, 942  
    ports\_out\_physical, 942  
    ports\_parser, 943  
    prepare, 939  
    read\_get\_cpu\_load, 940  
    read\_get\_scheduler, 941  
    read\_get\_xruns, 941  
    reconnect\_inports, 939  
    reconnect\_outports, 940  
    release, 939  
    servername, 941  
    state\_cpupload, 943  
    state\_parser, 943  
    state\_priority, 943  
    state\_scheduler, 943  
    state\_xruns, 943  
MHAIOJackdb, 110  
MHAIOJackdb.cpp, 1665  
dummy\_interface\_test, 1667  
ERR\_IHANDLE, 1666  
ERR\_SUCCESS, 1666  
ERR\_USER, 1666  
IODestroy, 1667, 1668  
IOInit, 1666, 1667  
IOPrepare, 1666, 1667  
IOResume, 1667, 1668  
IOSetVar, 1667, 1668  
IOStart, 1666, 1668  
IOStop, 1666, 1668  
IOStrError, 1667, 1668  
MAX\_USER\_ERR, 1666  
user\_err\_msg, 1669  
MHAIOJackdb::io\_jack\_t, 944  
    clientname, 949  
    connections\_in, 949  
    connections\_out, 949  
    fail\_on\_async\_jackerr, 949  
    fail\_on\_async\_jackerror, 946  
    fragsize\_ratio, 948  
    get\_all\_input\_ports, 947  
    get\_all\_output\_ports, 947  
    get\_physical\_input\_ports, 947  
    get\_physical\_output\_ports, 947  
    io\_jack\_t, 946  
    IOProcessEvent\_inner, 946  
    locate, 950  
    mha fragsize, 948  
    mha\_samplerate, 948  
    patchbay, 951  
    portnames\_in, 949  
    portnames\_out, 949  
    ports\_in\_all, 950  
    ports\_in\_physical, 950  
    ports\_out\_all, 950  
    ports\_out\_physical, 950  
    ports\_parser, 950  
    prepare, 946  
    proc\_event, 948  
    proc\_handle, 948  
    pwinner\_out, 951  
    read\_get\_cpu\_load, 947  
    read\_get\_scheduler, 948  
    read\_get\_xruns, 947  
    reconnect\_inports, 947  
    reconnect\_outports, 947  
    release, 946  
    server fragsize, 950  
    server\_srate, 950  
    servername, 949

set\_locate, 948  
 set\_use\_jack\_transport, 948  
 state\_cpupload, 951  
 state\_parser, 951  
 state\_priority, 951  
 state\_scheduler, 951  
 state\_xruns, 951  
 use\_jack\_transport, 949  
**MHAIOParser.cpp**, 1669  
 dummy\_interface\_test, 1671  
 ERR\_IHANDLE, 1670  
 ERR\_SUCCESS, 1670  
 ERR\_USER, 1670  
 IODestroy, 1671, 1672  
 IOInit, 1670, 1671  
 IOPrepare, 1670, 1671  
 IOResume, 1671, 1672  
 IOSetVar, 1671, 1672  
 IOStart, 1670, 1672  
 IOStop, 1670, 1672  
 IOStrError, 1671, 1672  
 MAX\_USER\_ERR, 1670  
 user\_err\_msg, 1673  
**MHAIOPortAudio**, 110  
 parserFriendlyName, 110  
**MHAIOPortAudio.cpp**, 1673  
 dummy\_interface\_test, 1675  
 ERR\_IHANDLE, 1674  
 ERR\_SUCCESS, 1674  
 ERR\_USER, 1674  
 IODestroy, 1675, 1677  
 IOInit, 1674, 1676  
 IOPrepare, 1675, 1676  
 IOResume, 1675, 1676  
 IOSetVar, 1675, 1677  
 IOStart, 1675, 1676  
 IOStop, 1675, 1676  
 IOStrError, 1675, 1677  
 MAX\_USER\_ERR, 1674  
 portaudio\_callback, 1676, 1677  
 user\_err\_msg, 1677  
**MHAIOPortAudio::device\_info\_t**, 952  
 defaultHighInputLatency, 954  
 defaultHighOutputLatency, 954  
 defaultLowInputLatency, 954  
 defaultLowOutputLatency, 954  
 defaultSampleRate, 954  
 device\_info\_t, 952  
 fill\_info, 953  
 hostApi, 953  
 maxInputChannels, 953  
 maxOutputChannels, 953  
 name, 953  
 numDevices, 953  
 structVersion, 953  
**MHAIOPortAudio::io\_portaudio\_t**, 955  
 ~io\_portaudio\_t, 956  
 cmd\_prepare, 957  
 cmd\_release, 958  
 cmd\_start, 957  
 cmd\_stop, 957  
 device\_index\_in, 960  
 device\_index\_in\_updated, 957  
 device\_index\_out, 960  
 device\_index\_out\_updated, 957  
 device\_info, 958  
 device\_name\_in, 960  
 device\_name\_in\_updated, 957  
 device\_name\_out, 960  
 device\_name\_out\_updated, 957  
 fragsize, 959  
 io\_portaudio\_t, 956  
 nchannels\_in, 959  
 nchannels\_out, 959  
 patchbay, 960  
 portaudio\_callback, 958  
 portaudio\_stream, 960  
 proc\_event, 959  
 proc\_handle, 959  
 s\_in, 958  
 s\_out, 958  
 samplerate, 958  
 start\_event, 959  
 start\_handle, 959  
 stop\_event, 959  
 stop\_handle, 959  
 stream\_info, 958  
 suggestedInputLatency, 960  
 suggestedOutputLatency, 960  
**MHAIOPortAudio::stream\_info\_t**, 961  
 fill\_info, 962  
 paInputLatency, 962  
 paOutputLatency, 962  
 paSampleRate, 962  
 stream\_info\_t, 961  
**MHAIOTCP.cpp**, 1678  
 copy\_error, 1681  
 dummy\_interface\_test, 1681  
 ERR\_IHANDLE, 1679  
 ERR\_SUCCESS, 1679  
 ERR\_USER, 1679  
 IODestroy, 1681, 1683

IOInit, 1680, 1682  
IOPrepare, 1680, 1682  
IOResume, 1681, 1682  
IOSetVar, 1681, 1682  
IOStart, 1680, 1682  
IOStop, 1681, 1682  
IOStrError, 1681, 1683  
MAX\_TCP\_PORT, 1680  
MAX\_TCP\_PORT\_STR, 1680  
MAX\_USER\_ERR, 1679  
MHA\_ErrorMsg2, 1679  
MHA\_ErrorMsg3, 1680  
MIN\_TCP\_PORT, 1680  
MIN\_TCP\_PORT\_STR, 1680  
thread\_startup\_function, 1681  
user\_err\_msg, 1683  
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, 1683  
    dummy\_jack\_proc\_cb, 1684  
    jack\_error\_handler, 1683  
    last\_jack\_err, 1684  
    last\_jack\_err\_msg, 1684  
    make\_friendly\_number, 1684  
mhajack.h, 1684  
    IO\_ERROR\_JACK, 1686  
    IO\_ERROR\_MHAJACKLIB, 1686  
    last\_jack\_err\_msg, 1686  
    MAX\_USER\_ERR, 1686  
    MHAJACK\_FW\_STARTED, 1685  
    MHAJACK\_STARTING, 1686  
    MHAJACK\_STOPPED, 1686  
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, 968, 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, 1685  
**MHAJACK\_STARTING**  
 mhajack.h, 1686  
**MHAJACK\_STOPPED**  
 mhajack.h, 1686  
**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, 1687  
 mhamain.cpp, 1686  
 GREETING\_TEXT, 1687  
 HELP\_TEXT, 1687  
 mhamain, 1687  
 NORELEASE\_WARNING, 1687  
 VERSION\_EXTENSION, 1687  
**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  
 funs, 1026  
 get\_fun, 1025  
 get\_name, 1025  
 hz2unit, 1025  
 names, 1026  
 scale\_var\_t, 1025  
 unit2hz, 1025

MHAOvlFilter::ShapeFun, 120  
 expflt, 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, 1031  
 activate\_query, 1037  
 add\_parent\_on\_insert, 1037  
 add\_replace\_pair, 1037  
 base\_t, 1030, 1031  
 data\_is\_initialized, 1039  
 fullname, 1037  
 help, 1039  
 id\_str, 1039  
 nested\_lock, 1039  
 notify, 1037  
 op\_query, 1033  
 op\_setval, 1033  
 op\_subparse, 1032  
 operator=, 1031  
 operators, 1039  
 oplist, 1037

parent, 1039  
 parse, 1031, 1032  
 prereadaccess, 1038  
 queries, 1038  
 query\_addsubst, 1036  
 query\_cmds, 1036  
 query\_dump, 1033  
 query\_entries, 1033  
 query\_help, 1036  
 query\_id, 1035  
 query\_listids, 1035  
 query\_perm, 1033  
 query\_range, 1034  
 query\_readfile, 1034  
 query\_savefile, 1035  
 query\_savefile\_compact, 1035  
 query\_savemons, 1035  
 query\_subst, 1036  
 query\_type, 1034  
 query\_val, 1034  
 query\_version, 1035  
 readaccess, 1038  
 repl\_list, 1039  
 repl\_list\_t, 1030  
 rm\_parent\_on\_remove, 1037  
 set\_help, 1036  
 set\_node\_id, 1036  
 thefullname, 1039  
 valuechanged, 1038  
 writeaccess, 1038

MHAParser::base\_t::replace\_t, 1040  
 a, 1041  
 b, 1041  
 get\_a, 1040  
 get\_b, 1040  
 replace, 1040  
 replace\_t, 1040

MHAParser::bool\_mon\_t, 1041  
 bool\_mon\_t, 1042  
 data, 1043  
 query\_type, 1042  
 query\_val, 1042

MHAParser::bool\_t, 1043  
 bool\_t, 1044  
 data, 1045  
 op\_setval, 1045  
 query\_type, 1045  
 query\_val, 1045

MHAParser::c\_ifc\_parser\_t, 1046  
 ~c\_ifc\_parser\_t, 1047  
 c\_ifc\_parser\_t, 1047

c\_parse\_cmd, 1048  
c\_parse\_err, 1048  
libdata, 1048  
liberr, 1048  
modulename, 1048  
op\_query, 1047  
op\_setval, 1047  
op\_subparse, 1047  
ret\_size, 1048  
retv, 1048  
set\_parse\_cb, 1047  
test\_error, 1048  
MHAParser::commit\_t< receiver\_t >, 1049  
commit\_t, 1050  
extern\_connector, 1050  
MHAParser::complex\_mon\_t, 1051  
complex\_mon\_t, 1052  
data, 1052  
query\_type, 1052  
query\_val, 1052  
MHAParser::complex\_t, 1053  
complex\_t, 1054  
data, 1055  
op\_setval, 1054  
query\_type, 1054  
query\_val, 1054  
MHAParser::entry\_t, 1055  
entry, 1056  
entry\_t, 1055  
name, 1055  
MHAParser::expression\_t, 1056  
expression\_t, 1056, 1057  
lval, 1057  
op, 1057  
rval, 1057  
MHAParser::float\_mon\_t, 1057  
data, 1059  
float\_mon\_t, 1058  
query\_type, 1059  
query\_val, 1058  
MHAParser::float\_t, 1059  
data, 1062  
float\_t, 1061  
op\_setval, 1061  
query\_type, 1061  
query\_val, 1062  
MHAParser::int\_mon\_t, 1062  
data, 1064  
int\_mon\_t, 1063  
query\_type, 1064  
query\_val, 1064  
MHAParser::int\_t, 1065  
data, 1067  
int\_t, 1066  
op\_setval, 1066  
query\_type, 1067  
query\_val, 1067  
MHAParser::keyword\_list\_t, 1067  
add\_entry, 1070  
empty\_string, 1071  
entries, 1071  
get\_entries, 1070  
get\_index, 1070  
get\_value, 1069  
index, 1070  
keyword\_list\_t, 1069  
set\_entries, 1069  
set\_index, 1070  
set\_value, 1069  
size\_t, 1068  
validate, 1070  
MHAParser::kw\_t, 1071  
data, 1074  
isval, 1073  
kw\_t, 1072, 1073  
op\_setval, 1073  
query\_range, 1074  
query\_type, 1074  
query\_val, 1074  
set\_range, 1073  
validate, 1073  
MHAParser::mcomplex\_mon\_t, 1075  
data, 1076  
mcomplex\_mon\_t, 1076  
query\_type, 1076  
query\_val, 1076  
MHAParser::mcomplex\_t, 1077  
data, 1079  
mcomplex\_t, 1078  
op\_setval, 1078  
query\_type, 1078  
query\_val, 1078  
MHAParser::mfloat\_mon\_t, 1079  
data, 1081  
mfloat\_mon\_t, 1080  
query\_type, 1080  
query\_val, 1080  
MHAParser::mfloat\_t, 1081  
data, 1083  
mfloat\_t, 1082  
op\_setval, 1083  
query\_type, 1083

query\_val, 1083  
**MHAParser::mhaconfig\_mon\_t**, 1084  
 channels, 1085  
 domain, 1085  
 fftlen, 1086  
 fragsize, 1085  
**mhaconfig\_mon\_t**, 1085  
 srate, 1086  
 update, 1085  
 wndlen, 1085  
**MHAParser::mhapluginloader\_t**, 1086  
**~mhapluginloader\_t**, 1088  
 ac\_, 1090  
 bookkeeping, 1090  
 cf\_in\_, 1090  
 cf\_out\_, 1090  
 connector, 1090  
 get\_cfin, 1089  
 get\_cfout, 1089  
 get\_last\_name, 1089  
 last\_name, 1090  
 load\_plug, 1089  
**mhapluginloader\_t**, 1088  
 parent\_, 1089  
 plug, 1089  
 plugname, 1090  
 plugname\_name\_, 1090  
 prefix\_, 1090  
 prepare, 1088  
 process, 1088, 1089  
 release, 1088  
**MHAParser::mint\_mon\_t**, 1091  
 data, 1093  
**mint\_mon\_t**, 1092  
 query\_type, 1092  
 query\_val, 1092  
**MHAParser::mint\_t**, 1093  
 data, 1095  
**mint\_t**, 1094  
 op\_setval, 1095  
 query\_type, 1095  
 query\_val, 1095  
**MHAParser::monitor\_t**, 1096  
 monitor\_t, 1096, 1097  
 op\_query, 1097  
 operator=, 1097  
 query\_dump, 1097  
 query\_perm, 1097  
**MHAParser::parser\_t**, 1098  
**~parser\_t**, 1100  
 entries, 1104  
 force\_remove\_item, 1101  
 has\_entry, 1104  
 id\_string, 1104  
 insert\_item, 1100  
 last\_errormsg, 1104  
 op\_query, 1102  
 op\_setval, 1102  
 op\_subparse, 1101  
 parser\_t, 1100  
 query\_dump, 1102  
 query\_entries, 1102  
 query\_listids, 1103  
 query\_readfile, 1102  
 query\_savefile, 1103  
 query\_savefile\_compact, 1103  
 query\_savemons, 1103  
 query\_type, 1102  
 query\_val, 1103  
 remove\_item, 1100, 1101  
 set\_id\_string, 1103  
**MHAParser::range\_var\_t**, 1104  
 check\_low, 1108  
 check\_range, 1108  
 check\_up, 1108  
 low\_incl, 1108  
 low\_limit, 1108  
 query\_range, 1106  
 range\_var\_t, 1106  
 set\_range, 1106  
 up\_incl, 1108  
 up\_limit, 1108  
 validate, 1106, 1107  
**MHAParser::StrCnv**, 128  
 bracket\_balance, 130  
 num\_brackets, 129  
 str2val, 130, 131  
 str2val< mha\_real\_t >, 131  
 val2str, 132–134  
**MHAParser::string\_mon\_t**, 1109  
 data, 1111  
 query\_type, 1110  
 query\_val, 1110  
 string\_mon\_t, 1110  
**MHAParser::string\_t**, 1111  
 data, 1113  
 op\_setval, 1113  
 query\_type, 1113  
 query\_val, 1113  
 string\_t, 1112  
**MHAParser::variable\_t**, 1114  
 locked, 1116

op\_setval, 1115  
query\_perm, 1115  
setlock, 1115  
variable\_t, 1115  
MHAParser::vcomplex\_mon\_t, 1116  
  data, 1118  
  query\_type, 1117  
  query\_val, 1117  
  vcomplex\_mon\_t, 1117  
MHAParser::vcomplex\_t, 1118  
  data, 1120  
  op\_setval, 1119  
  query\_type, 1120  
  query\_val, 1120  
  vcomplex\_t, 1119  
MHAParser::vfloat\_mon\_t, 1121  
  data, 1122  
  query\_type, 1122  
  query\_val, 1122  
  vfloat\_mon\_t, 1122  
MHAParser::vfloat\_t, 1123  
  data, 1125  
  op\_setval, 1125  
  query\_type, 1125  
  query\_val, 1125  
  vfloat\_t, 1124  
MHAParser::vint\_mon\_t, 1126  
  data, 1127  
  query\_type, 1127  
  query\_val, 1127  
  vint\_mon\_t, 1127  
MHAParser::vint\_t, 1128  
  data, 1130  
  op\_setval, 1129  
  query\_type, 1129  
  query\_val, 1129  
  vint\_t, 1129  
MHAParser::vstring\_mon\_t, 1130  
  data, 1132  
  query\_type, 1132  
  query\_val, 1131  
  vstring\_mon\_t, 1131  
MHAParser::vstring\_t, 1132  
  data, 1134  
  op\_setval, 1133  
  query\_type, 1133  
  query\_val, 1134  
  vstring\_t, 1133  
MHAParser::window\_t, 1134  
  get\_type, 1137  
  get\_window, 1136, 1137  
  setlock, 1137  
  user, 1138  
  window\_t, 1136  
  wnd\_bartlett, 1136  
  wnd\_blackman, 1136  
  wnd\_hamming, 1136  
  wnd\_hann, 1136  
  wnd\_rect, 1136  
  wnd\_user, 1136  
  wtype, 1138  
  wtype\_t, 1136  
MHAPLATFORM  
  mha\_parser.cpp, 1610  
mhaplug\_cfg\_t, 1138  
  ~mhaplug\_cfg\_t, 1139  
  mhaplug\_cfg\_t, 1138  
MHAPlugin, 134  
MHAPlugin::cfg\_node\_t< runtime\_cfg\_t >, 1139  
  ~cfg\_node\_t, 1140  
  cfg\_node\_t, 1140  
  data, 1141  
  next, 1140  
  not\_in\_use, 1141  
MHAPlugin::config\_t< runtime\_cfg\_t >, 1142  
  ~config\_t, 1144  
  cfg, 1146  
  cfg\_node\_current, 1147  
  cfg\_root, 1147  
  cleanup\_unused\_cfg, 1146  
  config\_t, 1144  
  peek\_config, 1145  
  poll\_config, 1145  
  push\_config, 1146  
  remove\_all\_cfg, 1146  
MHAPlugin::plugin\_t< runtime\_cfg\_t >, 1147  
  ~plugin\_t, 1149  
  ac, 1152  
  input\_cfg, 1151  
  input\_cfg\_, 1152  
  is\_prepared, 1151  
  is\_prepared\_, 1152  
  mhaconfig\_in, 1153  
  mhaconfig\_out, 1153  
  output\_cfg, 1151  
  output\_cfg\_, 1152  
  plugin\_t, 1149  
  prepare, 1149  
  prepare\_, 1151  
  release, 1150  
  release\_, 1151

tftype, 1152

MHAPLUGIN\_CALLBACKS  
  mha\_plugin.hh, 1620

MHAPLUGIN\_CALLBACKS\_PREFIX  
  mha\_plugin.hh, 1619

MHAPLUGIN\_DOCUMENTATION  
  mha\_plugin.hh, 1621

MHAPLUGIN\_DOCUMENTATION\_PREFIX  
  mha\_plugin.hh, 1620

MHAPLUGIN\_INIT\_CALLBACKS  
  mha\_plugin.hh, 1620

MHAPLUGIN\_INIT\_CALLBACKS\_PREFIX  
  mha\_plugin.hh, 1619

MHAPLUGIN\_OVERLOAD\_OUTDOMAIN  
  altconfig.hh, 1529  
  altplugs.cpp, 1530  
  matlab\_wrapper.hh, 1569  
  mha\_generic\_chain.h, 1597  
  split.cpp, 1705  
  wave2spec.hh, 1708

MHAPLUGIN\_PROC\_CALLBACK  
  mha\_plugin.hh, 1620

MHAPLUGIN\_PROC\_CALLBACK\_PREFIX  
  mha\_plugin.hh, 1619

MHAPlugin\_Resampling, 134

MHAPlugin\_Resampling::resampling\_if\_t,  
  1153  
  algo, 1156  
  fragsize, 1155  
  irslen\_inner2outer, 1155  
  irslen\_outer2inner, 1155  
  nyquist\_ratio, 1155  
  plugloader, 1155  
  prepare, 1154  
  process, 1154  
  release, 1155  
  resampling\_if\_t, 1154  
  srates, 1155

MHAPlugin\_Resampling::resampling\_t, 1156  
  inner2outer\_resampling, 1158  
  inner\_fragsize, 1157  
  inner\_signal, 1158  
  inner\_srates, 1157  
  nchannels\_in, 1157  
  nchannels\_out, 1158  
  outer2inner\_resampling, 1158  
  outer\_fragsize, 1157  
  outer\_srates, 1157  
  output\_signal, 1158  
  plugloader, 1158  
  process, 1157

  resampling\_t, 1156

  MHAPLUGIN\_SETCPP\_CALLBACK\_PREFIX  
    mha\_plugin.hh, 1619

  MHAPlugin\_Split, 135  
    INVALID\_THREAD\_PRIORITY, 136

  MHAPlugin\_Split::domain\_handler\_t, 1159  
    ~domain\_handler\_t, 1161  
    deallocate\_domains, 1162  
    domain\_handler\_t, 1160  
    get\_signal, 1163  
    operator=, 1161  
    process, 1164  
    processor, 1165  
    put\_signal, 1162  
    set\_input\_domain, 1161  
    set\_output\_domain, 1161  
    spec\_in, 1165  
    spec\_out, 1165  
    wave\_in, 1164  
    wave\_out, 1164

  MHAPlugin\_Split::dummy\_threads\_t, 1165  
    catch\_thread, 1167  
    dummy\_threads\_t, 1166  
    kick\_thread, 1166

  MHAPlugin\_Split::posix\_threads\_t, 1167  
    ~posix\_threads\_t, 1169  
    attr, 1171  
    catch\_condition, 1171  
    catch\_thread, 1169  
    current\_thread\_priority, 1170  
    current\_thread\_scheduler, 1170  
    kick\_condition, 1170  
    kick\_thread, 1169  
    kicked, 1171  
    main, 1170  
    mutex, 1170  
    posix\_threads\_t, 1169  
    priority, 1171  
    processing\_done, 1171  
    scheduler, 1171  
    termination\_request, 1172  
    thread, 1171  
    thread\_start, 1170

  MHAPlugin\_Split::split\_t, 1172  
    ~split\_t, 1174  
    algos, 1177  
    chains, 1178  
    channels, 1177  
    clear\_chains, 1175  
    collect\_result, 1176  
    copy\_output\_spec, 1175

copy\_output\_wave, 1175  
delay, 1178  
framework\_thread\_priority, 1178  
framework\_thread\_scheduler, 1177  
patchbay, 1176  
prepare\_, 1175  
process, 1175  
release\_, 1175  
signal\_out, 1176  
spec\_out, 1178  
split\_t, 1174  
thread\_platform, 1177  
trigger\_processing, 1176  
update, 1175  
wave\_out, 1178  
worker\_thread\_priority, 1177  
worker\_thread\_scheduler, 1177  
MHAPlugin\_Split::splitted\_part\_t, 1179  
~splitted\_part\_t, 1181  
collect\_result, 1183  
domain, 1183  
operator=, 1181  
parse, 1182  
plug, 1183  
prepare, 1181  
release, 1182  
splitted\_part\_t, 1180, 1181  
thread, 1184  
trigger\_processing, 1182  
MHAPlugin\_Split::thread\_platform\_t, 1184  
~thread\_platform\_t, 1186  
catch\_thread, 1186  
kick\_thread, 1186  
operator=, 1186  
processor, 1187  
thread\_platform\_t, 1185  
MHAPlugin\_Split::uni\_processor\_t, 1187  
~uni\_processor\_t, 1188  
process, 1188  
MHAPluginCategory\_t  
  mha.hh, 1578  
MHAPluginDocumentation\_t  
  mha.hh, 1578  
mhapluginloader.cpp, 1688  
mhapluginloader.h, 1688  
mhapluginloader\_t  
  MHAParser::mhapluginloader\_t, 1088  
  PluginLoader::mhapluginloader\_t, 1366  
MHAPrepare\_cb  
  PluginLoader::mhapluginloader\_t, 1370  
MHAPrepare\_t  
  mha.hh, 1577  
  MHAProc\_spec2spec\_cb  
    PluginLoader::mhapluginloader\_t, 1371  
  MHAProc\_spec2spec\_t  
    mha.hh, 1578  
  MHAProc\_spec2wave\_cb  
    PluginLoader::mhapluginloader\_t, 1371  
  MHAProc\_spec2wave\_t  
    mha.hh, 1578  
  MHAProc\_wave2spec\_cb  
    PluginLoader::mhapluginloader\_t, 1371  
  MHAProc\_wave2spec\_t  
    mha.hh, 1578  
  MHAProc\_wave2wave\_cb  
    PluginLoader::mhapluginloader\_t, 1371  
  MHAProc\_wave2wave\_t  
    mha.hh, 1578  
  MHARelease\_cb  
    PluginLoader::mhapluginloader\_t, 1370  
  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, 1371  
  MHASet\_t  
    mha.hh, 1577  
  MHASetcpp\_cb  
    PluginLoader::mhapluginloader\_t, 1371  
  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, 1688**  
 validator\_channels, 1689  
 validator\_length, 1689  
 write\_wave, 1688  
**mhasndfile.h, 1689**  
 write\_wave, 1689  
**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::mhapluginloader\_t, 1371  
**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\_xlims, 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, 1400  
MIN  
    mha\_defs.h, 1583  
min  
    MHASignal::doublebuffer\_t, 1206  
    spec2wave.cpp, 1699  
    Vector and matrix processing toolbox, 56  
min\_sleep\_time  
    dropgen\_t, 435  
MIN\_TCP\_PORT  
    MHAIOAsterisk.cpp, 1653  
    MHAITCP.cpp, 1680  
MIN\_TCP\_PORT\_STR  
    MHAIOAsterisk.cpp, 1653  
    MHAITCP.cpp, 1680  
minimum\_fill\_count  
    mha\_drifter\_fifo\_t< T >, 757  
minlen  
    plingploing::if\_t, 1339  
minlen\_  
    plingploing::plingploing\_t, 1342  
minLim  
    rohBeam::rohConfig, 1408  
minphase  
    MHAFilter::smoothspec\_t, 930  
minphase\_t  
    MHASignal::minphase\_t, 1233  
mint\_mon\_t  
    MHAParser::mint\_mon\_t, 1092  
mint\_t  
    MHAParser::mint\_t, 1094  
minw\_  
    wavwriter\_t, 1503  
minwrite  
    plugins::hoertech::acrec::acrec\_t, 1377  
    wavrec\_t, 1500  
mismatch  
    level\_matching::channel\_pair, 647  
mix  
    sine\_cfg\_t, 1430  
mixer  
    matrixmixer::matmix\_t, 712  
mixw\_ref  
    fshift\_hilbert::hilbert\_shifter\_t, 517  
mixw\_shift  
    fshift\_hilbert::hilbert\_shifter\_t, 517  
mode  
    ac2osc\_t, 183  
    addsndfile::addsndfile\_if\_t, 253  
    audiometerbackend::audiometer\_if\_t, 317  
    levelmeter\_t, 656  
    MHA\_TCP::OS\_EVENT\_TYPE, 811  
    noise\_t, 1316  
    sine\_t, 1433  
    smoothgains\_bridge::overlapadd\_if\_t,  
        1451  
modified  
    dc::dc\_vars\_t, 390  
    dc\_simple::dc\_if\_t, 395  
modulename  
    dynamiclib\_t, 446  
    MHAParser::c\_ifc\_parser\_t, 1048  
mon  
    acmon::ac\_monitor\_t, 205  
mon\_complex  
    acmon::ac\_monitor\_t, 206  
mon\_dump  
    MHAParser, 127  
mon\_g  
    dc\_simple::dc\_if\_t, 395  
    dc\_simple::dc\_t, 399  
mon\_l  
    dc\_simple::dc\_if\_t, 395  
    dc\_simple::dc\_t, 399  
mon\_mat  
    acmon::ac\_monitor\_t, 206  
mon\_mat\_complex  
    acmon::ac\_monitor\_t, 206

mon\_t  
     rmslevel::mon\_t, 1387  
 monitor variable, 4  
 monitor\_t  
     MHAParser::monitor\_t, 1096, 1097  
 monitors  
     matlab\_wrapper::matlab\_wrapper\_t, 699  
     rmslevel::rmslevel\_t, 1393  
 mpo  
     DynComp::dc\_afterburn\_vars\_t, 453  
 mpo\_inv  
     DynComp::dc\_afterburn\_rt\_t, 448  
 msg  
     MHA\_Error, 762  
 mu  
     gsc\_adaptive\_stage::gsc\_adaptive\_stage, 537  
     gsc\_adaptive\_stage::gsc\_adaptive\_stage\_if\_h, 544  
     MHAFilter::adapt\_filter\_param\_t, 860  
     MHAFilter::adapt\_filter\_t, 864  
 mu\_beta  
     adm\_if\_t, 274  
 mul4f  
     gtfb\_simd.cpp, 1558  
 multibandcompressor, 155  
 multibandcompressor.cpp, 1690  
 multibandcompressor::ffftb\_plug\_t, 1298  
     bwv, 1299  
     cfv, 1299  
     efv, 1299  
     ffftb\_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, 772  
     MHAParser\_Split::posix\_threads\_t, 1170  
 MXCSR\_DAZ  
     gtfb\_simd.cpp, 1558  
 MXCSR\_FTZ  
     gtfb\_simd.cpp, 1558  
 mXY  
     MHATableLookup::xy\_table\_t, 1285  
 mylogf  
     dc\_afterburn.cpp, 1539

N

N  
     lpc\_config, 674  
     MHAJack::client\_avg\_t, 966  
     MHASignal::hilbert\_fftw\_t, 1213  
     MHASignal::stat\_t, 1252  
 n\_channels  
     mha\_audio\_descriptor\_t, 738  
 N\_ERRNO  
     MHA\_TCP, 102  
 n\_freqs  
     mha\_audio\_descriptor\_t, 738  
 n\_im  
     MHASignal::fft\_t, 1210  
 n\_new\_samples  
     lsl2ac::save\_var\_t, 689  
 n\_no\_update  
     adaptive\_feedback\_canceller, 244  
     nlms\_t, 1308  
 n\_no\_update\_  
     adaptive\_feedback\_canceller\_config, 247  
     rt\_nlms\_t, 1417  
 n\_pad1  
     overlapadd::overlapadd\_t, 1333  
 n\_pad2  
     overlapadd::overlapadd\_t, 1334  
 n\_re  
     MHASignal::fft\_t, 1210  
 n\_samples  
     mha\_audio\_descriptor\_t, 738  
 n\_zero  
     overlapadd::overlapadd\_t, 1333  
 name  
     ac2lsl::type\_info, 179  
     ac2wave\_if\_t, 187  
     ac2wave\_t, 190

acmon::ac\_monitor\_t, 205  
acsave::save\_var\_t, 227  
fftfbpow::fftfbpow\_interface\_t, 492  
lsl2ac::save\_var\_t, 688  
MHA\_AC::ac2matrix\_helper\_t, 718  
MHA\_AC::double\_t, 727  
MHA\_AC::float\_t, 729  
MHA\_AC::int\_t, 730  
MHA\_AC::spectrum\_t, 733  
MHA\_AC::waveform\_t, 737  
MHAIOPortAudio::device\_info\_t, 953  
MHAJack::client\_avg\_t, 966  
MHAJack::client\_noncont\_t, 970  
MHAMultiSrc::channel\_t, 993  
MHAParser::entry\_t, 1055  
noise\_psd\_estimator::noise\_psd\_estimator\_if\_t, acsave::mat4head\_t, 225  
1310  
plugindescription\_t, 1355  
rmslevel::rmslevel\_if\_t, 1390  
shadowfilter\_end::cfg\_t, 1426  
name\_  
AuditoryProfile::parser\_t::fmap\_t, 330  
gtfb\_simple\_t, 570  
osc\_variable\_t, 1326  
name\_b  
lpc\_bl\_predictor, 663  
lpc\_bl\_predictor\_config, 666  
lpc\_burglattice, 669  
lpc\_burglattice\_config, 672  
name\_con\_AC  
acConcat\_wave, 201  
name\_d  
nlms\_t, 1307  
name\_d\_  
adaptive\_feedback\_canceller\_config, 247  
rt\_nlms\_t, 1417  
name\_e  
adaptive\_feedback\_canceller, 243  
nlms\_t, 1307  
name\_e\_  
rt\_nlms\_t, 1417  
name\_f  
adaptive\_feedback\_canceller, 243  
lpc\_bl\_predictor, 663  
lpc\_bl\_predictor\_config, 665  
lpc\_burglattice, 669  
lpc\_burglattice\_config, 671  
nlms\_t, 1308  
name\_kappa  
lpc\_bl\_predictor, 663  
lpc\_burglattice, 669  
name\_km  
lpc\_bl\_predictor\_config, 665  
name\_lpc  
adaptive\_feedback\_canceller, 243  
name\_lpc\_  
adaptive\_feedback\_canceller\_config, 247  
name\_lpc\_b  
lpc\_bl\_predictor, 663  
name\_lpc\_f  
lpc\_bl\_predictor, 663  
name\_u  
nlms\_t, 1307  
name\_u\_  
rt\_nlms\_t, 1417  
namelen  
names  
MHAOvIFilter::scale\_var\_t, 1026  
nangle  
acSteer\_config, 232  
steerbf\_config, 1474  
native\_thread\_platform\_type  
split.cpp, 1705  
naudiochannels  
dc::dc\_t, 386  
nbands  
coherence::cohflt\_t, 351  
combc\_t, 359  
dc::dc\_t, 386  
dc\_simple::dc\_t, 398  
dc\_simple::level\_smoothen\_t, 406  
DynComp::gaintable\_t, 457  
fftfilterbank::fftfb\_interface\_t, 502  
gtfb\_simple\_rt\_t, 564  
MHAFilter::thirdoctave\_analyzer\_t, 931  
MHAOvIFilter::fspacing\_t, 1016  
nbits  
calibrator\_variables\_t, 342  
nch  
dc::dc\_t, 386  
shadowfilter\_begin::cfg\_t, 1422  
shadowfilter\_begin::shadowfilter\_begin\_t,  
1425  
spec\_fader\_t, 1466  
nch\_out  
shadowfilter\_end::cfg\_t, 1426  
nchan  
acSteer\_config, 232  
gsc\_adaptive\_stage::gsc\_adaptive\_stage,  
537  
smooth\_cepstrum::smooth\_cepstrum\_t,

**1442**  
 steerbf\_config, 1474  
**nchan\_block**  
 rohBeam::rohConfig, 1405  
**nchannels**  
 DynComp::gaintable\_t, 457  
 equalize::cfg\_t, 460  
 fftfilterbank::fftfb\_interface\_t, 502  
 lsl2ac::lsl2ac\_t, 681  
 lsl2ac::save\_var\_t, 688  
 MHAFilter::adapt\_filter\_state\_t, 861  
 MHAFilter::adapt\_filter\_t, 864  
 MHAFilter::iir\_filter\_t, 898  
 MHAFilter::smoothspec\_t, 929  
 MHAFilter::thirdoctave\_analyzer\_t, 931  
**nchannels\_file\_in**  
 io\_file\_t, 604  
**nchannels\_in**  
 io\_file\_t, 604  
 io\_parser\_t, 615  
 mconv::MConv, 716  
 MHAFilter::partitioned\_convolution\_t, 915  
 MHAIOPortAudio::io\_portaudio\_t, 959  
 MHAJack::client\_t, 978  
 MHAPlugin\_Resampling::resampling\_t, 1157  
**nchannels\_out**  
 fw\_t, 525  
 io\_file\_t, 604  
 io\_parser\_t, 615  
 mconv::MConv, 716  
 MHAFilter::partitioned\_convolution\_t, 915  
 MHAIOPortAudio::io\_portaudio\_t, 959  
 MHAJack::client\_t, 978  
 MHAPlugin\_Resampling::resampling\_t, 1158  
**NDEBUG**  
 rohBeam.hh, 1696  
**ndim**  
 acsave::save\_var\_t, 227  
**needs\_write**  
 MHA\_TCP::Connection, 806  
**neigh**  
 acPooling\_wave\_config, 217  
**neighbourhood**  
 acPooling\_wave, 214  
**nelements**  
 MHASignal::matrix\_t, 1232  
**nested\_lock**  
 MHAParser::base\_t, 1039  
**new\_name**  
 lsl2ac::save\_var\_t, 687  
 newgains  
 fader\_if\_t, 485  
**next**  
 mha\_rt\_fifo\_element\_t< T >, 785  
 MHAPlugin::cfg\_node\_t< runtime\_cfg\_t >, 1140  
**next\_except\_str**  
 mha\_errno.c, 1585  
**next\_message**  
 mha\_tcp::buffered\_socket\_t, 797  
**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, 1329  
 shadowfilter\_end::cfg\_t, 1426  
 spec2wave\_t, 1465  
 wave2spec\_if\_t, 1491  
**nfft\_**  
 MHAOvIFilter::fspacing\_t, 1017  
**nframes**  
 acsave::save\_var\_t, 227  
**nfreq**  
 acSteer\_config, 232  
 gsc\_adaptive\_stage::gsc\_adaptive\_stage, 537  
 rohBeam::rohConfig, 1405  
 smooth\_cepstrum::smooth\_cepstrum\_t, 1441  
 steerbf\_config, 1474  
**nlms\_t**, 1305  
 algo, 1308  
 c, 1307  
 estimtype, 1307  
 lambda\_smoothing\_power, 1307  
**n\_no\_update**, 1308  
 name\_d, 1307  
 name\_e, 1307  
 name\_f, 1308  
 name\_u, 1307  
 nlms\_t, 1306  
 normtype, 1307  
 ntaps, 1307  
 patchbay, 1308  
 prepare, 1306  
 process, 1306

release, 1306  
rho, 1307  
update, 1306  
nlms\_wave.cpp, 1690  
ESTIM\_CUR, 1691  
ESTIM\_PREV, 1691  
ESTIMATION\_TYPES, 1691  
make\_friendly\_number\_by\_limiting, 1691  
NORM\_DEFAULT, 1691  
NORM\_NONE, 1691  
NORM\_SUM, 1691  
NORMALIZATION\_TYPES, 1691  
nm  
    lpc\_burglattice\_config, 671  
no\_iter  
    adaptive\_feedback\_canceller\_config, 247  
    rt\_nlms\_t, 1417  
noise.cpp, 1692  
noise\_field\_model  
    rohBeam::rohBeam, 1400  
noise\_integrate\_hrtf  
    rohBeam::rohBeam, 1398  
noise\_psd\_estimator, 155  
noise\_psd\_estimator.cpp, 1692  
    POWSPEC\_FACTOR, 1692  
noise\_psd\_estimator::noise\_psd\_estimator\_if\_t, noiseFuncPtr  
    1308  
    alphaPH1mean, 1310  
    alphaPSD, 1310  
    name, 1310  
    noise\_psd\_estimator\_if\_t, 1309  
    patchbay, 1310  
    prepare, 1309  
    process, 1309  
    q, 1310  
    update\_cfg, 1310  
    xiOptDb, 1310  
noise\_psd\_estimator::noise\_psd\_estimator\_t, 1311  
    alphaPH1mean\_, 1313  
    alphaPSD\_, 1313  
    estimateDebug, 1313  
    frameno, 1314  
    GLRDebug, 1313  
    GLRexp, 1314  
    inputPow, 1312  
    inputSpec, 1313  
    insert, 1312  
    logGLRFact, 1314  
    noise\_psd\_estimator\_t, 1311  
    noisePow, 1312  
    noisyPer, 1312  
    PH1Debug, 1313  
    PH1mean, 1312  
    priorFact, 1313  
    process, 1312  
    snrPost1Debug, 1313  
    xiOpt, 1314  
    noise\_psd\_estimator\_if\_t  
        noise\_psd\_estimator::noise\_psd\_estimator\_if\_t,  
            1309  
    noise\_psd\_estimator\_t  
        noise\_psd\_estimator::noise\_psd\_estimator\_t,  
            1311  
    noise\_t, 1314  
        frozennoise\_length, 1316  
        lev, 1316  
        mode, 1316  
        noise\_t, 1315  
        patchbay, 1316  
        prepare, 1315  
        process, 1315  
        seed, 1316  
        update\_cfg, 1316  
    noise\_type\_t  
        speechnoise\_t, 1468  
    noiseModelExport  
        rohBeam::rohBeam, 1399  
    noisePow  
        noise\_psd\_estimator::noise\_psd\_estimator\_t,  
            1312  
        smooth\_cepstrum::smooth\_cepstrum\_t,  
            1443  
    noisePow\_name  
        smooth\_cepstrum::smooth\_cepstrum\_if\_t,  
            1438  
        smooth\_cepstrum::smooth\_params, 1449  
    noisyPer  
        noise\_psd\_estimator::noise\_psd\_estimator\_t,  
            1312  
    non\_empty\_partitions  
        MHAFilter::transfer\_function\_t, 935  
        MHAFilter::transfer\_matrix\_t, 937  
    nondefault\_labels  
        altpicks\_t, 304  
    NORELEASE\_WARNING  
        mhamain.cpp, 1687  
    norm  
        lpc, 660  
        lpc\_config, 673

NORM\_DEFAULT  
     nlms\_wave.cpp, 1691

NORM\_NONE  
     nlms\_wave.cpp, 1691

norm\_phase  
     gtfb\_analyzer::gtfb\_analyzer\_cfg\_t, 548  
     gtfb\_analyzer::gtfb\_analyzer\_t, 552  
     gtfb\_simd\_cfg\_t, 556  
     gtfb\_simd\_t, 560

NORM\_SUM  
     nlms\_wave.cpp, 1691

NORMALIZATION\_TYPES  
     nlms\_wave.cpp, 1691

normalize  
     Complex arithmetics in the openMHA, 68  
     MHAoVFilter::ffftb\_vars\_t, 1009

normtype  
     nlms\_t, 1307

not\_in\_use  
     MHAPlugin::cfg\_node\_t< runtime\_cfg\_t >, 1141

not\_zero  
     dc\_simple, 88

notify  
     MHAParser::base\_t, 1037

notify\_release  
     io\_asterisk\_t, 600  
     io\_tcp\_t, 638

notify\_start  
     io\_asterisk\_t, 600  
     io\_tcp\_t, 638

notify\_stop  
     io\_asterisk\_t, 600  
     io\_tcp\_t, 638

now\_index  
     MHAFilter::polyphase\_resampling\_t, 923

npad1  
     spec2wave\_t, 1464  
     wave2spec\_t, 1497

npad2  
     spec2wave\_t, 1464  
     wave2spec\_t, 1497

nperiods  
     alsa\_dev\_par\_parser\_t, 290

nrefmic  
     acSteer, 230  
     acSteer\_config, 232

nrep  
     MHAJack::client\_avg\_t, 966

nsamples  
     lsl2ac::lsl2ac\_t, 681

lsl2ac::save\_var\_t, 688

nsteerchan  
     acSteer, 230  
     acSteer\_config, 232

ntaps  
     adaptive\_feedback\_canceller, 243  
     adaptive\_feedback\_canceller\_config, 246  
     MHAFilter::adapt\_filter\_state\_t, 861  
     MHAFilter::adapt\_filter\_t, 864  
     nlms\_t, 1307  
     rt\_nlms\_t, 1415

ntoh  
     io\_asterisk\_sound\_t, 595  
     io\_tcp\_sound\_t, 632

ntracks  
     shadowfilter\_begin::cfg\_t, 1423  
     shadowfilter\_begin::shadowfilter\_begin\_t, 1425  
     shadowfilter\_end::cfg\_t, 1426

null\_data  
     mha\_drifter\_fifo\_t< T >, 759

num\_AC  
     acConcat\_wave, 201

num\_accepted\_connections  
     mha\_tcp::server\_t, 820

num\_adms  
     adm\_rtconfig\_t, 277

num\_bins  
     equalize::cfg\_t, 460

num\_brackets  
     MHAParser::StrCnv, 129

num\_channels  
     ac\_mul\_t, 195  
     calibrator\_variables\_t, 343  
     DynComp::gaintable\_t, 458  
     mha\_spec\_t, 791  
     mha\_wave\_t, 837  
     MHAFilter::blockprocessing\_polyphase\_resampling\_t, 868  
     multibandcompressor::interface\_t, 1302  
     plugins::hoertech::acrec::acwriter\_t, 1383

NUM\_ENTR\_LTASS  
     speechnoise.cpp, 1701

NUM\_ENTR\_MHAORIG  
     speechnoise.cpp, 1701

NUM\_ENTR\_OLNOISE  
     speechnoise.cpp, 1701

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, 360  
testplugin::ac\_parser\_t, 1477  
num\_F  
    DynComp::gaintable\_t, 458  
num\_frames  
    ac\_mul\_t, 195  
    mha\_spec\_t, 791  
    mha\_wave\_t, 837  
num\_inchannels  
    io\_asterisk\_sound\_t, 596  
    io\_tcp\_sound\_t, 633  
num\_L  
    DynComp::gaintable\_t, 458  
num\_outchannels  
    io\_asterisk\_sound\_t, 596  
    io\_tcp\_sound\_t, 633  
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, 953  
numsamples  
    acPooling\_wave, 213  
    acTransform\_wave, 237  
numSamples\_AC  
    acConcat\_wave\_config, 203  
nupsample  
    doasvm\_feature\_extraction, 425  
nvvars  
    acsave::cfg\_t, 223  
nwnd  
    overlapadd::overlapadd\_if\_t, 1329  
    wave2spec\_if\_t, 1491  
    wave2spec\_t, 1496  
nwndshift  
    spec2wave\_t, 1465  
    wave2spec\_t, 1496  
nyquist\_ratio  
    MHAPlugin\_Resampling::resampling\_if\_t, 1155  
o1\_ar\_filter\_t  
    MHAFilter::o1\_ar\_filter\_t, 903  
o1\_lp\_coeffs  
    MHAFilter, 106  
o1flt\_lowpass\_t  
    MHAFilter::o1flt\_lowpass\_t, 907  
o1flt\_maxtrack\_t  
    MHAFilter::o1flt\_maxtrack\_t, 910  
o1flt\_mintrack\_t  
    MHAFilter::o1flt\_mintrack\_t, 912  
ob  
    lsl2ac::save\_var\_t, 688  
observe  
    MHA\_TCP::Event\_Watcher, 809  
observed\_by  
    MHA\_TCP::Wakeup\_Event, 833  
observers  
    MHA\_TCP::Wakeup\_Event, 834  
od  
    MHAFilter::adapt\_filter\_state\_t, 862  
offset  
    acTransform\_wave\_config, 239  
    dc::dc\_t, 385  
    dc::dc\_vars\_t, 389  
ola\_powspec\_scale  
    smooth\_cepstrum::smooth\_cepstrum\_t, 1442  
old\_algos  
    mhachain::chain\_base\_t, 840  
olnoise  
    speechnoise\_t, 1468  
on\_configuration\_update  
    double2acvar::double2acvar\_t, 432  
on\_model\_param\_valuechanged  
    gsc\_adaptive\_stage::gsc\_adaptive\_stage\_if, 544  
    rohBeam::rohBeam, 1399  
    smooth\_cepstrum::smooth\_cepstrum\_if\_t, 1436  
on\_preadaccess  
    example3\_t, 471  
    example4\_t, 475  
on\_received\_line  
    mha\_tcp::server\_t, 818  
    mhaserver\_t, 1191  
    mhaserver\_t::tcp\_server\_t, 1194  
on\_scale\_ch\_readaccess  
    example3\_t, 471  
    example4\_t, 475  
on\_scale\_ch\_valuechanged  
    example3\_t, 471  
    example4\_t, 475  
on\_scale\_ch\_writeaccess  
    example3\_t, 471  
    example4\_t, 475  
on\_set\_algos

altconfig\_t, 297  
 on\_set\_select  
     altconfig\_t, 297  
 on\_writeaccess  
     matlab\_wrapper::callback, 690  
 op  
     MHAParser::expression\_t, 1057  
 op\_query  
     MHAParser::base\_t, 1033  
     MHAParser::c\_ifc\_parser\_t, 1047  
     MHAParser::monitor\_t, 1097  
     MHAParser::parser\_t, 1102  
 op\_setval  
     MHAParser::base\_t, 1033  
     MHAParser::bool\_t, 1045  
     MHAParser::c\_ifc\_parser\_t, 1047  
     MHAParser::complex\_t, 1054  
     MHAParser::float\_t, 1061  
     MHAParser::int\_t, 1066  
     MHAParser::kw\_t, 1073  
     MHAParser::mcomplex\_t, 1078  
     MHAParser::mfloat\_t, 1083  
     MHAParser::mint\_t, 1095  
     MHAParser::parser\_t, 1102  
     MHAParser::string\_t, 1113  
     MHAParser::variable\_t, 1115  
     MHAParser::vcomplex\_t, 1119  
     MHAParser::vfloat\_t, 1125  
     MHAParser::vint\_t, 1129  
     MHAParser::vstring\_t, 1133  
 op\_subparse  
     MHAParser::base\_t, 1032  
     MHAParser::c\_ifc\_parser\_t, 1047  
     MHAParser::parser\_t, 1101  
 opact\_map\_t  
     MHAParser, 125  
 opact\_t  
     MHAParser, 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, 1637  
 operator>>  
     mha\_signal.hh, 1638  
 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()  
     dc\_simple::dc\_t::line\_t, 400  
     hanning\_ramps\_t, 571  
     MHAEvents::emitter\_t, 856  
     MHAFilter::gamma\_flt\_t, 890  
     MHAFilter::iir\_ord1\_real\_t, 900, 901  
     MHAFilter::o1\_ar\_filter\_t, 904, 905  
     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, 459
- gtfb\_simd\_cfg\_t, 555
- lsl2ac::save\_var\_t, 685
- MHA\_AC::acspace2matrix\_t, 723
- MHA\_Error, 762
- mha\_fifo\_t< T >, 777
- mha\_fifo\_thread\_platform\_t, 783
- MHAFilter::filter\_t, 886
- MHAParser::base\_t, 1031

MHAParser::monitor\_t, 1097  
MHAParser::monitor\_t, 1161  
MHAParser::monitor\_t, 1181  
MHAParser::monitor\_t, 1186  
MHASignal::matrix\_t, 1226  
MHASignal::uint\_vector\_t, 1257  
MHASignal::waveform\_t, 1263  
rmslevel::mon\_t, 1387  
rohBeam::rohConfig, 1404  
smooth\_cepstrum::smooth\_cepstrum\_t, 1441  
operator==  
    Complex arithmetics in the openMHA, 66  
    MHASignal::uint\_vector\_t, 1257  
operator[]  
    MHA\_AC::acspace2matrix\_t, 723, 724  
    MHASignal::spectrum\_t, 1247  
    MHASignal::uint\_vector\_t, 1257  
    MHASignal::waveform\_t, 1263  
operators  
    MHAParser::base\_t, 1039  
oplist  
    MHAParser::base\_t, 1037  
order  
    gtfb\_analyzer::gtfb\_analyzer\_cfg\_t, 548  
    gtfb\_analyzer::gtfb\_analyzer\_t, 552  
    gtfb\_simd\_cfg\_t, 555  
    gtfb\_simd\_t, 560  
    gtfb\_simple\_t, 569  
    lpc\_config, 674  
original\_content  
    mha\_stash\_environment\_variable\_t, 793  
origname  
    PluginLoader::config\_file\_splitter\_t, 1360  
os\_event  
    MHA\_TCP::Wakeup\_Event, 834  
os\_event\_valid  
    MHA\_TCP::Wakeup\_Event, 835  
osc2ac.cpp, 1692  
osc2ac\_t, 1317  
    host, 1319  
    osc2ac\_t, 1318  
    patchbay, 1319  
    port, 1319  
    prepare, 1318  
    process, 1318  
    release, 1318  
    setlock, 1319  
    size, 1319  
srv, 1319  
vars, 1319  
osc\_data  
    osc\_variable\_t, 1326  
osc\_server\_t, 1320  
    ~osc\_server\_t, 1320  
    ac\_insert, 1321  
    error\_h, 1321  
    insert\_variable, 1321  
    is\_running, 1322  
    lost, 1322  
    osc\_server\_t, 1320  
    pVars, 1321  
    server\_start, 1321  
    server\_stop, 1321  
    sync\_osc2ac, 1321  
osc\_variable\_t, 1322  
    ac\_data, 1325  
    ac\_insert, 1324  
    acname, 1325  
    handler, 1324, 1325  
    name\_, 1326  
    osc\_data, 1326  
    osc\_variable\_t, 1323  
    oscaddr, 1325  
    sync\_osc2ac, 1324  
oscaddr  
    osc\_variable\_t, 1325  
out  
    adm\_if\_t, 273  
    delaysum::delaysum\_wave\_t, 413  
out\_buf  
    overlapadd::overlapadd\_t, 1333  
    spec2wave\_t, 1465  
out\_cfg  
    rohBeam::rohConfig, 1405  
out\_chunk  
    MHAFilter::thirdoctave\_analyzer\_t, 932  
out\_chunk\_im  
    MHAFilter::thirdoctave\_analyzer\_t, 933  
out\_spec  
    shadowfilter\_begin::cfg\_t, 1422  
    shadowfilter\_end::cfg\_t, 1427  
outbuf  
    MHA\_TCP::Connection, 806  
outch  
    mconv::MConv, 716  
    MHAJack::client\_t, 979  
outchannel  
    audiometerbackend::audiometer\_if\_t, 317  
outchannels

combc\_if\_t, 357  
 outer2inner\_resampling  
     MHAPlugin\_Resampling::resampling\_t,  
         1158  
 outer\_ac  
     analysepath\_t, 308  
 outer\_ac\_copy  
     analysepath\_t, 308  
 outer\_error  
     mha\_dbdbuf\_t< FIFO >, 751  
 outer\_fragsize  
     MHAPlugin\_Resampling::resampling\_t,  
         1157  
 outer\_out  
     MHASignal::doublebuffer\_t, 1206  
 outer\_output  
     dbasync\_native::dbasync\_t, 378  
 outer\_process  
     dbasync\_native::dbasync\_t, 377  
     MHASignal::doublebuffer\_t, 1205  
 outer\_size  
     mha\_dbdbuf\_t< FIFO >, 749  
 outer\_srate  
     MHAPlugin\_Resampling::resampling\_t,  
         1157  
 outfile  
     plugins::hoertech::acrec::acwriter\_t, 1383  
 output  
     delaysum\_spec::delaysum\_t, 417  
     gtfb\_simple\_rt\_t, 564  
     io\_parser\_t, 616  
     mha\_dbdbuf\_t< FIFO >, 749  
     MHAJack::port\_t, 981  
 output\_cfg  
     MHAPlugin::plugin\_t< runtime\_cfg\_t >,  
         1151  
 output\_cfg\_  
     MHAPlugin::plugin\_t< runtime\_cfg\_t >,  
         1152  
 output\_channels  
     mha\_dbdbuf\_t< FIFO >, 750  
 output\_data  
     io\_asterisk\_sound\_t, 596  
 output\_domain  
     PluginLoader::mhapluginloader\_t, 1367  
 output\_fifo  
     mha\_dbdbuf\_t< FIFO >, 750  
 output\_partitions  
     MHAFilter::partitioned\_convolution\_t, 915  
 output\_portnames  
     MHAJack::client\_t, 980  
 output\_sample\_format  
     io\_file\_t, 605  
     wavrec\_t, 1500  
 output\_signal  
     MHAPlugin\_Resampling::resampling\_t,  
         1158  
 output\_signal\_spec  
     MHAFilter::partitioned\_convolution\_t, 916  
 output\_signal\_wave  
     MHAFilter::partitioned\_convolution\_t, 917  
 output\_spec  
     testplugin::signal\_parser\_t, 1485  
 output\_type  
     plugins::hoertech::acrec::acwriter\_t, 1379  
 output\_wave  
     testplugin::signal\_parser\_t, 1485  
 outputchannels  
     MHAFilter::fftfilterbank\_t, 882  
 outSpec  
     rohBeam::rohConfig, 1406  
     steerbf\_config, 1474  
 overlap\_save\_filterbank\_analytic\_t  
     MHAOvlFilter::overlap\_save\_filterbank\_analytic\_t,  
         1018  
 overlap\_save\_filterbank\_t  
     MHAOvlFilter::overlap\_save\_filterbank\_t,  
         1021  
 overlapadd, 156  
 overlapadd.cpp, 1693  
 overlapadd.hh, 1693  
 overlapadd::overlapadd\_if\_t, 1326  
     ~overlapadd\_if\_t, 1328  
     algo, 1330  
     cf\_in, 1330  
     cf\_out, 1330  
     nfft, 1329  
     nwnd, 1329  
     overlapadd\_if\_t, 1327  
     plugloader, 1330  
     postscale, 1330  
     prepare, 1328  
     prescale, 1330  
     process, 1328  
     release, 1328  
     setlock, 1328  
     strict\_window\_ratio, 1329  
     update, 1328  
     window, 1329  
     wndexp, 1329  
     wndpos, 1329  
     zerowindow, 1329

overlapadd::overlapadd\_t, 1331  
  ~overlapadd\_t, 1331  
  calc\_out, 1333  
  fft, 1332  
  n\_pad1, 1333  
  n\_pad2, 1334  
  n\_zero, 1333  
  out\_buf, 1333  
  overlapadd\_t, 1331  
  postwnd, 1333  
  prewnd, 1332  
  spec2wave, 1332  
  spec\_in, 1333  
  wave2spec, 1332  
  wave2spec\_apply\_window, 1332  
  wave2spec\_compute\_fft, 1332  
  wave2spec\_hop\_forward, 1332  
  wave\_in1, 1333  
  wave\_out1, 1333  
  write\_buf, 1333  
overlapadd\_if\_t  
  overlapadd::overlapadd\_if\_t, 1327  
  smoothgains\_bridge::overlapadd\_if\_t,  
    1450  
overlapadd\_t  
  overlapadd::overlapadd\_t, 1331  
overrun\_behavior  
  lsl2ac, 97  
  lsl2ac::lsl2ac\_t, 681  
ovltype  
  MHAOvlFilter::fftfb\_vars\_t, 1008  
oy  
  MHAFilter::adapt\_filter\_state\_t, 862  
  
P  
  gsc\_adaptive\_stage::gsc\_adaptive\_stage,  
    540  
p  
  acPooling\_wave\_config, 216  
  doasvm\_classification\_config, 422  
  pluginbrowser\_t, 1354  
p1  
  MHASignal::hilbert\_fftw\_t, 1213  
p2  
  MHASignal::hilbert\_fftw\_t, 1213  
p\_biased  
  acPooling\_wave\_config, 216  
p\_biased\_name  
  acPooling\_wave, 214  
p\_in  
  io\_alsa\_t, 580  
p\_max  
  acPooling\_wave\_config, 217  
  doasvm\_classification\_config, 422  
p\_name  
  acPooling\_wave, 214  
  doasvm\_classification, 420  
p\_out  
  io\_alsa\_t, 580  
p\_parser  
  acmon::ac\_monitor\_t, 206  
P\_Sum  
  rt\_nlms\_t, 1417  
pa22dbspl  
  MHASignal, 142  
pa2dbspl  
  MHASignal, 141, 142  
palInputLatency  
  MHAIOPortAudio::stream\_info\_t, 962  
pairings  
  level\_matching::level\_matching\_config\_t,  
    649  
paOutputLatency  
  MHAIOPortAudio::stream\_info\_t, 962  
params  
  smooth\_cepstrum::smooth\_cepstrum\_t,  
    1441  
parent  
  matlab\_wrapper::callback, 691  
  MHAParser::base\_t, 1039  
parent\_  
  MHAParser::mhapluginloader\_t, 1089  
parse  
  altplugs\_t, 302  
  io\_asterisk\_t, 599  
  io\_tcp\_t, 637  
  io\_wrapper, 640  
  MHAParser::base\_t, 1031, 1032  
  MHAParser::Split::splitted\_part\_t, 1182  
  plug\_wrapper, 1347  
  plug\_wrapperl, 1349  
  PluginLoader::fourway\_processor\_t, 1364  
  PluginLoader::mhapluginloader\_t, 1367  
parse\_1\_complex  
  mha\_parser.cpp, 1612  
parse\_1\_float  
  mha\_parser.cpp, 1610  
parser  
  io\_asterisk\_t, 599  
  io\_tcp\_t, 637  
  mhachain::plugs\_t, 845  
parser\_algos  
  altconfig\_t, 299

parser\_int\_dyn, 1334  
     parser\_int\_dyn, 1335  
     set\_max\_angle\_ind, 1335  
 parser\_plugs  
     altplugs\_t, 303  
 parser\_t  
     AuditoryProfile::parser\_t, 326  
     MHAParser::parser\_t, 1100  
 parserFriendlyName  
     MHAIOPortAudio, 110  
 parsername  
     latex\_doc\_t, 643  
 parserstate  
     fw\_t, 525  
 partitioned\_convolution\_t  
     MHAFilter::partitioned\_convolution\_t, 914  
 partitions  
     MHAFilter::transfer\_function\_t, 934  
     MHAFilter::transfer\_matrix\_t, 937  
 paSampleRate  
     MHAIOPortAudio::stream\_info\_t, 962  
 PASCALE  
     levelmeter.cpp, 1565  
 PATCH\_VAR  
     acConcat\_wave.cpp, 1520  
     acPooling\_wave.cpp, 1521  
     acSteer.cpp, 1524  
     acTransform\_wave.cpp, 1524  
     adaptive\_feedback\_canceller.cpp, 1525  
     doasvm\_classification.cpp, 1542  
     doasvm\_feature\_extraction.cpp, 1543  
     level\_matching.cpp, 1564  
     lpc.cpp, 1566  
     lpc\_bl\_predictor.cpp, 1567  
     lpc\_burg-lattice.cpp, 1568  
     smooth\_cepstrum.cpp, 1698  
     steerbf.cpp, 1705  
 patchbay  
     ac2lsl::ac2lsl\_t, 165  
     ac2osc\_t, 184  
     ac2wave\_if\_t, 188  
     acConcat\_wave, 202  
     acmon::acmon\_t, 210  
     acPooling\_wave, 214  
     acsave::acsave\_t, 222  
     acSteer, 231  
     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, 317  
     AuditoryProfile::parser\_t::fmap\_t, 330  
     calibrator\_t, 340  
     coherence::cohflt\_if\_t, 348  
     complex\_scale\_channel\_t, 363  
     cpupload::cpupload\_if\_t, 368  
     db\_if\_t, 370  
     dc::dc\_if\_t, 382  
     dc\_simple::dc\_if\_t, 396  
     delay::interface\_t, 408  
     delaysum::delaysum\_wave\_if\_t, 411  
     delaysum\_spec::delaysum\_spec\_if\_t, 416  
     doasvm\_classification, 420  
     doasvm\_feature\_extraction, 425  
     double2acvar::double2acvar\_t, 432  
     dropgen\_t, 435  
     DynComp::dc\_afterburn\_t, 451  
     equalize::freqgains\_t, 462  
     example3\_t, 473  
     example4\_t, 477  
     example6\_t, 480  
     fader\_if\_t, 484  
     fader\_wave::fader\_wave\_if\_t, 487  
     fftfbpow::fftfbpow\_interface\_t, 493  
     fftfilter::interface\_t, 499  
     fftfilterbank::fftfb\_interface\_t, 502  
     fshift::fshift\_t, 510  
     fshift\_hilbert::frequency\_translator\_t, 513  
     fw\_t, 527  
     gain::gain\_if\_t, 531  
     gsc\_adaptive\_stage::gsc\_adaptive\_stage\_if, 544  
     gtfb\_analyzer::gtfb\_analyzer\_t, 551  
     gtfb\_simd\_t, 559  
     io\_alsa\_t, 580  
     io\_parser\_t, 617  
     level\_matching::level\_matching\_t, 653  
     levelmeter\_t, 656  
     lpc, 660  
     lpc\_bl\_predictor, 663  
     lpc\_burglattice, 669  
     lsl2ac::lsl2ac\_t, 681  
     matlab\_wrapper::matlab\_wrapper\_t, 698  
     matrixmixer::matmix\_t, 712  
     mconv::MConv, 716  
     mhachain::chain\_base\_t, 841  
     MHAIOJack::io\_jack\_t, 943  
     MHAIOJackdb::io\_jack\_t, 951  
     MHAIOPortAudio::io\_portaudio\_t, 960

MHAPlugin\_Split::split\_t, 1176  
multibandcompressor::interface\_t, 1302  
nlms\_t, 1308  
noise\_psd\_estimator::noise\_psd\_estimator\_if\_t, ac\_proc::interface\_t, 198  
    1310  
noise\_t, 1316  
osc2ac\_t, 1319  
plingloing::if\_t, 1338  
plugin\_interface\_t, 1352  
plugins::hoertech::acrec::acrec\_t, 1377  
rmslevel::rmslevel\_if\_t, 1390  
rohBeam::rohBeam, 1401  
route::interface\_t, 1410  
sine\_t, 1433  
smooth\_cepstrum::smooth\_cepstrum\_if\_t,  
    1439  
smoothgains\_bridge::overlapadd\_if\_t,  
    1451  
softclip\_t, 1456  
steerbf, 1472  
testplugin::ac\_parser\_t, 1478  
testplugin::if\_t, 1483  
wavrec\_t, 1500  
windnoise::if\_t, 1511  
windowselector\_t, 1517  
path  
    addsndfile::addsndfile\_if\_t, 252  
pcm  
    alsa\_base\_t, 288  
pcm\_format  
    alsa\_t< T >, 294  
pcmlink  
    io\_alsa\_t, 580  
peak  
    levelmeter\_t, 656  
    MHASignal::loop\_wavefragment\_t, 1218  
peaklevel  
    calibrator\_variables\_t, 341  
    mha\_channel\_info\_t, 741  
    MHASignal::async\_rmslevel\_t, 1197  
peek\_config  
    MHAPlugin::config\_t< runtime\_cfg\_t >,  
        1145  
peer\_addr  
    MHA\_TCP::Connection, 807  
peer\_address  
    io\_asterisk\_parser\_t, 592  
    io\_tcp\_parser\_t, 628  
peer\_port  
    io\_asterisk\_parser\_t, 592  
    io\_tcp\_parser\_t, 628  
period  
    droptect\_t, 439  
permute  
pfragmentsize  
    fw\_vars\_t, 529  
PH1Debug  
    noise\_psd\_estimator::noise\_psd\_estimator\_t,  
        1313  
PH1mean  
    noise\_psd\_estimator::noise\_psd\_estimator\_t,  
        1312  
phase  
    cpupload::cpupload\_cfg\_t, 365  
    MHASignal::minphase\_t, 1234  
phase\_correction  
    MHAFilter::gamma\_flt\_t, 890  
phase\_div\_2pi  
    sine\_t, 1433  
phase\_gains  
    MHASignal::subsample\_delay\_t, 1254  
phase\_increment\_div\_2pi  
    sine\_cfg\_t, 1430  
phasemode  
    fshift\_hilbert::frequency\_translator\_t, 514  
phasemode  
    MHAOvIFilter::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, 529  
pink  
    speechnoise\_t, 1468  
pipe  
    MHA\_TCP::Async\_Notify, 795  
pitch  
    plingloing::if\_t, 1338  
pitch\_  
    plingloing::plingloing\_t, 1342  
pitch\_set\_first  
    smooth\_cepstrum::smooth\_cepstrum\_t,  
        1445  
pitch\_set\_last  
    smooth\_cepstrum::smooth\_cepstrum\_t,  
        1445

plan\_spec2analytic  
     fshift\_hilbert::hilbert\_shifter\_t, 517  
 plateau  
     MHAOvIFilter::ffftfb\_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, 1693  
 plingploing::if\_t, 1336  
     bassmod, 1339  
     bassperiod, 1340  
     bpm, 1339  
     fun1\_key, 1338  
     fun1\_range, 1339  
     fun2\_key, 1339  
     fun2\_range, 1339  
     if\_t, 1337  
     level, 1338  
     maxlen, 1339  
     minlen, 1339  
     patchbay, 1338  
     pitch, 1338  
     prepare, 1338  
     process, 1338  
     update, 1338  
 plingploing::plingploing\_t, 1340  
     alph, 1344  
     bass, 1342  
     bassmod\_, 1344  
     bassperiod\_, 1344  
     bt, 1342  
     cf, 1342  
     dist, 1343  
     dist1, 1343  
     dur\_, 1342  
     freq, 1343  
     fun1, 1343  
     fun1\_key, 1343  
     fun1\_range, 1343  
     fun2, 1343  
     fun2\_key, 1343  
     fun2\_range, 1343  
     hann1, 1344  
     hann2, 1344  
     len, 1342  
     level, 1344  
     maxlen\_, 1342  
     minlen\_, 1342  
     pitch\_, 1342  
     plingploing\_t, 1341  
     process, 1341  
     rms, 1344  
     t, 1342  
 plingploing\_t  
     plingploing::plingploing\_t, 1341  
 plug  
     ac\_proc::interface\_t, 198  
     analysispath\_if\_t, 312  
     gtfb\_simple\_t, 569  
     matlab\_wrapper::matlab\_wrapper\_t, 698  
     MHAParser::mhapluginloader\_t, 1089  
     MHAPlugIn\_Split::splitted\_part\_t, 1183  
     multibandcompressor::interface\_t, 1302  
     testplugin::if\_t, 1483  
 plug\_level  
     multibandcompressor::plugin\_signals\_t,  
         1304  
 plug\_output  
     multibandcompressor::plugin\_signals\_t,  
         1304  
 plug\_sigs  
     multibandcompressor::interface\_t, 1302  
 plug\_t, 1345  
     ~plug\_t, 1345  
     get\_ac, 1346  
     get\_handle, 1346  
     get\_process\_spec, 1346  
     get\_process\_wave, 1345  
     plug\_t, 1345  
 plug\_wrapper, 1346  
     ~plug\_wrapper, 1347  
     get\_categories, 1347  
     get\_documentation, 1347  
     has\_parser, 1347  
     has\_process, 1348  
     parse, 1347  
     plug\_wrapper, 1347  
 plug\_wrapperl, 1348  
     ~plug\_wrapperl, 1349  
     get\_categories, 1349  
     get\_documentation, 1349  
     has\_parser, 1349  
     has\_process, 1350  
     parse, 1349  
     plug\_wrapperl, 1349  
 plugin\_categories

io\_lib\_t, 612  
PluginLoader::mhaplugloader\_t, 1372  
plugin\_documentation  
  io\_lib\_t, 612  
  PluginLoader::mhaplugloader\_t, 1372  
plugin\_extension  
  pluginbrowser\_t, 1354  
plugin\_interface\_t, 1350  
  factor, 1352  
  patchbay, 1352  
  plugin\_interface\_t, 1351  
  prepare, 1351  
  process, 1351  
  scale\_ch, 1352  
  update\_cfg, 1352  
plugin\_macro  
  latex\_doc\_t, 644  
plugin\_paths  
  fw\_t, 526  
plugin\_signals\_t  
  multibandcompressor::plugin\_signals\_t,  
    1303  
plugin\_t  
  MHAPlugin::plugin\_t< runtime\_cfg\_t >,  
    1149  
pluginbrowser.cpp, 1694  
pluginbrowser.h, 1694  
pluginbrowser\_t, 1352  
  add\_plugin, 1353  
  add\_plugins, 1353  
  clear\_plugins, 1353  
  get\_paths, 1353  
  get\_plugins, 1354  
  library\_paths, 1354  
  p, 1354  
  plugin\_extension, 1354  
  pluginbrowser\_t, 1353  
  plugins, 1354  
  scan\_plugin, 1353  
  scan\_plugins, 1353  
plugindescription\_t, 1355  
  categories, 1355  
  documentation, 1355  
  fullname, 1355  
  name, 1355  
  queries, 1356  
  query\_cmds, 1356  
  spec2spec, 1356  
  spec2wave, 1356  
  wave2spec, 1355  
  wave2wave, 1355  
pluginlib\_t, 1356  
  ~pluginlib\_t, 1357  
  pluginlib\_t, 1357  
  resolve, 1358  
PluginLoader, 157  
  mhaconfig\_compare, 157  
  mhastrdomain, 157  
PluginLoader::config\_file\_splitter\_t, 1358  
  config\_file\_splitter\_t, 1359  
  configfile, 1360  
  configname, 1360  
  get\_configfile, 1360  
  get\_configname, 1359  
  get\_libname, 1359  
  get\_origname, 1359  
  libname, 1360  
  origname, 1360  
PluginLoader::fourway\_processor\_t, 1361  
  ~fourway\_processor\_t, 1362  
  parse, 1364  
  prepare, 1363  
  process, 1362, 1363  
  release, 1364  
PluginLoader::mhaplugloader\_t, 1364  
  ~mhaplugloader\_t, 1367  
  ac, 1370  
  b\_check\_version, 1372  
  b\_is\_prepared, 1372  
  cf\_input, 1371  
  cf\_output, 1372  
  get\_categories, 1369  
  get\_documentation, 1369  
  getfullname, 1368  
  has\_parser, 1367  
  has\_process, 1367  
  input\_domain, 1367  
  is\_prepared, 1369  
  lib\_data, 1370  
  lib\_err, 1370  
  lib\_handle, 1370  
  mha\_test\_struct\_size, 1369  
  MHADestroy\_cb, 1370  
  MHAGetVersion\_cb, 1370  
  MHAInit\_cb, 1370  
  mhaplugloader\_t, 1366  
  MHAPrepare\_cb, 1370  
  MHAProc\_spec2spec\_cb, 1371  
  MHAProc\_spec2wave\_cb, 1371  
  MHAProc\_wave2spec\_cb, 1371  
  MHAProc\_wave2wave\_cb, 1371  
  MHARelease\_cb, 1370

MHASet\_cb, 1371  
 MHASetcpp\_cb, 1371  
 MHALError\_cb, 1371  
 output\_domain, 1367  
 parse, 1367  
 plugin\_categories, 1372  
 plugin\_documentation, 1372  
 prepare, 1367  
 process, 1368  
 release, 1368  
 resolve\_and\_init, 1369  
 test\_error, 1369  
 test\_version, 1369  
 pluginloader\_t, 1372  
 ~pluginloader\_t, 1373  
 pluginloader\_t, 1373  
 plugins, 158  
 fw\_t, 526  
 pluginbrowser\_t, 1354  
 plugins::hoertech, 158  
 plugins::hoertech::acrec, 158  
 to\_iso8601, 158  
 plugins::hoertech::acrec::acrec\_t, 1373  
 ac, 1378  
 acrec\_t, 1375  
 cv, 1377  
 fifolen, 1377  
 minwrite, 1377  
 patchbay, 1377  
 prefix, 1377  
 prepare, 1376  
 process, 1375  
 record, 1377  
 release, 1376  
 start\_new\_session, 1376  
 use\_date, 1377  
 varname, 1377  
 plugins::hoertech::acrec::acwriter\_t, 1378  
 ~acwriter\_t, 1380  
 active, 1382  
 acwriter\_t, 1380  
 close\_session, 1382  
 create\_datafile, 1381  
 disk\_write\_threshold\_min\_num\_samples,  
     1382  
 diskbuffer, 1383  
 exit\_request, 1381  
 fifo, 1382  
 get\_varname, 1381  
 is\_complex, 1383  
 is\_num\_channels\_known, 1383  
 num\_channels, 1383  
 outfile, 1383  
 output\_type, 1379  
 process, 1380  
 varname, 1383  
 write\_thread, 1381  
 writethread, 1382  
 plugloader  
 bbcalib\_interface\_t, 335  
 db\_if\_t, 371  
 db\_t, 372  
 dbasync\_native::db\_if\_t, 375  
 dbasync\_native::dbasync\_t, 378  
 MHAPlugin\_Resampling::resampling\_if\_t,  
     1155  
 MHAPlugin\_Resampling::resampling\_t,  
     1158  
 overlapadd::overlapadd\_if\_t, 1330  
 smoothgains\_bridge::overlapadd\_if\_t,  
     1451  
 plugname  
 latex\_doc\_t, 643  
 MHAParser::mhaplugloader\_t, 1090  
 plugname\_name\_  
 MHAParser::mhaplugloader\_t, 1090  
 plugs  
 altplugs\_t, 304  
 plugs\_t  
 mhachain::plugs\_t, 843  
 pmode  
 audiometerbackend::audiometer\_if\_t, 317  
 calibrator\_runtime\_layer\_t, 338  
 poll  
 mha\_rt\_fifo\_t< T >, 788  
 poll\_1  
 mha\_rt\_fifo\_t< T >, 788  
 poll\_config  
 MHAParser::config\_t< runtime\_cfg\_t >,  
     1145  
 poll\_latest\_value\_and\_reinsert  
 double2acvar::double2acvar\_t, 431  
 polyphase\_resampling\_t  
 MHAFilter::polyphase\_resampling\_t, 921  
 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, 217

pooling\_type  
    acPooling\_wave, 213

pooling\_wndlen  
    acPooling\_wave, 213

port  
    ac2osc\_t, 183  
    MHA\_TCP::Server, 814  
    MHAJack::port\_t, 983  
    mhaserver\_t, 1193  
    osc2ac\_t, 1319

port\_t  
    MHAJack::port\_t, 981, 982

portaudio\_callback  
    MHAIOPortAudio.cpp, 1676, 1677  
    MHAIOPortAudio::io\_portaudio\_t, 958

portaudio\_stream  
    MHAIOPortAudio::io\_portaudio\_t, 960

portnames\_in  
    MHAIOJack::io\_jack\_t, 942  
    MHAIOJackdb::io\_jack\_t, 949

portnames\_out  
    MHAIOJack::io\_jack\_t, 942  
    MHAIOJackdb::io\_jack\_t, 949

ports\_in\_all  
    MHAIOJack::io\_jack\_t, 942  
    MHAIOJackdb::io\_jack\_t, 950

ports\_in\_physical  
    MHAIOJack::io\_jack\_t, 942  
    MHAIOJackdb::io\_jack\_t, 950

ports\_out\_all  
    MHAIOJack::io\_jack\_t, 942  
    MHAIOJackdb::io\_jack\_t, 950

ports\_out\_physical  
    MHAIOJack::io\_jack\_t, 942  
    MHAIOJackdb::io\_jack\_t, 950

ports\_parser  
    MHAIOJack::io\_jack\_t, 943  
    MHAIOJackdb::io\_jack\_t, 950

pos  
    addsndfile::level\_adapt\_t, 256  
    audiometerbackend::level\_adapt\_t, 319  
    cfg\_t, 346  
    fader\_wave::level\_adapt\_t, 489  
    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, 1169

posixthreads  
    split.cpp, 1705

post\_plugin  
    gtfb\_simple\_rt\_t, 562

post\_trigger\_read\_line  
    mha\_tcp::server\_t, 819

postfilter  
    rohBeam::rohConfig, 1404

postscale  
    overlapadd::overlapadd\_if\_t, 1330

postwindow  
    spec2wave\_t, 1465

postwnd  
    overlapadd::overlapadd\_t, 1333

power  
    MHASignal::waveform\_t, 1269

powSpec  
    smooth\_cepstrum::smooth\_cepstrum\_t,  
        1443

powspec  
    MHASignal::waveform\_t, 1270  
    windnoise::cfg\_t, 1508

POWSPEC\_FACTOR  
    noise\_psd\_estimator.cpp, 1692

pre\_plugin  
    gtfb\_simple\_rt\_t, 562

prefix  
    plugins::hoertech::acrec::acrec\_t, 1377  
    wavrec\_t, 1500

prefix\_  
    MHAParser::mhapluginloader\_t, 1090

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, 212  
    acsave::acsave\_t, 220  
    acSteer, 229  
    acTransform\_wave, 235  
    adaptive\_feedback\_canceller, 242  
    addsndfile::addsndfile\_if\_t, 251  
    adm\_if\_t, 272  
    altconfig\_t, 297

altplugs\_t, 301  
 analysispath\_if\_t, 311  
 attenuate20\_t, 314  
 audiometerbackend::audiometer\_if\_t, 316  
 bbcalib\_interface\_t, 334  
 calibrator\_t, 339  
 coherence::cohflt\_if\_t, 348  
 combc\_if\_t, 356  
 complex\_scale\_channel\_t, 362  
 cpupload::cpupload\_if\_t, 367  
 db\_if\_t, 370  
 dbasync\_native::db\_if\_t, 374  
 dc::dc\_if\_t, 381  
 dc\_simple::dc\_if\_t, 393  
 delay::interface\_t, 408  
 delaysum::delaysum\_wave\_if\_t, 410  
 delaysum\_spec::delaysum\_spec\_if\_t, 415  
 doasvm\_classification, 419  
 doasvm\_feature\_extraction, 424  
 dropgen\_t, 434  
 droptect\_t, 437  
 ds\_t, 441  
 equalize::freqgains\_t, 461  
 example1\_t, 464  
 example2\_t, 467  
 example3\_t, 471  
 example4\_t, 475  
 example6\_t, 479  
 example7\_t, 482  
 fader\_if\_t, 484  
 fader\_wave::fader\_wave\_if\_t, 486  
 fftfbpow::fftfbpow\_interface\_t, 491  
 fftfilter::interface\_t, 498  
 fftfilterbank::fftfb\_interface\_t, 501  
 fshift::fshift\_t, 509  
 fshift\_hilbert::frequency\_translator\_t, 513  
 fw\_t, 522  
 gain::gain\_if\_t, 531  
 gsc\_adaptive\_stage::gsc\_adaptive\_stage\_if, 543  
 gtfb\_analyzer::gtfb\_analyzer\_t, 551  
 gtfb\_simd\_t, 559  
 gtfb\_simple\_t, 568  
 identity\_t, 573  
 io\_alsa\_t, 577, 578  
 io\_asterisk\_sound\_t, 594  
 io\_asterisk\_t, 598  
 io\_file\_t, 603  
 io\_lib\_t, 609  
 io\_parser\_t, 614  
 io\_tcp\_sound\_t, 631  
 io\_tcp\_t, 636  
 level\_matching::level\_matching\_t, 652  
 levelmeter\_t, 655  
 lpc, 658  
 lpc\_bl\_predictor, 662  
 lpc\_burglattice, 668  
 lsl2ac::lsl2ac\_t, 679  
 matlab\_wrapper::matlab\_wrapper\_t, 696  
 matlab\_wrapper::matlab\_wrapper\_t::wrapped\_plugin\_t, 703  
 matrixmixer::matmix\_t, 711  
 mconv::MConv, 715  
 mhachain::chain\_base\_t, 840  
 mhachain::plugs\_t, 844  
 MHAIOJack::io\_jack\_t, 939  
 MHAIOJackdb::io\_jack\_t, 946  
 MHAJack::client\_t, 973  
 MHAParser::mhapluginloader\_t, 1088  
 MHAPlugIn::plugin\_t< runtime\_cfg\_t >, 1149  
 MHAPlugIn\_Resampling::resampling\_if\_t, 1154  
 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, 1309  
 noise\_t, 1315  
 osc2ac\_t, 1318  
 overlapadd::overlapadd\_if\_t, 1328  
 plingploing::if\_t, 1338  
 plugin\_interface\_t, 1351  
 PluginLoader::fourway\_processor\_t, 1363  
 PluginLoader::mhapluginloader\_t, 1367  
 plugins::hoertech::acrec::acrec\_t, 1376  
 rmslevel::rmslevel\_if\_t, 1389  
 rohBeam::rohBeam, 1397  
 route::interface\_t, 1409  
 save\_spec\_t, 1419  
 save\_wave\_t, 1421  
 shadowfilter\_begin::shadowfilter\_begin\_t, 1424  
 shadowfilter\_end::shadowfilter\_end\_t, 1428  
 sine\_t, 1432  
 smooth\_cepstrum::smooth\_cepstrum\_if\_t, 1436  
 smoothgains\_bridge::overlapadd\_if\_t, 1450  
 softclip\_t, 1455

spec2wave\_if\_t, 1462  
steerbf, 1471  
testplugin::if\_t, 1482  
us\_t, 1486  
wave2spec\_if\_t, 1489  
wavrec\_t, 1499  
windnoise::if\_t, 1510  
prepare\_  
    ac\_mul\_t, 193  
    double2acvar::double2acvar\_t, 431  
    iirfilter\_t, 575  
    MHAPlugin::plugin\_t< runtime\_cfg\_t >, 1151  
    MHAPlugin\_Split::split\_t, 1175  
    proc\_counter\_t, 1385  
prepare\_impl  
    MHAJack::client\_t, 976  
prepare\_vars  
    fw\_t, 524  
PREPARED  
    MHA\_TCP::Thread, 826  
prepared  
    ac2wave\_if\_t, 188  
    altplugs\_t, 305  
    calibrator\_t, 340  
    dc\_simple::dc\_if\_t, 396  
    example3\_t, 472  
    example4\_t, 477  
    fader\_wave::fader\_wave\_if\_t, 487  
    fftfilterbank::fftfb\_interface\_t, 502  
    gtfb\_analyzer::gtfb\_analyzer\_t, 552  
    gtfb\_simd\_t, 559  
    mhachain::plugs\_t, 844  
    rohBeam::rohBeam, 1401  
    route::interface\_t, 1411  
    smooth\_cepstrum::smooth\_cepstrum\_if\_t, 1439  
prereadaccess  
    MHAParser::base\_t, 1038  
prescale  
    overlapadd::overlapadd\_if\_t, 1330  
preset  
    dc::dc\_vars\_t, 390  
    dc\_simple::dc\_if\_t, 395  
prestages  
    gtfb\_simple\_t, 569  
prewnd  
    overlapadd::overlapadd\_t, 1332  
print\_ac  
    analysemhaplugin.cpp, 1530  
print\_plugin\_references  
    generatemhaplugin.cpp, 1552  
prior\_q  
    smooth\_cepstrum::smooth\_cepstrum\_if\_t, 1438  
    smooth\_cepstrum::smooth\_params, 1448  
priorFact  
    noise\_psd\_estimator::noise\_psd\_estimator\_t, 1313  
    smooth\_cepstrum::smooth\_cepstrum\_t, 1445  
priority  
    analysepath\_t, 309  
    analysispath\_if\_t, 312  
    dbasync\_native::dbasync\_t, 378  
    io\_alsa\_t, 580  
    MHAPlugin\_Split::posix\_threads\_t, 1171  
prob\_bias  
    acPooling\_wave, 214  
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, 1453  
proc\_2  
    smoothgains\_bridge::smoothspec\_wrap\_t, 1453  
proc\_cnt  
    mhachain::plugs\_t, 846  
proc\_counter.cpp, 1694  
proc\_counter\_t, 1384  
    ~proc\_counter\_t, 1385  
    ac, 1386  
    prepare\_, 1385  
    proc\_counter\_t, 1385  
    process, 1385  
    release\_, 1385  
proc\_err  
    io\_asterisk\_fwcbs\_t, 584  
    io\_tcp\_fwcbs\_t, 621  
proc\_error  
    fw\_t, 527  
proc\_error\_string  
    fw\_t, 527  
proc\_event  
    io\_alsa\_t, 579  
    io\_asterisk\_fwcbs\_t, 584  
    io\_file\_t, 605  
    io\_parser\_t, 615

io\_tcp\_fwcb\_t, 620  
 MHAIOJackdb::io\_jack\_t, 948  
 MHAIOPortAudio::io\_portaudio\_t, 959  
 MHAJack::client\_t, 978  
 proc\_handle  
     io\_alsa\_t, 579  
     io\_asterisk\_fwcb\_t, 584  
     io\_file\_t, 605  
     io\_parser\_t, 615  
     io\_tcp\_fwcb\_t, 621  
     MHAIOJackdb::io\_jack\_t, 948  
     MHAIOPortAudio::io\_portaudio\_t, 959  
     MHAJack::client\_t, 978  
 proc\_lib  
     fw\_t, 526  
 proc\_name  
     fw\_t, 525  
 proc\_ramp  
     altplugs\_t, 303  
 proc\_thread  
     io\_alsa\_t, 579  
 proc\_wave  
     doasvm\_feature\_extraction\_config, 428  
 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, 212  
     acPooling\_wave\_config, 216  
     acsave::acsave\_t, 220, 221  
     acSteer, 229  
     acTransform\_wave, 234  
     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, 314  
     audiometerbackend::audiometer\_if\_t, 316  
     bbcilib\_interface\_t, 334  
     calibrator\_runtime\_layer\_t, 336  
     calibrator\_t, 339  
     cfg\_t, 345  
     coherence::cohflt\_if\_t, 348  
     coherence::cohflt\_t, 350  
     combc\_if\_t, 356  
     combc\_t, 358  
     complex\_scale\_channel\_t, 362  
     cpupload::cpupload\_cfg\_t, 364  
     cpupload::cpupload\_if\_t, 367  
     db\_if\_t, 370  
     dbasync\_native::db\_if\_t, 374  
     dc::dc\_if\_t, 381  
     dc::dc\_t, 384  
     dc\_simple::dc\_if\_t, 394  
     dc\_simple::dc\_t, 397, 398  
     dc\_simple::level\_smoothen\_t, 405  
     delay::interface\_t, 408  
     delaysum::delaysum\_wave\_if\_t, 410  
     delaysum::delaysum\_wave\_t, 413  
     delaysum\_spec::delaysum\_spec\_if\_t, 415  
     delaysum\_spec::delaysum\_t, 417  
     doasvm\_classification, 419  
     doasvm\_classification\_config, 422  
     doasvm\_feature\_extraction, 424  
     doasvm\_feature\_extraction\_config, 427  
     double2acvar::double2acvar\_t, 431  
     dropgen\_t, 434  
     droptect\_t, 438  
     ds\_t, 441  
     equalize::freqgains\_t, 461  
     example1\_t, 464  
     example2\_t, 468  
     example3\_t, 472  
     example4\_t, 476  
     example5\_t, 478  
     example6\_t, 479  
     example7\_t, 482  
     fader\_if\_t, 484  
     fader\_wave::fader\_wave\_if\_t, 486  
     fftfbpow::fftfbpow\_interface\_t, 492  
     fftfilter::fftfilter\_t, 496  
     fftfilter::interface\_t, 498  
     fftfilterbank::fftfb\_interface\_t, 501, 502  
     fftfilterbank::fftfb\_plug\_t, 504  
     fshift::fshift\_config\_t, 506  
     fshift::fshift\_t, 509  
     fshift\_hilbert::frequency\_translator\_t, 512  
     fshift\_hilbert::hilbert\_shifter\_t, 516  
     fw\_t, 523

gain::gain\_if\_t, 530  
gsc\_adaptive\_stage::gsc\_adaptive\_stage, 535  
gsc\_adaptive\_stage::gsc\_adaptive\_stage\_if, 543  
gtfb\_analyzer::gtfb\_analyzer\_t, 551  
gtfb\_simd\_cfg\_t, 555  
gtfb\_simd\_t, 559  
gtfb\_simple\_t, 568  
identity\_t, 573  
iirfilter\_t, 575  
io\_alsa\_t, 578  
io\_asterisk\_fwc\_b\_t, 582  
io\_tcp\_fwc\_b\_t, 619  
level\_matching::level\_matching\_config\_t, 648, 649  
level\_matching::level\_matching\_t, 651, 652  
levelmeter\_t, 655  
lpc, 658  
lpc\_bl\_predictor, 662  
lpc\_bl\_predictor\_config, 664  
lpc\_burglattice, 668  
lpc\_burglattice\_config, 670  
lpc\_config, 673  
lsl2ac::cfg\_t, 676  
lsl2ac::lsl2ac\_t, 679  
matlab\_wrapper::matlab\_wrapper\_t, 695, 696  
matrixmixer::cfg\_t, 709  
matrixmixer::matmix\_t, 711, 712  
mconv::MConv, 715  
mha\_dbdbuf\_t< FIFO >, 748  
mhachain::chain\_base\_t, 839, 840  
mhachain::plugs\_t, 844  
MHAFilter::partitioned\_convolution\_t, 915  
MHAFilter::thirdoctave\_analyzer\_t, 931  
MHAParser::mhaplugloader\_t, 1088, 1089  
MHAParser\_Resampling::resampling\_if\_t, 1154  
MHAParser\_Resampling::resampling\_t, 1157  
MHAParser\_Split::domain\_handler\_t, 1164  
MHAParser\_Split::split\_t, 1175  
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, 1309  
noise\_psd\_estimator::noise\_psd\_estimator\_t, 1312  
noise\_t, 1315  
osc2ac\_t, 1318  
overlapadd::overlapadd\_if\_t, 1328  
plingploing::if\_t, 1338  
plingploing::plingploing\_t, 1341  
plugin\_interface\_t, 1351  
PluginLoader::fourway\_processor\_t, 1362, 1363  
PluginLoader::mhaplugloader\_t, 1368  
plugins::hoertech::acrec::acrec\_t, 1375  
plugins::hoertech::acrec::acwriter\_t, 1380  
proc\_counter\_t, 1385  
rmslevel::rmslevel\_if\_t, 1389  
rmslevel::rmslevel\_t, 1392  
rohBeam::rohBeam, 1397  
rohBeam::rohConfig, 1404  
route::interface\_t, 1410  
route::process\_t, 1412  
rt\_nlms\_t, 1415  
save\_spec\_t, 1419  
save\_wave\_t, 1421  
shadowfilter\_begin::cfg\_t, 1422  
shadowfilter\_begin::shadowfilter\_begin\_t, 1424  
shadowfilter\_end::cfg\_t, 1426  
shadowfilter\_end::shadowfilter\_end\_t, 1428  
sine\_t, 1432  
smooth\_cepstrum::smooth\_cepstrum\_if\_t, 1435  
smooth\_cepstrum::smooth\_cepstrum\_t, 1441  
smoothgains\_bridge::overlapadd\_if\_t, 1451  
softclip\_t, 1455  
softclipper\_t, 1457  
spec2wave\_if\_t, 1462  
spec2wave\_t, 1464  
steerbf, 1471  
steerbf\_config, 1473  
testplugin::if\_t, 1482  
us\_t, 1486  
wave2spec\_if\_t, 1489, 1490

wave2spec\_t, 1495  
 wavrec\_t, 1499  
 wavwriter\_t, 1502  
 windnoise::cfg\_t, 1506  
 windnoise::if\_t, 1511  
 process\_cc  
   ac\_mul\_t, 194  
 process\_cr  
   ac\_mul\_t, 194  
 process\_frame  
   io\_parser\_t, 615  
 process\_rc  
   ac\_mul\_t, 194  
 process\_rr  
   ac\_mul\_t, 194  
 process\_ss  
   matlab\_wrapper::matlab\_wrapper\_t::wrapped\_plugin\_t,  
     702  
 process\_sw  
   matlab\_wrapper::matlab\_wrapper\_t::wrapped\_plugin\_t,  
     703  
 process\_t  
   route::process\_t, 1412  
 process\_ws  
   matlab\_wrapper::matlab\_wrapper\_t::wrapped\_plugin\_t,  
     702  
 process\_ww  
   matlab\_wrapper::matlab\_wrapper\_t::wrapped\_plugin\_t,  
     701  
 processing\_done  
   MHAPlugin\_Split::posix\_threads\_t, 1171  
 ProcessMutex  
   analysepath\_t, 309  
 processor  
   MHAPlugin\_Split::domain\_handler\_t,  
     1165  
   MHAPlugin\_Split::thread\_platform\_t,  
     1187  
 prof\_algos  
   mhachain::plugs\_t, 845  
 prof\_cfg  
   mhachain::plugs\_t, 846  
 prof\_init  
   mhachain::plugs\_t, 846  
 prof\_load\_con  
   mhachain::plugs\_t, 847  
 prof\_prepare  
   mhachain::plugs\_t, 846  
 prof\_process  
   mhachain::plugs\_t, 846  
 prof\_process\_load  
   mhachain::plugs\_t, 846  
 prof\_process\_tt  
   mhachain::plugs\_t, 846  
 prof\_release  
   mhachain::plugs\_t, 846  
 prof\_tt\_con  
   mhachain::plugs\_t, 847  
 profiling  
   mhachain::plugs\_t, 845  
 prop\_type  
   rohBeam::rohBeam, 1399  
 propExport  
   rohBeam::rohBeam, 1402  
 provoke\_inner\_error  
   mha\_dbdbuf\_t< FIFO >, 747  
 provoke\_outer\_error  
   mha\_dbdbuf\_t< FIFO >, 748  
 PSD\_Lowpass  
   windnoise::cfg\_t, 1508  
 PSD\_val\_t  
   adaptive\_feedback\_canceller\_config, 247  
 psrate  
   fw\_vars\_t, 529  
 Psum  
   gsc\_adaptive\_stage::gsc\_adaptive\_stage,  
     540  
 Pu  
   adaptive\_feedback\_canceller\_config, 246  
   rt\_nlms\_t, 1416  
 publish\_ac\_variables  
   wave2spec\_t, 1495  
 pull\_samples\_discard  
   lsl2ac::save\_var\_t, 685  
 pull\_samples\_ignore  
   lsl2ac::save\_var\_t, 685  
 push  
   mha\_rt\_fifo\_t< T >, 788  
   MHASignal::stat\_t, 1251, 1252  
 push\_config  
   MHAPlugin::config\_t< runtime\_cfg\_t >,  
     1146  
 put\_signal  
   MHAPlugin\_Split::domain\_handler\_t,  
     1162  
 pVars  
   osc\_server\_t, 1321  
 pwinner\_out  
   MHAIOJackdb::io\_jack\_t, 951  
 q  
   noise\_psd\_estimator::noise\_psd\_estimator\_if\_t,  
     1310

q\_high  
    smooth\_cepstrum::smooth\_cepstrum\_t,  
        1442

q\_low  
    smooth\_cepstrum::smooth\_cepstrum\_t,  
        1442

quant  
    calibrator\_runtime\_layer\_t, 337

quantile  
    MHASignal, 147

quantizer\_t  
    MHASignal::quantizer\_t, 1235

queries  
    MHAParser::base\_t, 1038  
    plugindescription\_t, 1356

query\_addsubst  
    MHAParser::base\_t, 1036

query\_cmds  
    MHAParser::base\_t, 1036  
    plugindescription\_t, 1356

query\_dump  
    MHAParser::base\_t, 1033  
    MHAParser::monitor\_t, 1097  
    MHAParser::parser\_t, 1102

query\_entries  
    MHAParser::base\_t, 1033  
    MHAParser::parser\_t, 1102

query\_help  
    MHAParser::base\_t, 1036

query\_id  
    MHAParser::base\_t, 1035

query\_listids  
    MHAParser::base\_t, 1035  
    MHAParser::parser\_t, 1103

query\_map\_t  
    MHAParser, 126

query\_peak  
    levelmeter\_t, 656

query\_perm  
    MHAParser::base\_t, 1033  
    MHAParser::monitor\_t, 1097  
    MHAParser::variable\_t, 1115

query\_range  
    MHAParser::base\_t, 1034  
    MHAParser::kw\_t, 1074  
    MHAParser::range\_var\_t, 1106

query\_readfile  
    MHAParser::base\_t, 1034  
    MHAParser::parser\_t, 1102

query\_rms  
    levelmeter\_t, 655

query\_savefile  
    MHAParser::base\_t, 1035  
    MHAParser::parser\_t, 1103

query\_savefile\_compact  
    MHAParser::base\_t, 1035  
    MHAParser::parser\_t, 1103

query\_savemons  
    MHAParser::base\_t, 1035  
    MHAParser::parser\_t, 1103

query\_subst  
    MHAParser::base\_t, 1036

query\_t  
    MHAParser, 125

query\_type  
    MHAParser::base\_t, 1034  
    MHAParser::bool\_mon\_t, 1042  
    MHAParser::bool\_t, 1045  
    MHAParser::complex\_mon\_t, 1052  
    MHAParser::complex\_t, 1054  
    MHAParser::float\_mon\_t, 1059  
    MHAParser::float\_t, 1061  
    MHAParser::int\_mon\_t, 1064  
    MHAParser::int\_t, 1067  
    MHAParser::kw\_t, 1074  
    MHAParser::mcomplex\_mon\_t, 1076  
    MHAParser::mcomplex\_t, 1078  
    MHAParser::mfloat\_mon\_t, 1080  
    MHAParser::mfloat\_t, 1083  
    MHAParser::mint\_mon\_t, 1092  
    MHAParser::mint\_t, 1095  
    MHAParser::parser\_t, 1102  
    MHAParser::string\_mon\_t, 1110  
    MHAParser::string\_t, 1113  
    MHAParser::vcomplex\_mon\_t, 1117  
    MHAParser::vcomplex\_t, 1120  
    MHAParser::vfloat\_mon\_t, 1122  
    MHAParser::vfloat\_t, 1125  
    MHAParser::vint\_mon\_t, 1127  
    MHAParser::vint\_t, 1129  
    MHAParser::vstring\_mon\_t, 1132  
    MHAParser::vstring\_t, 1133

query\_val  
    MHAParser::base\_t, 1034  
    MHAParser::bool\_mon\_t, 1042  
    MHAParser::bool\_t, 1045  
    MHAParser::complex\_mon\_t, 1052  
    MHAParser::complex\_t, 1054  
    MHAParser::float\_mon\_t, 1058  
    MHAParser::float\_t, 1062  
    MHAParser::int\_mon\_t, 1064  
    MHAParser::int\_t, 1067

MHAParser::kw\_t, 1074  
 MHAParser::mcomplex\_mon\_t, 1076  
 MHAParser::mcomplex\_t, 1078  
 MHAParser::mfloat\_mon\_t, 1080  
 MHAParser::mfloat\_t, 1083  
 MHAParser::mint\_mon\_t, 1092  
 MHAParser::mint\_t, 1095  
 MHAParser::parser\_t, 1103  
 MHAParser::string\_mon\_t, 1110  
 MHAParser::string\_t, 1113  
 MHAParser::vcomplex\_mon\_t, 1117  
 MHAParser::vcomplex\_t, 1120  
 MHAParser::vfloat\_mon\_t, 1122  
 MHAParser::vfloat\_t, 1125  
 MHAParser::vint\_mon\_t, 1127  
 MHAParser::vint\_t, 1129  
 MHAParser::vstring\_mon\_t, 1131  
 MHAParser::vstring\_t, 1134  
 query\_version  
   MHAParser::base\_t, 1035  
 queue\_write  
   mha\_tcp::buffered\_socket\_t, 796  
 quit  
   fw\_t, 522

R

- AuditoryProfile::parser\_t, 327
- AuditoryProfile::profile\_t, 332
- lpc\_config, 674
- MHA\_TCP::OS\_EVENT\_TYPE, 810

r

- dropgen\_t, 435
- mha\_real\_test\_array\_t, 784

rad2smp

- Vector and matrix processing toolbox, 40

ramp\_a

- hanning\_ramps\_t, 571

ramp\_b

- hanning\_ramps\_t, 572

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, 317
- fader\_wave::fader\_wave\_if\_t, 487

spec2wave\_if\_t, 1463

ramps

- spec2wave\_t, 1465

rand\_dist

- cfg\_t, 346

random

- audiometerbackend::lnn3rdoct\_t, 322

random\_engine

- dropgen\_t, 435

range

- level\_matching::level\_matching\_config\_t, 649
- level\_matching::level\_matching\_t, 653
- Vector and matrix processing toolbox, 38

range\_var\_t

- MHAParser::range\_var\_t, 1106

ratio

- ds\_t, 441
- us\_t, 1487

raw\_p\_max\_name

- acTransform\_wave, 236
- acTransform\_wave\_config, 238

raw\_p\_name

- acPooling\_wave\_config, 216
- acTransform\_wave, 236
- acTransform\_wave\_config, 238

rb\_f\_t

- mha\_ruby.cpp, 1623

rcoefficients

- gtfb\_simd\_cfg\_t, 556

rdata

- mha\_audio\_t, 740
- MHASignal::matrix\_t, 1232

re

- mha\_complex\_t, 742

read

- alsa\_base\_t, 287
- alsa\_t< T >, 293
- mha\_drifter\_fifo\_t< T >, 755
- mha\_fifo\_if\_t< T >, 765
- mha\_fifo\_lw\_t< T >, 768
- mha\_fifo\_t< T >, 776
- MHAFilter::blockprocessing\_polyphase\_resampling\_t, 867
- MHAFilter::polyphase\_resampling\_t, 922
- MHAJack::port\_t, 982

read\_bytes

- MHA\_TCP::Connection, 805

read\_event

- MHA\_TCP::Connection, 806

read\_get\_cpu\_load

MHAIOJack::io\_jack\_t, 940  
MHAIOJackdb::io\_jack\_t, 947

read\_get\_scheduler  
  MHAIOJack::io\_jack\_t, 941  
  MHAIOJackdb::io\_jack\_t, 948

read\_get\_xruns  
  MHAIOJack::io\_jack\_t, 941  
  MHAIOJackdb::io\_jack\_t, 947

read\_levels  
  calibrator\_t, 340

read\_line  
  MHA\_TCP::Connection, 804

read\_modified  
  dc\_simple::dc\_if\_t, 394

read\_ptr  
  mha\_fifo\_t< T >, 779

readable\_frames  
  MHAFilter::polyphase\_resampling\_t, 922

readaccess  
  MHAParser::base\_t, 1038

reader\_started  
  mha\_drifter\_fifo\_t< T >, 758

reader\_xruns\_in\_succession  
  mha\_drifter\_fifo\_t< T >, 759

reader\_xruns\_since\_start  
  mha\_drifter\_fifo\_t< T >, 759

reader\_xruns\_total  
  mha\_drifter\_fifo\_t< T >, 758

real  
  MHASignal::matrix\_t, 1227–1230

rear\_channel  
  adm\_rtconfig\_t, 278

rear\_channels  
  adm\_if\_t, 273  
  adm\_rtconfig\_t, 278

rec\_frames  
  acsave::cfg\_t, 224

receive\_frame  
  lsl2ac::save\_var\_t, 685

receiver  
  MHAEvents::connector\_t< receiver\_t >, 854

reciprocal  
  Complex arithmetics in the openMHA, 67

reclen  
  acsave::acsave\_t, 221

recmode  
  acmon::acmon\_t, 210

reconnect\_inports  
  MHAIOJack::io\_jack\_t, 939  
  MHAIOJackdb::io\_jack\_t, 947

reconnect\_outports  
  MHAIOJack::io\_jack\_t, 940  
  MHAIOJackdb::io\_jack\_t, 947

record  
  plugins::hoertech::acrec::acrec\_t, 1377  
  wavrec\_t, 1500

rect  
  MHAOvlFilter::ShapeFun, 120  
  MHAWindow, 154

rect\_t  
  MHAWindow::rect\_t, 1296

refL  
  rohBeam, 160

refR  
  rohBeam, 160

register\_configuration\_variable  
  windnoise.cpp, 1709

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, 208  
  acPooling\_wave, 213  
  acsave::acsave\_t, 220  
  acSteer, 230  
  actransform\_wave, 236  
  adaptive\_feedback\_canceller, 242  
  addsndfile::addsndfile\_if\_t, 252  
  adm\_if\_t, 273  
  altconfig\_t, 297  
  altpicks\_t, 301  
  analysispath\_if\_t, 311  
  attenuate20\_t, 314  
  bbcilib\_interface\_t, 335  
  calibrator\_t, 339  
  coherence::cohfilt\_if\_t, 348  
  db\_if\_t, 370  
  dbasync\_native::db\_if\_t, 375  
  dc\_simple::dc\_if\_t, 393  
  delaysum::delaysum\_wave\_if\_t, 411  
  doasvm\_classification, 419  
  doasvm\_feature\_extraction, 425  
  dropgen\_t, 434  
  droptect\_t, 438  
  ds\_t, 441  
  example1\_t, 464  
  example2\_t, 468

example3\_t, 471  
 example4\_t, 476  
 example7\_t, 482  
 fader\_wave::fader\_wave\_if\_t, 486  
 fftfilterbank::fftfb\_interface\_t, 501  
 fshift::fshift\_t, 510  
 fshift\_hilbert::frequency\_translator\_t, 513  
 fw\_t, 522  
 gain::gain\_if\_t, 531  
 gsc\_adaptive\_stage::gsc\_adaptive\_stage\_ifrelease\_543  
 gtfb\_simple\_t, 568  
 identity\_t, 573  
 io\_alsa\_t, 577  
 io\_asterisk\_sound\_t, 594  
 io\_asterisk\_t, 598  
 io\_file\_t, 603  
 io\_lib\_t, 610  
 io\_parser\_t, 614  
 io\_tcp\_sound\_t, 631  
 io\_tcp\_t, 636  
 level\_matching::level\_matching\_t, 652  
 lpc, 659  
 lpc\_bl\_predictor, 662  
 lpc\_burglattice, 668  
 lsl2ac::lsl2ac\_t, 679  
 matlab\_wrapper::matlab\_wrapper\_t, 697  
 matlab\_wrapper::matlab\_wrapper\_t::wrapped\_phobj\_tt\_fifo\_t< T >, 789  
 703  
 mconv::MConv, 715  
 mhachain::chain\_base\_t, 840  
 mhachain::plugs\_t, 844  
 MHAIOJack::io\_jack\_t, 939  
 MHAIOJackdb::io\_jack\_t, 946  
 MHAJack::client\_t, 974  
 MHAParser::mhaplugloader\_t, 1088  
 MHAParser::mhaplugloader\_t< runtime\_cfg\_t >, 1150  
 1155  
 MHAParser::mhaplugloader\_t< runtime\_cfg\_t >, 1182  
 multibandcompressor::interface\_t, 1301  
 nlms\_t, 1306  
 osc2ac\_t, 1318  
 overlapadd::overlapadd\_if\_t, 1328  
 PluginLoader::fourway\_processor\_t, 1364  
 PluginLoader::mhaplugloader\_t, 1368  
 plugins::hoertech::acrec::acrec\_t, 1376  
 rohBeam::rohBeam, 1397  
 route::interface\_t, 1410  
 smooth\_cepstrum::smooth\_cepstrum\_if\_t, 1436  
 smoothgains\_bridge::overlapadd\_if\_t, 1450  
 spec2wave\_if\_t, 1462  
 steerbf, 1472  
 us\_t, 1486  
 wave2spec\_if\_t, 1489  
 wavrec\_t, 1499  
 windnoise::if\_t, 1511  
 ac\_mul\_t, 193  
 double2acvar::double2acvar\_t, 432  
 iirfilter\_t, 575  
 MHAParser::base\_t< runtime\_cfg\_t >, 1151  
 MHAParser::Split::split\_t, 1175  
 proc\_counter\_t, 1385  
 release\_mutex  
 mha\_fifo\_posix\_threads\_t, 771  
 mha\_fifo\_thread\_platform\_t, 782  
 remapping  
 windnoise::cfg\_t, 1507  
 remove  
 MHAUtils, 151  
 remove\_abandonned  
 mha\_rt\_fifo\_t< T >, 789  
 remove\_all  
 mha\_rt\_fifo\_t< T >, 789  
 remove\_all\_cfg  
 MHAParser::config\_t< runtime\_cfg\_t >, 1146  
 remove\_item  
 MHAParser::parser\_t, 1100, 1101  
 remove\_ref  
 algo\_comm\_t, 282  
 MHAKernel::algo\_comm\_class\_t, 987  
 remove\_var  
 algo\_comm\_t, 282  
 MHAKernel::algo\_comm\_class\_t, 987  
 repl\_list  
 MHAParser::base\_t, 1039  
 repl\_list\_t  
 MHAParser::base\_t, 1030  
 replace  
 MHAParser::base\_t::replace\_t, 1040  
 MHASignal::loop\_wavefragment\_t, 1218  
 replace\_  
 cfg\_t, 345  
 replace\_t  
 MHAParser::base\_t::replace\_t, 1040  
 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,  
        867  
resampling.cpp, 1694  
resampling\_factors  
    MHAFilter, 109  
resampling\_filter\_t  
    MHAFilter::resampling\_filter\_t, 925  
resampling\_if\_t  
    MHAParser::resampling\_if\_t,  
        1154  
resampling\_t  
    MHAParser::resampling\_t,  
        1156  
resamplingmode  
    addsndfile::addsndfile\_if\_t, 253  
reset  
    droptect\_t, 438  
    MHA\_TCP::Async\_Notify, 795  
    MHA\_TCP::Wakeups\_Event, 834  
reset\_state  
    MHAFilter::gamma\_filt\_t, 891  
resize  
    MHAFilter::iir\_filter\_t, 896  
resolution  
    acTransform\_wave\_config, 239  
resolve  
    dynamiclib\_t, 444  
    pluginlib\_t, 1358  
resolve\_and\_init  
    PluginLoader::mhaplugloader\_t, 1369  
resolve\_checked  
    dynamiclib\_t, 444  
restore\_state  
    altconfig\_t, 298  
result  
    cpupload::cpupload\_cfg\_t, 365  
resynthesis\_gain  
    gtfb\_simple\_t, 570  
    MHAFilter::gamma\_filt\_t, 892  
ret\_size  
    MHAParser::c\_ifc\_parser\_t, 1048  
return\_imag  
    fftfilterbank::fftfb\_interface\_t, 502  
return\_imag  
    fftfilterbank::fftfb\_plug\_t, 505  
return\_sig  
    audiometerbackend, 83  
return\_value  
    MHA\_TCP::Thread, 828  
return\_wave  
    wave2spec\_if\_t, 1491  
rewinding\_t,  
    MHAParser::c\_ifc\_parser\_t, 1048  
rewind  
    MHASignal::loop\_wavefragment\_t, 1220  
rho  
    adaptive\_feedback\_canceller, 242  
    nlms\_t, 1307  
ringbuffer  
    MHAFilter::polyphase\_resampling\_t, 924  
ringbuffer\_t  
    MHASignal::ringbuffer\_t, 1237  
rinputs  
    gtfb\_simd\_cfg\_t, 556  
rm\_parent\_on\_remove  
    MHAParser::base\_t, 1037  
rms  
    levelmeter\_t, 656  
    MHASignal::loop\_wavefragment\_t, 1218  
    plingploing::plingploing\_t, 1344  
rms\_limit40  
    MHASignal::loop\_wavefragment\_t, 1218  
rmsdb  
    example6\_t, 480  
rmslevel, 158  
    calibrator\_variables\_t, 342  
    dc::dc\_t, 385  
    HL, 159  
    MHASignal::async\_rmslevel\_t, 1197  
    SPL, 159  
    UNIT, 159  
    Vector and matrix processing toolbox, 53,  
        55  
rmslevel.cpp, 1694  
rmslevel::mon\_t, 1386  
    mon\_t, 1387  
    operator=, 1387  
    store, 1387  
rmslevel::rmslevel\_if\_t, 1388  
    name, 1390  
    patchbay, 1390  
    prepare, 1389  
    process, 1389  
    rmslevel\_if\_t, 1389

unit, 1390  
 update, 1389  
**rmslevel::rmslevel\_t**, 1390  
 ~rmslevel\_t, 1391  
 domain, 1393  
 fftlen, 1393  
 fill, 1392  
 freq\_offsets, 1393  
 insert, 1392  
 monitors, 1393  
 process, 1392  
**rmslevel\_t**, 1391  
 sum\_and\_fill, 1392  
 unit, 1393  
**rmslevel\_if\_t**  
 rmslevel::rmslevel\_if\_t, 1389  
**rmslevel\_t**  
 rmslevel::rmslevel\_t, 1391  
**rmslevelmeter**  
 transducers.cpp, 1707  
**rng**  
 audiometerbackend::Inn3rdoct\_t, 322  
 cfg\_t, 346  
**rohBeam**, 159  
 CONST\_C, 160  
 j0, 159  
 refL, 160  
 refR, 160  
 rohBeam::rohBeam, 1396  
 scalarify, 160  
 rohBeam.cpp, 1695  
 rohBeam.hh, 1695  
 NDEBUG, 1696  
 rohBeam::configOptions, 1393  
 alpha\_blocking\_XkXi, 1394  
 alpha\_blocking\_XkY, 1394  
 alpha\_postfilter, 1394  
 binaural\_type\_index, 1394  
 enable\_adaptive\_beam, 1394  
 rohBeam::rohBeam, 1395  
 ~rohBeam, 1396  
 beamExport, 1401  
 binaural\_type, 1400  
 compute\_beamW, 1398  
 compute\_delaycomp\_vec, 1398  
 compute\_diff2D, 1398  
 compute\_diff3D, 1398  
 compute\_head\_model\_alpha, 1397  
 compute\_head\_model\_mat, 1397  
 compute\_head\_model\_T, 1397  
 compute\_uncorr, 1398  
 compute\_wng, 1399  
 diag\_loading\_mu, 1400  
 enable\_adaptive\_beam, 1400  
 enable\_export, 1401  
 enable\_wng\_optimization, 1401  
 export\_beam\_design, 1399  
 get\_noise\_model\_func, 1399  
 head\_model\_sphere\_radius\_cm, 1400  
 intermic\_distance\_cm, 1400  
 mic\_azimuth\_degrees\_vec, 1400  
 noise\_field\_model, 1400  
 noise\_integrate\_hrtf, 1398  
 noiseFuncPtr, 1399  
 noiseModelExport, 1402  
 on\_model\_param\_valuechanged, 1399  
 patchbay, 1401  
 prepare, 1397  
 prepared, 1401  
 process, 1397  
 prop\_type, 1399  
 propExport, 1402  
 release, 1397  
 rohBeam, 1396  
 sampled\_hrir\_path, 1399  
 solve\_MVDR, 1398  
 source\_azimuth\_degrees, 1400  
 tau\_blocking\_XkXi\_ms, 1401  
 tau\_blocking\_XkY\_ms, 1401  
 tau\_postfilter\_ms, 1401  
 update\_cfg, 1397  
**rohBeam::rohConfig**, 1402  
 ~rohConfig, 1404  
 alpha\_blocking\_XkXi, 1406  
 alpha\_blocking\_XkY, 1406  
 alpha\_postfilter, 1406  
 beam1, 1406  
 beamA, 1406  
 beamW, 1406  
 binaural\_type\_index, 1405  
 blockSpec, 1406  
 blockXp, 1407  
 copyfixedbfoutput, 1405  
 corrLL, 1407  
 corrRR, 1407  
 corrXpXp, 1407  
 corrXpYf, 1407  
 corrZZ, 1407  
 delayComp, 1406  
 enable\_adaptive\_beam, 1405  
 freqResp, 1407  
 headModel, 1405

hhCorrXpXp, 1407  
in\_cfg, 1405  
init\_dynamic, 1404  
magResp, 1408  
maxLim, 1408  
minLim, 1408  
nchan\_block, 1405  
nextXpYf, 1407  
nfreq, 1405  
operator=, 1404  
out\_cfg, 1405  
outSpec, 1406  
phasereconstruction, 1404  
postfilter, 1404  
process, 1404  
rohConfig, 1403, 1404  
rohConfig  
    rohBeam::rohConfig, 1403, 1404  
root  
    mha\_rt\_fifo\_t< T >, 789  
rotated\_i  
    acTransform\_wave\_config, 239  
rotated\_p  
    acTransform\_wave\_config, 239  
rotated\_p\_max\_name  
    acTransform\_wave, 236  
rotated\_p\_name  
    acTransform\_wave, 236  
route, 160  
route.cpp, 1696  
route::interface\_t, 1408  
    algo, 1411  
    cfac, 1411  
    cfin, 1411  
    cfout, 1411  
    interface\_t, 1409  
    patchbay, 1410  
    prepare, 1409  
    prepared, 1411  
    process, 1410  
    release, 1410  
    route\_ac, 1410  
    route\_out, 1410  
    stopped, 1411  
    update, 1410  
route::process\_t, 1411  
    process, 1412  
    process\_t, 1412  
    sout, 1413  
    sout\_ac, 1413  
    wout, 1412  
    wout\_ac, 1413  
route\_ac  
    route::interface\_t, 1410  
route\_out  
    route::interface\_t, 1410  
rows  
    acsave::mat4head\_t, 224  
rstates  
    gtfb\_simd\_cfg\_t, 556  
rt\_nlms\_t, 1413  
    ~rt\_nlms\_t, 1414  
    ac, 1415  
    channels, 1415  
    F, 1415  
    frames, 1415  
    fu, 1416  
    fu\_previous, 1416  
    fuflt, 1416  
    insert, 1415  
    n\_no\_update\_, 1417  
    name\_d\_, 1417  
    name\_e\_, 1417  
    name\_u\_, 1417  
    no\_iter, 1417  
    ntaps, 1415  
    P\_Sum, 1417  
    process, 1415  
    Pu, 1416  
    rt\_nlms\_t, 1414  
    s\_E, 1417  
    U, 1416  
    Uflt, 1416  
    y\_previous, 1416  
rt\_process  
    analysepath\_t, 307  
rt\_strict  
    ac2lsl::ac2lsl\_t, 165  
    ac2osc\_t, 184  
rtcalibrator  
    transducers.cpp, 1707  
rtmem  
    ac2osc\_t, 184  
run  
    mha\_tcp::server\_t, 818  
    MHA\_TCP::Thread, 827  
    mhaserver\_t, 1192  
RUNNING  
    MHA\_TCP::Thread, 826  
runtime configuration, 4  
rval  
    MHAParser::expression\_t, 1057

s\_b  
 lpc\_bl\_predictor\_config, 666  
 lpc\_burglattice\_config, 672

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, 666  
 lpc\_burglattice\_config, 672

s\_file\_in  
 io\_file\_t, 606

s\_in  
 ac\_proc::interface\_t, 199  
 io\_asterisk\_sound\_t, 596  
 io\_file\_t, 606  
 io\_parser\_t, 616  
 io\_tcp\_sound\_t, 633  
 MHAIOPortAudio::io\_portaudio\_t, 958  
 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, 352  
 combc\_t, 359  
 fftfilterbank::fftfb\_plugin\_t, 505  
 gtfb\_analyzer::gtfb\_analyzer\_cfg\_t, 548  
 gtfb\_simd\_cfg\_t, 557  
 io\_file\_t, 606  
 io\_parser\_t, 616  
 MHAIOPortAudio::io\_portaudio\_t, 958  
 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, 1627  
 mha\_signal.hh, 1637

sample  
 lpc\_config, 674

sampled\_hrir\_path  
 rohBeam::rohBeam, 1399

samplerate  
 io\_asterisk\_sound\_t, 595  
 io\_file\_t, 604  
 io\_tcp\_sound\_t, 633  
 MHAIOPortAudio::io\_portaudio\_t, 958  
 MHAJack::client\_t, 978

samples\_AC  
 acConcat\_wave, 201

samplingrate  
 MHAOvLFilter::fftfb\_t, 1006

save\_m  
 acsave::save\_var\_t, 226

save\_mat4  
 acsave::save\_var\_t, 226

save\_spec.cpp, 1696

save\_spec\_t, 1418  
 basename, 1419  
 prepare, 1419  
 process, 1419  
 save\_spec\_t, 1418

save\_state  
 altconfig\_t, 298

save\_txt  
 acsave::save\_var\_t, 226

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, 684

save\_vars  
 acmon::acmon\_t, 209

save\_wave.cpp, 1696

save\_wave\_t, 1420  
 basename, 1421  
 prepare, 1421  
 process, 1421  
 save\_wave\_t, 1420

saveas\_mat4  
 MHASignal, 148, 149

sc  
 MHASignal::hilbert\_fftw\_t, 1214

spec2wave\_t, 1465  
scalarify  
    rohBeam, 160  
scale  
    cfg\_t, 345  
    delaysum\_spec::delaysum\_t, 417  
    example5\_t, 478  
    MHASignal, 144  
    MHASignal::fft\_t, 1211  
    MHASignal::spectrum\_t, 1248  
    MHASignal::waveform\_t, 1270  
scale\_ch  
    complex\_scale\_channel\_t, 363  
    example2\_t, 468  
    example3\_t, 472  
    example4\_t, 476  
    plugin\_interface\_t, 1352  
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, 532  
scan\_dir  
    addsndfile::addsndfile\_if\_t, 252  
scan\_plugin  
    pluginbrowser\_t, 1353  
scan\_plugins  
    pluginbrowser\_t, 1353  
scan\_syntax  
    ac\_mul\_t, 193  
scheduler  
    analysepath\_t, 309  
    dbasync\_native::dbasync\_t, 378  
    MHAPlugin\_Split::posix\_threads\_t, 1171  
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, 1316  
select\_plug  
    altconfig\_t, 299  
    altparts\_t, 304  
select\_source  
    MHAMultiSrc::base\_t, 992  
selectall  
    altconfig\_t, 299  
selected\_plug  
    altparts\_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, 812  
server  
    io\_asterisk\_t, 600  
    io\_tcp\_t, 638  
server\_fragsize  
    MHAIOJackdb::io\_jack\_t, 950  
server\_port\_open  
    io\_asterisk\_parser\_t, 591  
    io\_tcp\_parser\_t, 628  
server\_srate  
    MHAIOJackdb::io\_jack\_t, 950  
server\_start  
    osc\_server\_t, 1321  
server\_stop  
    osc\_server\_t, 1321  
server\_t  
    mha\_tcp::server\_t, 816  
servername  
    MHAIOJack::io\_jack\_t, 941  
    MHAIOJackdb::io\_jack\_t, 949  
serversocket  
    MHA\_TCP::Server, 814  
set  
    Complex arithmetics in the openMHA, 60, 61  
    MHA\_TCP::Async\_Notify, 795  
    testplugin::config\_parser\_t, 1479  
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, 863  
 set\_connected  
     io\_asterisk\_parser\_t, 589  
     io\_tcp\_parser\_t, 626  
 set\_entries  
     MHAParser::keyword\_list\_t, 1069  
 set\_errno  
     io\_asterisk\_fwcb\_t, 583  
     io\_tcp\_fwcb\_t, 619  
 set\_error  
     mha\_fifo\_lw\_t< T >, 769  
 set\_fb\_pars  
     DynComp::dc\_afterburn\_t, 450  
 set\_format  
     wavwriter\_t, 1502  
 set\_help  
     MHAParser::base\_t, 1036  
 set\_id\_string  
     MHAParser::parser\_t, 1103  
 set\_index  
     MHAParser::keyword\_list\_t, 1070  
 set\_input\_domain  
     MHAParser::base\_t, 1036  
     MHAParser::parser\_t, 1103  
     MHAParser::keyword\_list\_t, 1069  
     MHAParser::domain\_handler\_t, 1161  
 set\_input\_portnames  
     MHAJack::client\_t, 975  
 set\_level  
     addsndfile::addsndfile\_if\_t, 252  
     audiometerbackend::audiometer\_if\_t, 316  
     fader\_wave::fader\_wave\_if\_t, 487  
 set\_level\_db  
     MHASignal::loop\_wavefragment\_t, 1220  
 set\_level\_lin  
     MHASignal::loop\_wavefragment\_t, 1220  
 set\_local\_port  
     io\_asterisk\_parser\_t, 588  
     io\_tcp\_parser\_t, 624  
 set\_locate  
     MHAIQJackdb::io\_jack\_t, 948  
 set\_max\_angle\_ind  
     parser\_int\_dyn, 1335  
 set\_minabs  
     mha\_signal.cpp, 1627  
     mha\_signal.hh, 1637  
 set\_new\_peer  
     io\_asterisk\_parser\_t, 590  
     io\_tcp\_parser\_t, 626  
     set\_node\_id  
         MHAParser::base\_t, 1036  
     set\_output\_domain  
         MHAParser::base\_t, 1036  
         MHAParser::domain\_handler\_t, 1161  
     set\_output\_portnames  
         MHAJack::client\_t, 975  
     set\_parse\_cb  
         MHAParser::c\_ifc\_parser\_t, 1047  
     set\_range  
         MHAParser::kw\_t, 1073  
         MHAParser::range\_var\_t, 1106  
     set\_server\_port\_open  
         io\_asterisk\_parser\_t, 588  
         io\_tcp\_parser\_t, 625  
     set\_state  
         MHAFilter::complex\_bandpass\_t, 870  
         MHAFilter::iir\_ord1\_real\_t, 900  
     set\_tau  
         MHAFilter::o1flt\_lowpass\_t, 908  
         MHAFilter::o1flt\_maxtrack\_t, 910  
         MHAFilter::o1flt\_mintrack\_t, 912  
     set\_tau\_attack  
         MHAFilter::o1\_ar\_filter\_t, 904  
     set\_tau\_release  
         MHAFilter::o1\_ar\_filter\_t, 904  
     set\_use\_jack\_transport  
         MHAIQJackdb::io\_jack\_t, 948  
         MHAJack::client\_t, 976  
     set\_value  
         MHAParser::keyword\_list\_t, 1069  
     set\_weights  
         MHAFilter::complex\_bandpass\_t, 870  
         MHAFilter::gamma\_flt\_t, 891  
     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, 568  
         io\_file\_t, 604  
         lsl2ac::lsl2ac\_t, 680  
         MHAParser::variable\_t, 1115  
         MHAParser::window\_t, 1137  
         osc2ac\_t, 1319

overlapadd::overlapadd\_if\_t, 1328  
spec2wave\_if\_t, 1462  
testplugin::config\_parser\_t, 1479  
wave2spec\_if\_t, 1490  
windowselector\_t, 1515  
  
sf  
  MHASndFile::sf\_t, 1273  
  wavwriter\_t, 1503  
sf\_in  
  io\_file\_t, 606  
sf\_out  
  io\_file\_t, 607  
sf\_t  
  MHASndFile::sf\_t, 1272  
sf\_wave\_t  
  MHASndFile::sf\_wave\_t, 1274  
sfinf\_in  
  io\_file\_t, 607  
sfinf\_out  
  io\_file\_t, 607  
shadowfilter\_begin, 160  
shadowfilter\_begin.cpp, 1697  
shadowfilter\_begin::cfg\_t, 1421  
  cfg\_t, 1422  
  in\_spec\_copy, 1422  
  nch, 1422  
  ntracks, 1423  
  out\_spec, 1422  
  process, 1422  
shadowfilter\_begin::shadowfilter\_begin\_t, 1423  
  basename, 1424  
  nch, 1425  
  ntracks, 1425  
  prepare, 1424  
  process, 1424  
  shadowfilter\_begin\_t, 1424  
shadowfilter\_begin\_t  
  shadowfilter\_begin::shadowfilter\_begin\_t, 1424  
shadowfilter\_end, 161  
shadowfilter\_end.cpp, 1697  
shadowfilter\_end::cfg\_t, 1425  
  ac, 1426  
  cfg\_t, 1425  
  gains, 1427  
  in\_spec, 1426  
  name, 1426  
  nch\_out, 1426  
  nfft, 1426  
  ntracks, 1426  
  
out\_spec, 1427  
process, 1426  
shadowfilter\_end::shadowfilter\_end\_t, 1427  
  basename, 1428  
  prepare, 1428  
  process, 1428  
  shadowfilter\_end\_t, 1428  
shadowfilter\_end\_t  
  shadowfilter\_end::shadowfilter\_end\_t, 1428  
  
shape  
  MHAOvIFilter::fftfb\_t, 1006  
shapes  
  MHAOvIFilter::fftfb\_vars\_t, 1010  
shift  
  lpc, 659  
  lpc\_config, 673  
shifted  
  fshift\_hilbert::hilbert\_shifter\_t, 516  
shutdown  
  mha\_tcp::server\_t, 819  
side  
  mha\_channel\_info\_t, 741  
sign\_t  
  MHASignal::schroeder\_t, 1242  
signal  
  testplugin::if\_t, 1483  
signal\_counter  
  MHASignal, 149  
signal\_dimensions  
  matlab\_wrapper::matlab\_wrapper\_t::wrapped\_plugin\_t, 706  
signal\_gen\_t  
  audiometerbackend::signal\_gen\_t, 323  
signal\_out  
  MHAPlugin\_Split::split\_t, 1176  
signal\_parser\_t  
  testplugin::signal\_parser\_t, 1484  
signal\_type  
  matlab\_wrapper::types< MHA\_SPECTRUM >, 707  
  matlab\_wrapper::types< MHA\_WAVEFORM >, 708  
sigtype  
  audiometerbackend::audiometer\_if\_t, 317  
sin125  
  speechnoise\_t, 1468  
sin1k  
  speechnoise\_t, 1468  
sin250  
  speechnoise\_t, 1468

**sin2k**  
 speechnoise\_t, 1468  
**sin4k**  
 speechnoise\_t, 1468  
**sin500**  
 speechnoise\_t, 1468  
**sin8k**  
 speechnoise\_t, 1468  
**sinc**  
 MHAFilter, 109  
**sine.cpp**, 1697  
**sine\_cfg\_t**, 1429  
 amplitude, 1430  
 channels, 1430  
 mix, 1430  
 phase\_increment\_div\_2pi, 1430  
 sine\_cfg\_t, 1429  
**sine\_t**, 1431  
 audiometerbackend::sine\_t, 324  
 channels, 1433  
 frequency, 1433  
 lev, 1433  
 mode, 1433  
 patchbay, 1433  
 phase\_div\_2pi, 1433  
 prepare, 1432  
 process, 1432  
 sine\_t, 1432  
 update\_cfg, 1433  
**sInput**  
 MHAFilter::fftfilter\_t, 877  
**size**  
 MHA\_AC::ac2matrix\_helper\_t, 718  
 MHA\_AC::acspace2matrix\_t, 724  
 MHAKernel::algo\_comm\_class\_t, 989  
 MHASignal::matrix\_t, 1227  
 osc2ac\_t, 1319  
 Vector and matrix processing toolbox, 43,  
     44  
**size\_t**  
 MHParse::keyword\_list\_t, 1068  
**skip**  
 ac2lsl::ac2lsl\_t, 165  
 ac2lsl::cfg\_t, 168  
 ac2osc\_t, 184  
 lsl2ac::save\_var\_t, 688  
**skipcnt**  
 ac2lsl::cfg\_t, 168  
 ac2osc\_t, 184  
**Sleep**  
 mha\_tcp.hh, 1643

**slope**  
 softclipper\_t, 1458  
 softclipper\_variables\_t, 1460  
**slope\_db**  
 softclip\_t, 1456  
**smooth\_cepstrum**, 161  
**smooth\_cepstrum.cpp**, 1697  
 INSERT\_PATCH, 1698  
 INSERT\_VAR, 1698  
 PATCH\_VAR, 1698  
**smooth\_cepstrum.hh**, 1698  
**smooth\_cepstrum::smooth\_cepstrum\_if\_t**,  
     1434  
 alpha\_const\_limits\_hz, 1438  
 alpha\_const\_vals, 1438  
 alpha\_pitch, 1437  
 beta\_const, 1437  
 delta\_pitch, 1437  
 f0\_high, 1437  
 f0\_low, 1437  
 gain\_min\_db, 1438  
 kappa\_const, 1437  
 lambda\_thresh, 1437  
 noisePow\_name, 1438  
 on\_model\_param\_valuechanged, 1436  
 patchbay, 1439  
 prepare, 1436  
 prepared, 1439  
 prior\_q, 1438  
 process, 1435  
 release, 1436  
**smooth\_cepstrum\_if\_t**, 1435  
 spp, 1438  
 update\_cfg, 1436  
 win\_f0, 1438  
 xi\_min\_db, 1437  
 xi\_opt\_db, 1438  
**smooth\_cepstrum::smooth\_cepstrum\_t**, 1439  
 ~smooth\_cepstrum\_t, 1440  
 ac, 1441  
 alpha\_const, 1442  
 alpha\_frame, 1444  
 alpha\_hat, 1443  
 alpha\_prev, 1442  
 fftlen, 1441  
 gain\_min, 1442  
 gain\_wiener, 1444  
 gamma\_post, 1443  
 GLR, 1445  
 GLRexp, 1445  
 lambda\_ceps, 1444

lambda\_ceps\_prev, 1444  
lambda\_ml\_ceps, 1443  
lambda\_ml\_full, 1443  
lambda\_ml\_smooth, 1443  
lambda\_spec, 1444  
log\_lambda\_spec, 1444  
logGLRFact, 1445  
max\_q, 1445  
max\_val, 1445  
mha\_fft, 1441  
nchan, 1442  
nfreq, 1441  
noisePow, 1443  
ola\_powspec\_scale, 1442  
operator=, 1441  
params, 1441  
pitch\_set\_first, 1445  
pitch\_set\_last, 1445  
powSpec, 1443  
priorFact, 1445  
process, 1441  
q\_high, 1442  
q\_low, 1442  
smooth\_cepstrum\_t, 1440  
spec\_out, 1444  
winF0, 1442  
xi\_est, 1444  
xi\_min, 1442  
xi\_ml, 1443  
xiOpt, 1445  
smooth\_cepstrum::smooth\_params, 1446  
alpha\_const\_limits\_hz, 1448  
alpha\_const\_vals, 1448  
alpha\_pitch, 1448  
beta\_const, 1448  
delta\_pitch, 1447  
f0\_high, 1447  
f0\_low, 1447  
gain\_min\_db, 1448  
in\_cfg, 1447  
kappa\_const, 1448  
lambda\_thresh, 1447  
noisePow\_name, 1449  
prior\_q, 1448  
smooth\_params, 1446  
winF0, 1448  
xi\_min\_db, 1447  
xi\_opt\_db, 1448  
smooth\_cepstrum\_if\_t  
smooth\_cepstrum::smooth\_cepstrum\_if\_t, 1435  
smooth\_cepstrum\_t  
smooth\_cepstrum::smooth\_cepstrum\_t, 1440  
smooth\_params  
smooth\_cepstrum::smooth\_params, 1446  
smoothgains\_bridge, 161  
smoothgains\_bridge.cpp, 1698  
smoothgains\_bridge::overlapadd\_if\_t, 1449  
~overlapadd\_if\_t, 1450  
algo, 1452  
cf\_in, 1452  
cf\_out, 1452  
epsilon, 1451  
irswnd, 1451  
mode, 1451  
overlapadd\_if\_t, 1450  
patchbay, 1451  
plugloader, 1451  
prepare, 1450  
process, 1451  
release, 1450  
update, 1451  
smoothgains\_bridge::smoothspec\_wrap\_t, 1452  
proc\_1, 1453  
proc\_2, 1453  
smoothspec, 1453  
smoothspec\_epsilon, 1454  
smoothspec\_wrap\_t, 1453  
spec\_in\_copy, 1453  
use\_smoothspec, 1453  
smoothspec  
MHAFilter::smoothspec\_t, 928  
smoothgains\_bridge::smoothspec\_wrap\_t, 1453  
smoothspec\_epsilon  
smoothgains\_bridge::smoothspec\_wrap\_t, 1454  
smoothspec\_t  
MHAFilter::smoothspec\_t, 927  
smoothspec\_wrap\_t  
smoothgains\_bridge::smoothspec\_wrap\_t, 1453  
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, 814  
 sock\_init\_t  
   MHA\_TCP::sock\_init\_t, 821  
 sock\_initializer  
   MHA\_TCP, 103  
 Sockaccept\_Event  
   MHA\_TCP::Sockaccept\_Event, 822  
 SOCKET  
   MHA\_TCP, 102  
   mha\_tcp.cpp, 1641  
 SOCKET\_ERROR  
   mha\_tcp.cpp, 1640  
 Sockread\_Event  
   MHA\_TCP::Sockread\_Event, 823  
 Sockwrite\_Event  
   MHA\_TCP::Sockwrite\_Event, 824  
 softclip  
   calibrator\_runtime\_layer\_t, 337  
   calibrator\_variables\_t, 343  
 softclip.cpp, 1699  
 softclip\_t, 1454  
   attack, 1456  
   decay, 1456  
   patchbay, 1456  
   prepare, 1455  
   process, 1455  
   slope\_db, 1456  
   softclip\_t, 1455  
   start\_limit, 1456  
   tftype, 1456  
   update, 1455  
 softclipper\_t, 1456  
   attack, 1457  
   clipmeter, 1457  
   decay, 1457  
   hardlimit, 1458  
   linear, 1458  
   process, 1457  
   slope, 1458  
 softclipper\_t, 1457  
   threshold, 1458  
 softclipper\_variables\_t, 1458  
   clipped, 1460  
   hardlimit, 1460  
   linear, 1460  
   max\_clipped, 1460  
   slope, 1460  
   softclipper\_variables\_t, 1459  
   tau\_attack, 1459  
   tau\_clip, 1459  
   tau\_decay, 1459  
   threshold, 1460  
 solve\_MVDR  
   rohBeam::rohBeam, 1398  
 sort\_fftw2spec  
   MHASignal::fft\_t, 1210  
 sort\_spec2fftw  
   MHASignal::fft\_t, 1210  
 sound  
   io\_asterisk\_t, 599  
   io\_tcp\_t, 637  
 source\_azimuth\_degrees  
   rohBeam::rohBeam, 1400  
 source\_channel\_index  
   MHAFilter::partitioned\_convolution\_t::index\_t, 919  
   MHAFilter::transfer\_function\_t, 936  
 source\_id  
   ac2lsl::ac2lsl\_t, 165  
   ac2lsl::cfg\_t, 168  
 sout  
   matrixmixer::cfg\_t, 710  
   route::process\_t, 1413  
 sout\_ac  
   route::process\_t, 1413  
 sout\_buf  
   gtfb\_simd\_cfg\_t, 557  
 spec2fir  
   MHAFilter, 108  
   MHAFilter::smoothspec\_t, 929  
 spec2spec  
   plugindescription\_t, 1356  
 spec2wave  
   MHASignal::fft\_t, 1208, 1209  
   overlapadd::overlapadd\_t, 1332  
   plugindescription\_t, 1356  
 spec2wave.cpp, 1699  
   max, 1699  
   min, 1699  
 spec2wave\_if\_t, 1461

prepare, 1462  
process, 1462  
ramplen, 1463  
release, 1462  
setlock, 1462  
spec2wave\_if\_t, 1462  
update, 1462  
window\_config, 1463  
spec2wave\_scale  
    MHASignal::fft\_t, 1209  
spec2wave\_t, 1463  
    ~spec2wave\_t, 1464  
    calc\_out, 1465  
    ft, 1464  
    nfft, 1465  
    npad1, 1464  
    npad2, 1464  
    nwndshift, 1465  
    out\_buf, 1465  
    postwindow, 1465  
    process, 1464  
    ramps, 1465  
    sc, 1465  
    spec2wave\_t, 1464  
    write\_buf, 1465  
spec\_fader\_t, 1466  
    ~spec\_fader\_t, 1466  
    fr, 1466  
    gains, 1466  
    nch, 1466  
    spec\_fader\_t, 1466  
spec\_in  
    matlab\_wrapper::matlab\_wrapper\_t::wrapped\_plugin\_type\_t, 1468  
        706  
    MHAPlugin\_Split::domain\_handler\_t, 1165  
    overlapadd::overlapadd\_t, 1333  
    wave2spec\_t, 1497  
spec\_in\_copy  
    smoothgains\_bridge::smoothspec\_wrap\_t, 1453  
spec\_out  
    matlab\_wrapper::matlab\_wrapper\_t::wrapped\_plugin\_type\_t, 1468, 1469  
        706  
    MHAPlugin\_Split::domain\_handler\_t, 1165  
    MHAPlugin\_Split::split\_t, 1178  
    smooth\_cepstrum::smooth\_cepstrum\_t, 1444  
specSteer1  
    acSteer\_config, 232  
specSteer2  
    acSteer\_config, 232  
spectrum\_t  
    MHA\_AC::spectrum\_t, 732  
    MHAMultiSrc::spectrum\_t, 995  
    MHASignal::spectrum\_t, 1245, 1246  
speechnoise  
    calibrator\_runtime\_layer\_t, 337  
speechnoise.cpp, 1699  
    bandw\_correction, 1702  
    erb\_hz\_f\_hz, 1701  
    fHz2bandno, 1701  
    hz2hz, 1701  
    NUM\_ENTR\_LTASS, 1701  
    NUM\_ENTR\_MHAORIG, 1701  
    NUM\_ENTR\_OLNOISE, 1701  
    vLTASS\_combined\_lev, 1703  
    vLTASS\_female\_lev, 1703  
    vLTASS\_freq, 1702  
    vLTASS\_male\_lev, 1703  
    vMHAOrigFreq, 1702  
    vMHAOrigSpec, 1702  
    vOlnoiseFreq, 1703  
    vOlnoiseLev, 1703  
speechnoise.h, 1703  
speechnoise\_t, 1467  
    brown, 1468  
    creator, 1469  
    LTASS\_combined, 1468  
    LTASS\_female, 1468  
    LTASS\_male, 1468  
    mha, 1468  
    noise, 1468  
    olnoise, 1468  
    pink, 1468  
    sin125, 1468  
    sin1k, 1468  
    sin250, 1468  
    sin2k, 1468  
    sin4k, 1468  
    sin500, 1468  
    sin8k, 1468  
    SPL, 1468  
    rmslevel, 159  
spl2hl  
    MHAUtils, 153

split.cpp, 1704  
     MHAPLUGIN\_OVERLOAD\_OUTDOMAIN, 1705  
     native\_thread\_platform\_type, 1705  
     posixthreads, 1705  
 split\_t  
     MHAPlugin\_Split::split\_t, 1174  
 splitted\_part\_t  
     MHAPlugin\_Split::splitted\_part\_t, 1180, 1181  
 spnoise\_channels  
     calibrator\_variables\_t, 342  
 spnoise\_level  
     calibrator\_variables\_t, 342  
 spnoise\_mode  
     calibrator\_variables\_t, 342  
 spnoise\_parser  
     calibrator\_variables\_t, 342  
 spp  
     smooth\_cepstrum::smooth\_cepstrum\_if\_t, 1438  
 sq2db  
     MHASignal, 140  
 srate  
     ac2lsl::cfg\_t, 168  
     adm\_if\_t, 275  
     calibrator\_variables\_t, 342  
     mhaconfig\_t, 849  
     MHAParser::mhaconfig\_mon\_t, 1086  
     MHAPlugin\_Resampling::resampling\_if\_t, 1155  
     testplugin::config\_parser\_t, 1480  
 srate\_  
     MHAFilter::gamma\_flt\_t, 892  
 srv  
     osc2ac\_t, 1319  
 start  
     alsa\_base\_t, 287  
     alsa\_t< T >, 293  
     fw\_t, 522  
     io\_alsa\_t, 577  
     io\_asterisk\_fwcb\_t, 582  
     io\_asterisk\_t, 598  
     io\_file\_t, 603  
     io\_lib\_t, 610  
     io\_parser\_t, 614  
     io\_tcp\_fwcb\_t, 619  
     io\_tcp\_t, 636  
     MHAJack::client\_t, 974  
 START\_BETA  
     ADM, 82

start\_event  
     io\_alsa\_t, 579  
     io\_asterisk\_fwcb\_t, 584  
     io\_file\_t, 605  
     io\_parser\_t, 616  
     io\_tcp\_fwcb\_t, 620  
     MHAIOPortAudio::io\_portaudio\_t, 959  
     MHAJack::client\_t, 978  
 start\_handle  
     io\_alsa\_t, 579  
     io\_asterisk\_fwcb\_t, 584  
     io\_file\_t, 605  
     io\_parser\_t, 616  
     io\_tcp\_fwcb\_t, 621  
     MHAIOPortAudio::io\_portaudio\_t, 959  
     MHAJack::client\_t, 978  
 start\_limit  
     softclip\_t, 1456  
 start\_lin  
     cfg\_t, 346  
 start\_new\_session  
     plugins::hoertech::acrec::acrec\_t, 1376  
     wavrec\_t, 1499  
 start\_stdin\_thread  
     mhserver\_t, 1191  
 started  
     fw\_t, 523  
     io\_parser\_t, 615  
 starting  
     mha\_drifter\_fifo\_t< T >, 757  
 startpos  
     addsndfile::addsndfile\_if\_t, 253  
 startsample  
     io\_file\_t, 606  
 startup\_zeros  
     mha\_drifter\_fifo\_t< T >, 760  
 stat\_t  
     MHA\_AC::stat\_t, 734  
     MHASignal::stat\_t, 1251  
 state  
     fw\_t, 527  
     gtfb\_analyzer::gtfb\_analyzer\_cfg\_t, 549  
     matlab\_wrapper::matlab\_wrapper\_t::wrapped\_plugin\_t, 704  
     MHA\_TCP::Thread, 828  
     MHAFilter::filter\_t, 888  
 state\_cpupload  
     MHAIOJack::io\_jack\_t, 943  
     MHAIOJackdb::io\_jack\_t, 951  
 state\_parser  
     MHAIOJack::io\_jack\_t, 943

MHAIOJackdb::io\_jack\_t, 951  
state\_priority  
  MHAIOJack::io\_jack\_t, 943  
  MHAIOJackdb::io\_jack\_t, 951  
state\_scheduler  
  MHAIOJack::io\_jack\_t, 943  
  MHAIOJackdb::io\_jack\_t, 951  
state\_t  
  fw\_t, 521  
state\_xruns  
  MHAIOJack::io\_jack\_t, 943  
  MHAIOJackdb::io\_jack\_t, 951  
states  
  gtfb\_analyzer::gtfb\_analyzer\_cfg\_t, 547  
staticgain  
  coherence::cohflt\_t, 353  
  coherence::vars\_t, 355  
status  
  MHA\_TCP::Wakeup\_Event, 834  
std  
  MHA\_AC::stat\_t, 735  
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, 1470  
  ~steerbf, 1471  
  angle\_ind, 1472  
  angle\_src, 1472  
  bf\_src, 1472  
  patchbay, 1472  
  prepare, 1471  
  process, 1471  
  release, 1472  
  steerbf, 1471  
  update\_cfg, 1472  
steerbf.cpp, 1705  
  INSERT\_PATCH, 1705  
  PATCH\_VAR, 1705  
steerbf.h, 1706  
steerbf\_config, 1473  
  \_steerbf, 1474  
  ~steerbf\_config, 1473  
  ac, 1474  
  bf\_src\_copy, 1474  
  bf\_vec, 1474  
  nangle, 1474  
  nchan, 1474  
  nfreq, 1474  
  outSpec, 1474  
  process, 1473  
  steerbf\_config, 1473  
steerFile  
  acSteer, 230  
stime  
  MHA\_TCP, 103  
stop  
  alsa\_base\_t, 287  
  alsa\_t< T >, 293  
  fw\_t, 522  
  io\_alsa\_t, 578  
  io\_asterisk\_fwcb\_t, 583  
  io\_asterisk\_t, 598  
  io\_file\_t, 603  
  io\_lib\_t, 610  
  io\_parser\_t, 614  
  io\_tcp\_fwcb\_t, 620  
  io\_tcp\_t, 636  
  mha\_drifter\_fifo\_t< T >, 757  
  MHAJack::client\_t, 974  
stop\_event  
  io\_alsa\_t, 579  
  io\_asterisk\_fwcb\_t, 584  
  io\_file\_t, 605  
  io\_parser\_t, 616  
  io\_tcp\_fwcb\_t, 620  
  MHAIOPortAudio::io\_portaudio\_t, 959  
  MHAJack::client\_t, 978  
stop\_handle  
  io\_alsa\_t, 579  
  io\_asterisk\_fwcb\_t, 584  
  io\_file\_t, 605  
  io\_parser\_t, 616  
  io\_tcp\_fwcb\_t, 621  
  MHAIOPortAudio::io\_portaudio\_t, 959  
  MHAJack::client\_t, 978  
stopped  
  fw\_t, 522, 523  
  io\_file\_t, 603  
  io\_parser\_t, 615  
  MHAJack::client\_t, 977  
  route::interface\_t, 1411  
store  
  rmslevel::mon\_t, 1387  
store\_frame  
  acsave::cfg\_t, 223  
  acsave::save\_var\_t, 226  
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, 1530  
 latex\_doc\_t, 642  
**stream**  
 ac2lsl::save\_var\_t< mha\_complex\_t >, 178  
 ac2lsl::save\_var\_t< T >, 175  
 lsl2ac::save\_var\_t, 686  
**stream\_dir**  
 alsa\_dev\_par\_parser\_t, 290  
**stream\_info**  
 MHAIOPortAudio::io\_portaudio\_t, 958  
**stream\_info\_t**  
 MHAIOPortAudio::stream\_info\_t, 961  
**streambuf**  
 mha\_tcp::buffered\_socket\_t, 797  
**streams**  
 lsl2ac::lsl2ac\_t, 680  
**STRERROR**  
 MHA\_TCP, 102  
**strict\_channel\_match**  
 io\_file\_t, 606  
**strict\_srate\_match**  
 io\_file\_t, 606  
**strict\_window\_ratio**  
 overlapadd::overlapadd\_if\_t, 1329  
 wave2spec\_if\_t, 1491  
**stride**  
 comm\_var\_t, 361  
 testplugin::ac\_parser\_t, 1477  
**string\_mon\_t**  
 MHAParser::string\_mon\_t, 1110  
**string\_t**  
 MHAParser::string\_t, 1112  
**strip**  
 MHAUtils, 151  
**STRLEN**  
 mha\_errno.c, 1585  
**strNames\_AC**  
 acConcat\_wave\_config, 203  
**strreplace**  
 MHAParser, 127

**structVersion**  
 MHAIOPortAudio::device\_info\_t, 953  
**sub4f**  
 gtfb\_simd.cpp, 1558  
**sub\_ac**  
 dbasync\_native::db\_if\_t, 375  
**subsample\_delay\_t**  
 MHASignal::subsample\_delay\_t, 1253  
**subsampledelay\_coeff**  
 ADM, 81  
**suggestedInputLatency**  
 MHAIOPortAudio::io\_portaudio\_t, 960  
**suggestedOutputLatency**  
 MHAIOPortAudio::io\_portaudio\_t, 960  
**sum**  
 MHASignal::stat\_t, 1252  
 MHASignal::waveform\_t, 1265  
**sum2**  
 MHASignal::stat\_t, 1252  
**sum\_and\_fill**  
 rmslevel::rmslevel\_t, 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, 378  
**sWeights**  
 MHAFilter::fftfilter\_t, 877  
**symmetry\_scale**  
 MHAOvlFilter::fspacing\_t, 1017  
**sync**  
 mha\_fifo\_lw\_t< T >, 769  
 mha\_fifo\_thread\_guard\_t, 780  
**sync\_osc2ac**  
 osc\_server\_t, 1321  
 osc\_variable\_t, 1324  
**sysread**  
 MHA\_TCP::Connection, 802  
**syswrite**  
 MHA\_TCP::Connection, 802

**T**  
 MHA\_TCP::OS\_EVENT\_TYPE, 810

**t**  
 acsave::mat4head\_t, 224  
 mha\_tictoc\_t, 835

plingoing::plingoing\_t, 1342  
table  
    cpupload::cpupload\_cfg\_t, 365  
table\_size  
    cpupload::cpupload\_if\_t, 368  
table\_t  
    MHATableLookup::table\_t, 1280  
tail  
    MHAFilter::fftfilterbank\_t, 883  
target\_channel\_index  
    MHAFilter::partitioned\_convolution\_t::index\_t, speechnoise\_t, 1468  
        919  
    MHAFilter::transfer\_function\_t, 936  
tau  
    coherence::vars\_t, 354  
    droptect\_t, 439  
    fader\_if\_t, 484  
    levelmeter\_t, 656  
tau\_attack  
    softclipper\_variables\_t, 1459  
tau\_beta  
    adm\_if\_t, 274  
tau\_blocking\_XkXi\_ms  
    rohBeam::rohBeam, 1401  
tau\_blocking\_XkY\_ms  
    rohBeam::rohBeam, 1401  
tau\_clip  
    softclipper\_variables\_t, 1459  
tau\_decay  
    softclipper\_variables\_t, 1459  
tau\_level  
    calibrator\_variables\_t, 342  
tau\_Lowpass  
    windnoise::if\_t, 1512  
tau\_postfilter\_ms  
    rohBeam::rohBeam, 1401  
tau\_unit  
    coherence::vars\_t, 354  
tauattack  
    dc::dc\_vars\_t, 388  
    dc\_simple::dc\_vars\_t, 402  
taudecay  
    dc::dc\_vars\_t, 389  
    dc\_simple::dc\_vars\_t, 402  
taugain  
    DynComp::dc\_afterburn\_vars\_t, 453  
taurmslevel  
    dc::dc\_vars\_t, 388  
tc  
    lsl2ac::save\_var\_t, 687  
tc\_name  
    lsl2ac::save\_var\_t, 687  
tcp\_connect\_to  
    mha\_tcp.cpp, 1641  
tcp\_connect\_to\_with\_timeout  
    mha\_tcp.cpp, 1641  
tcp\_server\_t  
    mhaserver\_t::tcp\_server\_t, 1194  
tcpserver  
    mhaserver\_t, 1192  
TEN\_SPL  
    TEN\_SPL\_250\_8k  
        speechnoise\_t, 1468  
    TEN\_SPL\_50\_16k  
        speechnoise\_t, 1468  
termination\_request  
    MHAPlugin\_Split::posix\_threads\_t, 1172  
test\_error  
    io\_lib\_t, 610  
    MHAParser::c\_ifc\_parser\_t, 1048  
    PluginLoader::mhaplugloader\_t, 1369  
test\_fail  
    dc\_simple, 87  
test\_prepare  
    testplugin::if\_t, 1482  
test\_process  
    testplugin::if\_t, 1482  
test\_version  
    PluginLoader::mhaplugloader\_t, 1369  
testalsadevice.c, 1706  
    main, 1706  
testplugin, 161  
testplugin.cpp, 1706  
testplugin::ac\_parser\_t, 1475  
    \_MHA\_AC\_CHAR, 1476  
    \_MHA\_AC\_DOUBLE, 1476  
    \_MHA\_AC\_FLOAT, 1476  
    \_MHA\_AC\_INT, 1476  
    \_MHA\_AC\_MHACOMPLEX, 1476  
    \_MHA\_AC\_MHAREAL, 1476  
    \_unknown, 1476  
    ac\_parser\_t, 1476  
    char\_data, 1477  
    complex\_data, 1477  
    data\_type, 1477  
    data\_type\_t, 1476  
    do\_get\_var, 1476  
    do\_insert\_var, 1476  
    float\_data, 1477  
    get\_var, 1477  
    insert\_var, 1477

int\_data, 1477  
 num\_entries, 1477  
 patchbay, 1478  
 stride, 1477  
 testplugin::config\_parser\_t, 1478  
     channels, 1479  
     config\_parser\_t, 1479  
     domain, 1480  
     ffflen, 1480  
     fragsize, 1480  
     get, 1479  
     set, 1479  
     setlock, 1479  
     srate, 1480  
     wndlen, 1480  
 testplugin::if\_t, 1481  
     \_prepare, 1483  
     ac, 1483  
     config\_in, 1483  
     config\_out, 1483  
     if\_t, 1482  
     patchbay, 1483  
     plug, 1483  
     prepare, 1482  
     process, 1482  
     signal, 1483  
     test\_prepare, 1482  
     test\_process, 1482  
 testplugin::signal\_parser\_t, 1484  
     input\_spec, 1485  
     input\_wave, 1485  
     output\_spec, 1485  
     output\_wave, 1485  
     signal\_parser\_t, 1484  
 tftype  
     MHAPlugin::plugin\_t< runtime\_cfg\_t >, 1152  
     softclip\_t, 1456  
 The MHA Framework interface, 22  
 The openMHA configuration language, 30  
 The openMHA Toolbox library, 31  
 thefullname  
     MHAParser::base\_t, 1039  
 thirdoctave\_analyzer\_t  
     MHAFilter::thirdoctave\_analyzer\_t, 931  
 this\_outer\_out  
     MHASignal::doublebuffer\_t, 1206  
 thr\_f  
     MHA\_TCP::Thread, 825  
 Thread  
     MHA\_TCP::Thread, 826  
 thread  
     analysepath\_t, 309  
     dbasync\_native::dbasync\_t, 378  
     io\_asterisk\_t, 600  
     io\_tcp\_t, 638  
     MHAPlugin\_Split::posix\_threads\_t, 1171  
     MHAPlugin\_Split::splitted\_part\_t, 1184  
 thread\_arg  
     MHA\_TCP::Thread, 828  
 thread\_attr  
     MHA\_TCP::Thread, 827  
 thread\_finish\_event  
     MHA\_TCP::Thread, 828  
 thread\_func  
     MHA\_TCP::Thread, 828  
 thread\_handle  
     MHA\_TCP::Thread, 827  
 thread\_platform  
     MHAPlugin\_Split::split\_t, 1177  
 thread\_platform\_t  
     MHAPlugin\_Split::thread\_platform\_t, 1185  
 thread\_start  
     analysispath.cpp, 1531  
     dbasync\_native, 86  
     io\_alsa\_t, 578  
     MHAPlugin\_Split::posix\_threads\_t, 1170  
 thread\_start\_func  
     mha\_tcp.cpp, 1641  
 thread\_startup\_function  
     MHAIOAsterisk.cpp, 1655  
     MHAIOTCP.cpp, 1681  
 threshold  
     droptect\_t, 439  
     softclipper\_t, 1458  
     softclipper\_variables\_t, 1460  
 threshold\_compare  
     windnoise::cfg\_t, 1506  
 tic  
     lsl2ac::save\_var\_t, 688  
 tictoc  
     mhachain::plugs\_t, 847  
 timeout  
     MHA\_TCP::OS\_EVENT\_TYPE, 811  
     MHA\_TCP::Timeout\_Watcher, 831  
 Timeout\_Event  
     MHA\_TCP::Timeout\_Event, 829  
 Timeout\_Watcher  
     MHA\_TCP::Timeout\_Watcher, 831  
 timeshift  
     Vector and matrix processing toolbox, 45

tmp  
    level\_matching::level\_matching\_config\_t,  
        649  
tmp\_spec  
    MHAFilter::smoothspec\_t, 930  
tmp\_wave  
    MHAFilter::smoothspec\_t, 930  
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, 607  
transducers.cpp, 1707  
    kw\_index2type, 1707  
    rmslevelmeter, 1707  
    rtcalibrator, 1707  
    vint\_0123n1, 1708  
transfer\_function\_t  
    MHAFilter::transfer\_function\_t, 934  
trigger\_accept  
    mha\_tcp::server\_t, 819  
trigger\_processing  
    MHAPlugin\_Split::split\_t, 1176  
    MHAPlugin\_Split::splitted\_part\_t, 1182  
trigger\_read\_line  
    mha\_tcp::server\_t, 819  
trim  
    MHAParser, 126  
try\_accept  
    MHA\_TCP::Server, 813  
try\_write  
    MHA\_TCP::Connection, 805  
ts  
    lsl2ac::save\_var\_t, 687  
ts\_buf  
    lsl2ac::save\_var\_t, 686  
ts\_name  
    lsl2ac::save\_var\_t, 687  
ttl  
    ac2osc\_t, 183  
tv1  
    mha\_tictoc\_t, 835  
tv2  
    mha\_tictoc\_t, 835  
types  
    ac2lsl, 78  
tz  
    mha\_tictoc\_t, 835  
    rt\_nlms\_t, 1416  
UbufferPrew  
    adaptive\_feedback\_canceller\_config, 248  
UCL  
    AuditoryProfile::parser\_t::ear\_t, 328  
    AuditoryProfile::profile\_t::ear\_t, 333  
Uflt  
    rt\_nlms\_t, 1416  
uint\_mode  
    addsndfile::addsndfile\_if\_t, 254  
uint\_vector\_t  
    MHASignal::uint\_vector\_t, 1256  
underflow  
    MHAFilter::polyphase\_resampling\_t, 923  
UNIT  
    rmslevel, 159  
unit  
    MHAOvlFilter::fscale\_t, 1013  
    rmslevel::rmslevel\_if\_t, 1390  
    rmslevel::rmslevel\_t, 1393  
unit2hz  
    MHAOvlFilter::scale\_var\_t, 1025  
unlock\_channels  
    fw\_vars\_t, 528  
unlock\_srate\_fragsize  
    fw\_vars\_t, 528  
unset\_fb\_pars  
    DynComp::dc\_afterburn\_t, 450  
up  
    MHASignal::schroeder\_t, 1242  
up\_incl  
    MHAParser::range\_var\_t, 1108  
up\_limit  
    MHAParser::range\_var\_t, 1108  
    MHASignal::quantizer\_t, 1236  
up\_thresh  
    acPooling\_wave\_config, 217  
update  
    ac2lsl::ac2lsl\_t, 165  
    ac2wave\_if\_t, 187  
    addsndfile::addsndfile\_if\_t, 252  
    adm\_if\_t, 273  
    audiometerbackend::audiometer\_if\_t, 316  
    calibrator\_t, 340  
    coherence::cohflt\_if\_t, 348  
    cpupload::cpupload\_if\_t, 367  
    dc::dc\_if\_t, 381  
    delay::interface\_t, 408  
    DynComp::dc\_afterburn\_t, 450



matlab\_wrapper::matlab\_wrapper\_t, 697  
update\_mu  
    MHAFilter::adapt\_filter\_t, 863  
update\_ntaps  
    MHAFilter::adapt\_filter\_t, 864  
update\_parser  
    windowselector\_t, 1516  
update\_proc\_load  
    mhachain::plugs\_t, 845  
update\_PSD\_Lowpass  
    windnoise::cfg\_t, 1506  
update\_ramplen  
    altplugs\_t, 303  
update\_recmode  
    acmon::acmon\_t, 209  
update\_selector\_list  
    altplugs\_t, 303  
update\_tau  
    levelmeter\_t, 655  
update\_tau\_level  
    calibrator\_t, 340  
update\_varlist  
    ac2isl::cfg\_t, 168  
updated  
    windowselector\_t, 1516  
updater  
    MHAOvIFilter::fscale\_bw\_t, 1012  
    MHAOvIFilter::fscale\_t, 1014  
upper\_threshold  
    acPooling\_wave, 213  
UPrew  
    adaptive\_feedback\_canceller\_config, 249  
UPrewW  
    adaptive\_feedback\_canceller\_config, 249  
upsample.cpp, 1708  
upsampling\_factor  
    MHAFilter::polyphase\_resampling\_t, 923  
upscale  
    MHASignal::quantizer\_t, 1236  
us\_t, 1485  
    antialias, 1487  
    prepare, 1486  
    process, 1486  
    ratio, 1487  
    release, 1486  
    us\_t, 1486  
use\_date  
    plugins::hoertech::acrec::acrec\_t, 1377  
    wavrec\_t, 1500  
use\_frozen\_cfg\_t, 345  
use\_jack\_transport  
    MHAIOJackdb::io\_jack\_t, 949  
    MHAJack::client\_t, 979  
use\_mat  
    acmon::ac\_monitor\_t, 206  
use\_own\_ac  
    altplugs\_t, 303  
use\_sine  
    cpupload::cpupload\_cfg\_t, 365  
    cpupload::cpupload\_if\_t, 368  
use\_smoothspec  
    smoothgains\_bridge::smoothspec\_wrap\_t,  
        1453  
UseChannel\_LF\_attenuation  
    windnoise::cfg\_t, 1507  
    windnoise::if\_t, 1512  
user  
    MHAParser::window\_t, 1138  
user\_config  
    matlab\_wrapper::callback, 690  
    matlab\_wrapper::matlab\_wrapper\_rt\_cfg\_t,  
        692  
    matlab\_wrapper::matlab\_wrapper\_t::wrapped\_plugin\_t,  
        704  
user\_err\_msg  
    MHAIoalsa.cpp, 1651  
    MHAIoAsterisk.cpp, 1656  
    MHAIoFile.cpp, 1660  
    MHAIoJack.cpp, 1665  
    MHAIoJackdb.cpp, 1669  
    MHAIoParser.cpp, 1673  
    MHAIoPortAudio.cpp, 1677  
    MHAIoTCP.cpp, 1683  
user\_t  
    MHAWindow::user\_t, 1297  
username  
    MHA\_AC::ac2matrix\_helper\_t, 718  
userwnd  
    windowselector\_t, 1517  
useVAD  
    gsc\_adaptive\_stage::gsc\_adaptive\_stage,  
        538  
    gsc\_adaptive\_stage::gsc\_adaptive\_stage\_if,  
        545  
v\_G  
    adaptive\_feedback\_canceller\_config, 247  
vadName  
    gsc\_adaptive\_stage::gsc\_adaptive\_stage,  
        538  
    gsc\_adaptive\_stage::gsc\_adaptive\_stage\_if,  
        545

val2str  
   MHAParser::StrCnv, 132–134

VAL\_COMPLEX  
   ac\_mul.hh, 1520

VAL\_REAL  
   ac\_mul.hh, 1520

val\_type\_t  
   ac\_mul.hh, 1519

validate  
   AuditoryProfile::parser\_t::fmap\_t, 330  
   MHAParser::keyword\_list\_t, 1070  
   MHAParser::kw\_t, 1073  
   MHAParser::range\_var\_t, 1106, 1107

validator\_channels  
   mhasndfile.cpp, 1689

validator\_length  
   mhasndfile.cpp, 1689

value  
   AuditoryProfile::parser\_t::fmap\_t, 330  
   mha\_signal.hh, 1637  
   MHASignal::ringbuffer\_t, 1238  
   MHASignal::spectrum\_t, 1247  
   MHASignal::waveform\_t, 1263, 1264  
   Vector and matrix processing toolbox, 46–49

value\_type  
   mha\_dbdbuf\_t< FIFO >, 745  
   mha\_fifo\_t< T >, 774

valuechanged  
   MHAParser::base\_t, 1038

var  
   matlab\_wrapper::callback, 691

variable, 4

variable\_name  
   mha\_stash\_environment\_variable\_t, 793

variable\_t  
   MHAParser::variable\_t, 1115

variables, 4  
   acsave::acsave\_t, 221

varlist  
   ac2lsl::cfg\_t, 168  
   acmon::acmon\_t, 209  
   acsave::acsave\_t, 221  
   acsave::cfg\_t, 223  
   lsl2ac::cfg\_t, 676

varlist\_t  
   acsave::acsave\_t, 219

varname  
   plugins::hoertech::acrec::acrec\_t, 1377  
   plugins::hoertech::acrec::acwriter\_t, 1383

vars  
   ac2lsl::ac2lsl\_t, 165  
   ac2osc\_t, 183  
   acmon::acmon\_t, 210  
   analysispath\_if\_t, 312  
   calibrator\_t, 340  
   coherence::cohflt\_if\_t, 349  
   matlab\_wrapper::matlab\_wrapper\_t, 698  
   MHAKernel::algo\_comm\_class\_t, 990  
   osc2ac\_t, 1319

vars\_t  
   coherence::vars\_t, 354  
   MHAOvIFilter::overlap\_save\_filterbank\_t::vars\_t, 1022

vbark  
   MHAOvIFilter::barkscale, 117

vbin1  
   MHAOvIFilter::fftfb\_t, 1005

vbin2  
   MHAOvIFilter::fftfb\_t, 1005

vcomplex\_mon\_t  
   MHAParser::vcomplex\_mon\_t, 1117

vcomplex\_t  
   MHAParser::vcomplex\_t, 1119

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, 1687  
vF  
  DynComp::gaintable\_t, 458  
vfloat\_mon\_t  
  MHAParser::vfloat\_mon\_t, 1122  
vfloat\_t  
  MHAParser::vfloat\_t, 1124  
vFlag  
  DynComp::gaintable\_t, 458  
vfreq  
  MHAoVlFilter::barkscale, 117  
vGCC  
  acConcat\_wave\_config, 203  
  doasvm\_feature\_extraction\_config, 428  
vGCC\_ac  
  doasvm\_feature\_extraction\_config, 428  
vGCC\_con  
  acConcat\_wave\_config, 203  
vGCC\_name  
  doasvm\_classification, 420  
  doasvm\_feature\_extraction, 425  
vint\_0123n1  
  transducers.cpp, 1708  
vint\_mon\_t  
  MHAParser::vint\_mon\_t, 1127  
vint\_t  
  MHAParser::vint\_t, 1129  
vL  
  DynComp::gaintable\_t, 458  
vLTASS\_combined\_lev  
  speechnoise.cpp, 1703  
vLTASS\_female\_lev  
  speechnoise.cpp, 1703  
vLTASS\_freq  
  speechnoise.cpp, 1702  
vLTASS\_male\_lev  
  speechnoise.cpp, 1703  
vmax  
  gain::gain\_if\_t, 532  
vMHAOrigFreq  
  speechnoise.cpp, 1702  
vMHAOrigSpec  
  speechnoise.cpp, 1702  
vmin  
  gain::gain\_if\_t, 531  
vOlNoiseFreq  
  speechnoise.cpp, 1703  
vOlNoiseLev  
  speechnoise.cpp, 1703  
vstring\_mon\_t  
  MHAParser::vstring\_mon\_t, 1131  
vstring\_t  
  MHAParser::vstring\_t, 1133  
vy  
  MHATableLookup::linear\_table\_t, 1278  
W  
  gsc\_adaptive\_stage::gsc\_adaptive\_stage,  
    538  
  MHA\_TCP::OS\_EVENT\_TYPE, 810  
  MHAFilter::adapt\_filter\_state\_t, 861  
w  
  ac2wave\_t, 190  
  doasvm\_classification, 420  
  MHAoVlFilter::fftfb\_t, 1005  
w\_out  
  combc\_t, 359  
wait  
  MHA\_TCP::Event\_Watcher, 809  
wait\_for\_decrease  
  ma\_fifo\_posix\_threads\_t, 771  
  ma\_fifo\_thread\_platform\_t, 782  
wait\_for\_increase  
  ma\_fifo\_posix\_threads\_t, 771  
  ma\_fifo\_thread\_platform\_t, 783  
WakeUp\_Event  
  MHA\_TCP::WakeUp\_Event, 833  
wave  
  alsa\_t< T >, 294  
wave2spec  
  MHASignal::fft\_t, 1208  
  overlapadd::overlapadd\_t, 1332  
  plugindescription\_t, 1355  
wave2spec.cpp, 1708  
wave2spec.hh, 1708  
  MHAPLUGIN\_OVERLOAD\_OUTDOMAIN,  
    1708  
wave2spec\_apply\_window  
  overlapadd::overlapadd\_t, 1332  
wave2spec\_compute\_fft  
  overlapadd::overlapadd\_t, 1332  
wave2spec\_hop\_forward  
  overlapadd::overlapadd\_t, 1332  
wave2spec\_if\_t, 1487

algo, 1492  
 nfft, 1491  
 nwnd, 1491  
 prepare, 1489  
 process, 1489, 1490  
 release, 1489  
 return\_wave, 1491  
 setlock, 1490  
 strict\_window\_ratio, 1491  
 update, 1490  
 wave2spec\_if\_t, 1489  
 window\_config, 1491  
 wndpos, 1491  
 zeropadding, 1492  
 wave2spec\_scale  
     MHASignal::fft\_t, 1209  
 wave2spec\_t, 1492  
     ~wave2spec\_t, 1494  
     ac\_wndshape\_name, 1497  
     calc\_in, 1497  
     calc\_pre\_wnd, 1496  
     ft, 1496  
     get\_zeropadding, 1495  
     in\_buf, 1497  
     npad1, 1497  
     npad2, 1497  
     nwnd, 1496  
     nwndshift, 1496  
     process, 1495  
     publish\_ac\_variables, 1495  
     spec\_in, 1497  
     wave2spec\_t, 1494  
     window, 1497  
 wave2wave  
     plugindescription\_t, 1355  
 wave\_fifo  
     analysepath\_t, 308  
 wave\_in  
     matlab\_wrapper::matlab\_wrapper\_t::wrapped\_plugin\_t, 706  
     MHAPlugin\_Split::domain\_handler\_t, 1164  
 wave\_in1  
     overlapadd::overlapadd\_t, 1333  
 wave\_out  
     matlab\_wrapper::matlab\_wrapper\_t::wrapped\_plugin\_t, 706  
     MHAPlugin\_Split::domain\_handler\_t, 1164  
     MHAPlugin\_Split::split\_t, 1178  
 wave\_out1  
     overlapadd::overlapadd\_t, 1333  
 wave\_reader  
     addsndfile, 80  
 waveform\_proxy\_t  
     addsndfile::waveform\_proxy\_t, 261  
 waveform\_t  
     MHA\_AC::waveform\_t, 736  
     MHAMultiSrc::waveform\_t, 996  
     MHASignal::waveform\_t, 1261, 1262  
 wavrec.cpp, 1709  
 wavrec\_t, 1498  
     fifolen, 1500  
     minwrite, 1500  
     output\_sample\_format, 1500  
     patchbay, 1500  
     prefix, 1500  
     prepare, 1499  
     process, 1499  
     record, 1500  
     release, 1499  
     start\_new\_session, 1499  
     use\_date, 1500  
     wavrec\_t, 1499  
 wavwriter\_t, 1501  
     ~wavwriter\_t, 1502  
     act\_, 1503  
     cf\_, 1503  
     close\_session, 1503  
     create\_soundfile, 1502  
     data, 1504  
     exit\_request, 1502  
     fifo, 1503  
     format\_name, 1504  
     minw\_, 1503  
     process, 1502  
     set\_format, 1502  
     sf, 1503  
     wavwriter\_t, 1501  
     write\_thread, 1502  
     writethread, 1503  
 weights  
     delaysum::delaysum\_wave\_if\_t, 411  
     delaysum::delaysum\_wave\_t, 413  
 what  
     MHA\_Error, 762  
 win\_f0  
     smooth\_cepstrum::smooth\_cepstrum\_if\_t, 1438  
 WINAPI

mha\_plugin.hh, 1618  
windnoise, 161  
windnoise.cpp, 1709  
    register\_configuration\_variable, 1709  
windnoise.hh, 1709  
windnoise::cfg\_t, 1504  
    alpha\_Lowpass, 1507  
    cfg\_t, 1505  
    compensation, 1507  
    FrequencyBinLowPass, 1507  
    LowPassFraction, 1508  
    LowPassWindGain, 1508  
    powspec, 1508  
    process, 1506  
    PSD\_Lowpass, 1508  
    remapping, 1507  
    threshold\_compare, 1506  
    update\_PSD\_Lowpass, 1506  
    UseChannel\_LF\_attenuation, 1507  
windnoise::if\_t, 1509  
    detected, 1512  
    detected\_acname, 1512  
    if\_t, 1510  
    insert, 1511  
    lowpass\_quotient, 1512  
    lowpass\_quotient\_acname, 1513  
    LowPassCutOffFrequency, 1512  
    LowPassFraction, 1512  
    LowPassWindGain, 1512  
    patchbay, 1511  
    prepare, 1510  
    process, 1511  
    release, 1511  
    tau\_Lowpass, 1512  
    update, 1511  
    UseChannel\_LF\_attenuation, 1512  
    WindNoiseDetector, 1512  
WindNoiseDetector  
    windnoise::if\_t, 1512  
window  
    MHAFilter::smoothspec\_t, 929  
    overlapadd::overlapadd\_if\_t, 1329  
    wave2spec\_t, 1497  
window\_config  
    spec2wave\_if\_t, 1463  
    wave2spec\_if\_t, 1491  
window\_t  
    MHAParser::window\_t, 1136  
windowselector.cpp, 1709  
windowselector.h, 1709  
windowselector\_t, 1513  
    ~windowselector\_t, 1515  
    get\_window\_data, 1515  
    insert\_items, 1515  
    invalidate\_window\_data, 1516  
    patchbay, 1517  
    setlock, 1515  
    update\_parser, 1516  
    updated, 1516  
    userwnd, 1517  
    windowselector\_t, 1514  
    wnd, 1516  
    wndexp, 1517  
    wndtype, 1516  
winF0  
    smooth\_cepstrum::smooth\_cepstrum\_t,  
        1442  
    smooth\_cepstrum::smooth\_params, 1448  
wInput  
    MHAFilter::fftfilter\_t, 877  
wInput\_fft  
    MHAFilter::fftfilter\_t, 877  
wIRS\_fft  
    MHAFilter::fftfilter\_t, 877  
wnd  
    addsndfile::level\_adapt\_t, 256  
    audiometerbackend::level\_adapt\_t, 320  
    fader\_wave::level\_adapt\_t, 489  
    windowselector\_t, 1516  
wnd\_bartlett  
    MHAParser::window\_t, 1136  
wnd\_blackman  
    MHAParser::window\_t, 1136  
wnd\_funs  
    mha\_windowparser.cpp, 1645  
wnd\_hamming  
    MHAParser::window\_t, 1136  
wnd\_hann  
    MHAParser::window\_t, 1136  
wnd\_rect  
    MHAParser::window\_t, 1136  
wnd\_user  
    MHAParser::window\_t, 1136  
wndexp  
    overlapadd::overlapadd\_if\_t, 1329  
    windowselector\_t, 1517  
wndlen  
    doasvm\_feature\_extraction\_config, 427  
    mhaconfig\_t, 849  
    MHAParser::mhaconfig\_mon\_t, 1085  
    testplugin::config\_parser\_t, 1480  
wndpos

overlapadd::overlapadd\_if\_t, 1329  
 wave2spec\_if\_t, 1491  
 wndtype  
     windowselector\_t, 1516  
 worker\_thread\_priority  
     dbasync\_native::db\_if\_t, 375  
     MHAParser::split\_t, 1177  
 worker\_thread\_scheduler  
     dbasync\_native::db\_if\_t, 375  
     MHAParser::split\_t, 1177  
 wout  
     matrixmixer::cfg\_t, 710  
     route::process\_t, 1412  
 wout\_ac  
     route::process\_t, 1413  
 wOutput  
     MHAFilter::fftfilter\_t, 877  
 wOutput\_fft  
     MHAFilter::fftfilter\_t, 877  
 wrapped\_plugin\_t  
     matlab\_wrapper::matlab\_wrapper\_t::wrapped\_plugin\_t, 701  
 write  
     alsa\_base\_t, 288  
     alsa\_t< T >, 293  
     mha\_drifter\_fifo\_t< T >, 755  
     mha\_fifo\_lf\_t< T >, 764  
     mha\_fifo\_lw\_t< T >, 768  
     mha\_fifo\_t< T >, 775  
     MHA\_TCP::Connection, 805  
     MHAFilter::blockprocessing\_polyphase\_resampling, 866  
     MHAFilter::polyphase\_resampling\_t, 922  
     MHAJack::port\_t, 982  
     MHASignal::matrix\_t, 1231  
     MHASignal::ringbuffer\_t, 1239  
     MHASignal::uint\_vector\_t, 1258  
 write\_buf  
     overlapadd::overlapadd\_t, 1333  
     spec2wave\_t, 1465  
 write\_event  
     MHA\_TCP::Connection, 806  
 write\_float  
     mha\_parser.cpp, 1610  
 write\_ptr  
     mha\_fifo\_t< T >, 778  
 write\_thread  
     plugins::hoertech::acrec::acwriter\_t, 1381  
     wavwriter\_t, 1502  
 write\_to\_table  
     cpupload::cpupload\_cfg\_t, 365  
 write\_wave  
     mhasndfile.cpp, 1688  
     mhasndfile.h, 1689  
 writeaccess  
     MHAParser::base\_t, 1038  
 writer\_started  
     mha\_drifter\_fifo\_t< T >, 758  
 writer\_xruns\_in\_succession  
     mha\_drifter\_fifo\_t< T >, 759  
 writer\_xruns\_since\_start  
     mha\_drifter\_fifo\_t< T >, 758  
 writer\_xruns\_total  
     mha\_drifter\_fifo\_t< T >, 758  
 writethread  
     plugins::hoertech::acrec::acwriter\_t, 1382  
     wavwriter\_t, 1503  
 Writing openMHA Plugins. A step-by-step tutorial, 6  
 wtype  
     MHAParser::window\_t, 1138  
 wtype\_t  
     MHAParser::window\_t, 1136

X

gsc\_adaptive\_stage::gsc\_adaptive\_stage, 538  
 MHA\_TCP::OS\_EVENT\_TYPE, 810  
 MHAFilter::adapt\_filter\_state\_t, 861  
 x  
     doasvm\_classification, 420  
     gsc\_adaptive\_stage::gsc\_adaptive\_stage, 538  
 xfun  
     MHATableLookup::xy\_table\_t, 1285  
 xi\_est  
     smooth\_cepstrum::smooth\_cepstrum\_t, 1444  
 xi\_min  
     smooth\_cepstrum::smooth\_cepstrum\_t, 1442  
 xi\_min\_db  
     smooth\_cepstrum::smooth\_cepstrum\_if\_t, 1437  
     smooth\_cepstrum::smooth\_params, 1447  
 xi\_ml  
     smooth\_cepstrum::smooth\_cepstrum\_t, 1443  
 xi\_opt\_db  
     smooth\_cepstrum::smooth\_cepstrum\_if\_t, 1438  
     smooth\_cepstrum::smooth\_params, 1448  
 xiOpt

noise\_psd\_estimator::noise\_psd\_estimator\_t,  
    1314  
smooth\_cepstrum::smooth\_cepstrum\_t,  
    1445  
xiOptDb  
    noise\_psd\_estimator::noise\_psd\_estimator\_if\_t,  
        1310  
xmax  
    MHATableLookup::linear\_table\_t, 1279  
xmin  
    MHATableLookup::linear\_table\_t, 1278  
Xs  
    MHAFilter::fftfilterbank\_t, 882  
xw  
    MHAFilter::fftfilterbank\_t, 882  
xy\_table\_t  
    MHATableLookup::xy\_table\_t, 1282  
xyfun  
    MHATableLookup::xy\_table\_t, 1285  
  
Y  
gsc\_adaptive\_stage::gsc\_adaptive\_stage,  
    538  
y  
doasvm\_classification, 420  
gsc\_adaptive\_stage::gsc\_adaptive\_stage,  
    539  
y0  
dc\_simple::dc\_t::line\_t, 400  
y\_previous  
rt\_nlms\_t, 1416  
yfun  
MHATableLookup::xy\_table\_t, 1285  
Yn  
MHAFilter::complex\_bandpass\_t, 872  
MHAFilter::iir\_ord1\_real\_t, 901  
YPrew  
adaptive\_feedback\_canceller\_config, 249  
Ys  
MHAFilter::fftfilterbank\_t, 883  
yw  
MHAFilter::fftfilterbank\_t, 883  
yw\_temp  
MHAFilter::fftfilterbank\_t, 883  
  
zeropadding  
wave2spec\_if\_t, 1492  
zeros  
ac2wave\_if\_t, 188  
zerowindow  
overlapadd::overlapadd\_if\_t, 1329