

libAHetech Reference Manual

0.1

Generated by Doxygen 1.4.4

Fri Apr 7 01:18:19 2006

Contents

1	libAHetech Directory Hierarchy	1
1.1	libAHetech Directories	1
2	libAHetech Namespace Index	3
2.1	libAHetech Namespace List	3
3	libAHetech Hierarchical Index	5
3.1	libAHetech Class Hierarchy	5
4	libAHetech Class Index	9
4.1	libAHetech Class List	9
5	libAHetech File Index	13
5.1	libAHetech File List	13
6	libAHetech Directory Documentation	15
6.1	AH/ Directory Reference	15
6.2	AH/Etech/ Directory Reference	16
7	libAHetech Namespace Documentation	17
7.1	AH Namespace Reference	17
7.2	Ah Namespace Reference	18
8	libAHetech Class Documentation	27
8.1	Ah::AD711 Class Reference	27
8.2	Ah::AD745 Class Reference	28
8.3	Ah::AD797 Class Reference	29
8.4	Ah::AD845 Class Reference	30
8.5	Ah::Amplifier Class Reference	31
8.6	Ah::AnalogDevOpAmpCreator Class Reference	33
8.7	Ah::Box Class Reference	34

8.8	Ah::BPT Class Reference	36
8.9	Ah::BurrBrownOpAmpCreator Class Reference	38
8.10	Ah::Capacitor Class Reference	39
8.11	Ah::ClosedBox Class Reference	42
8.13	Ah::Cplx Class Reference	45
8.13	Ah::Cplx Class Reference	45
8.14	Ah::Current Class Reference	46
8.15	Ah::Divider Class Reference	49
8.16	Ah::FET Class Reference	50
8.17	Ah::Filter2PoleInv Class Reference	53
8.18	Ah::FreqImpedance Class Reference	55
8.19	Ah::Impedance Class Reference	57
8.20	Ah::Inductor Class Reference	60
8.21	Ah::IntPol Class Reference	63
8.22	Ah::IntPolData Struct Reference	65
8.23	Ah::IntPolLin1Lin2 Class Reference	66
8.24	Ah::IntPolLog1Lin2 Class Reference	67
8.25	Ah::InvAmp Class Reference	68
8.26	Ah::Kef_B110_A Class Reference	70
8.27	Ah::Kef_B110_B Class Reference	71
8.28	Ah::Kef_B139_B Class Reference	72
8.29	Ah::Kef_B200_A Class Reference	73
8.30	Ah::Kef_B200_B Class Reference	74
8.31	Ah::Kef_B300_B Class Reference	75
8.32	Ah::LinTechOpAmpCreator Class Reference	76
8.33	Ah::LM833 Class Reference	77
8.34	Ah::LT1028 Class Reference	78
8.35	Ah::LT1115 Class Reference	79
8.36	Ah::MathConst Class Reference	80
8.37	Ah::MiscOpAmpCreator Class Reference	81
8.38	Ah::Monacor_SP45_4 Class Reference	82
8.39	Ah::Monacor_SP45_8 Class Reference	83
8.40	Ah::Monacor_SP60_4 Class Reference	84
8.41	Ah::Monacor_SP60_8 Class Reference	85
8.42	Ah::Monacor_SPP110_4 Class Reference	86
8.43	Ah::Monacor_SPP110_8 Class Reference	87

8.44	Ah::NE5532 Class Reference	89
8.45	Ah::NoneInvAmp Class Reference	90
8.46	Ah::OP07 Class Reference	92
8.47	Ah::OP177 Class Reference	93
8.48	Ah::OP27 Class Reference	94
8.49	Ah::OP37 Class Reference	95
8.50	Ah::OP77 Class Reference	96
8.51	Ah::OPA134 Class Reference	97
8.52	Ah::OPA604 Class Reference	98
8.53	Ah::OpAmp Class Reference	99
8.54	Ah::OpAmpCreator Class Reference	103
8.55	Ah::PhysConst Class Reference	104
8.56	Ah::PhysUnitCplx Class Reference	105
8.57	Ah::PhysUnitDbl Class Reference	110
8.58	Ah::PickUp Class Reference	112
8.59	Ah::PickUpClearaudioAlphaWood Class Reference	115
8.60	Ah::PickUpClearaudioDiscovery Class Reference	116
8.61	Ah::PickUpClearaudioGammaSGold Class Reference	117
8.62	Ah::PickUpClearaudioVirtuosoWood Class Reference	118
8.63	Ah::PickUpGradoMaster Class Reference	119
8.64	Ah::PickUpGradoPlatinum Class Reference	120
8.65	Ah::PickUpGradoReference Class Reference	121
8.66	Ah::PickUpGradoStatement Class Reference	122
8.67	Ah::PickUpOrtofon540mk2 Class Reference	123
8.68	Ah::PickUpVdhBlackBeautySPX Class Reference	124
8.69	Ah::PickUpVdhColibriXGP Class Reference	125
8.70	Ah::PickUpVdhDDT2Special Class Reference	126
8.71	Ah::PickUpVdhFrogCopper Class Reference	127
8.72	Ah::PickUpVdhMc1Special Class Reference	128
8.73	Ah::PickUpVdhMm1 Class Reference	129
8.74	Ah::Power Class Reference	130
8.75	Ah::Resistor Class Reference	133
8.76	Ah::SalKey Class Reference	136
8.77	Ah::Speaker Class Reference	138
8.78	Ah::SpeakerCreator Class Reference	142
8.79	Ah::Transistor Class Reference	143

8.80	Ah::TransistorFollower Class Reference	145
8.81	Ah::VierPol Class Reference	147
8.82	Ah::VierPolA Class Reference	149
8.83	Ah::VierPolH Class Reference	152
8.84	Ah::VierPolY Class Reference	155
8.85	Ah::VierPolZ Class Reference	158
8.86	Ah::Visaton_SC_10 Class Reference	161
8.87	Ah::Voltage Class Reference	163
9	libAHetech File Documentation	167
9.1	AH/Etech/aaa.h File Reference	167
9.2	AH/Etech/Algo.h File Reference	168
9.3	AH/Etech/Amplifier.h File Reference	169
9.4	AH/Etech/AnalogDevOpAmps.h File Reference	170
9.5	AH/Etech/AnalogDevOpCreat.h File Reference	171
9.6	AH/Etech/Box.h File Reference	172
9.7	AH/Etech/BPT.h File Reference	173
9.8	AH/Etech/BurrBrownOpAmps.h File Reference	174
9.9	AH/Etech/BurrBrownOpCreat.h File Reference	175
9.10	AH/Etech/Capacitor.h File Reference	176
9.11	AH/Etech/Cd.h File Reference	177
9.12	AH/Etech/ClosedBox.h File Reference	178
9.13	AH/Etech/Current.h File Reference	179
9.14	AH/Etech/Divider.h File Reference	180
9.15	AH/Etech/Etech.h File Reference	181
9.16	AH/Etech/FET.h File Reference	182
9.17	AH/Etech/Filter2PoleInv.h File Reference	183
9.18	AH/Etech/FreqImp.h File Reference	184
9.19	AH/Etech/Impedance.h File Reference	185
9.20	AH/Etech/Inductor.h File Reference	186
9.21	AH/Etech/IntPol.h File Reference	187
9.22	AH/Etech/InvAmp.h File Reference	188
9.23	AH/Etech/KefSpeakers.h File Reference	189
9.24	AH/Etech/LinTechOpAmps.h File Reference	190
9.25	AH/Etech/LinTechOpCreat.h File Reference	191
9.26	AH/Etech/MathConst.h File Reference	192
9.27	AH/Etech/MiscOpAmps.h File Reference	193

9.28	AH/Etech/MiscOpCreat.h File Reference	194
9.29	AH/Etech/MonacorSpeakers.h File Reference	195
9.30	AH/Etech/NoneInvAmp.h File Reference	196
9.31	AH/Etech/OpAmp.h File Reference	197
9.32	AH/Etech/OpAmpCreator.h File Reference	198
9.33	AH/Etech/PhysConst.h File Reference	199
9.34	AH/Etech/PhysUnit.h File Reference	200
9.35	AH/Etech/PickUp.h File Reference	201
9.36	AH/Etech/Power.h File Reference	203
9.37	AH/Etech/Resistor.h File Reference	204
9.38	AH/Etech/Riaa.h File Reference	205
9.39	AH/Etech/SalKey.h File Reference	206
9.40	AH/Etech/Speaker.h File Reference	207
9.41	AH/Etech/SpeakerCreator.h File Reference	208
9.42	AH/Etech/TransFollow.h File Reference	209
9.43	AH/Etech/Transistor.h File Reference	210
9.44	AH/Etech/Types.h File Reference	211
9.45	AH/Etech/VdHPickUps.h File Reference	212
9.46	AH/Etech/Version.h File Reference	213
9.47	AH/Etech/VierPol.h File Reference	214
9.48	AH/Etech/VisatonSpeakers.h File Reference	215
9.49	AH/Etech/Voltage.h File Reference	216

Chapter 1

libAHetech Directory Hierarchy

1.1 libAHetech Directories

This directory hierarchy is sorted roughly, but not completely, alphabetically:

AH	15
Etech	16

Chapter 2

libAHetech Namespace Index

2.1 libAHetech Namespace List

Here is a list of all namespaces with brief descriptions:

AH	17
Ah	18

Chapter 3

libAHetech Hierarchical Index

3.1 libAHetech Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

Ah::Amplifier	31
Ah::Filter2PoleInv	53
Ah::InvAmp	68
Ah::NoneInvAmp	90
Ah::SalKey	136
Ah::AnalogDevOpAmpCreator	33
Ah::Box	34
Ah::ClosedBox	42
Ah::BurrBrownOpAmpCreator	38
Ah::Cplx	45
Ah::Cplx	45
Ah::Divider	49
Ah::IntPol	63
Ah::IntPolLin1Lin2	66
Ah::IntPolLog1Lin2	67
Ah::IntPolData	65
Ah::LinTechOpAmpCreator	76
Ah::MathConst	80
Ah::MiscOpAmpCreator	81
Ah::OpAmp	99
Ah::AD711	27
Ah::AD745	28
Ah::AD797	29
Ah::AD845	30
Ah::LM833	77
Ah::LT1028	78
Ah::LT1115	79
Ah::NE5532	89
Ah::OP07	92
Ah::OP177	93
Ah::OP27	94
Ah::OP37	95

Ah::OP77	96
Ah::OPA134	97
Ah::OPA604	98
Ah::OpAmpCreator	103
Ah::PhysConst	104
Ah::PhysUnitCplx	105
Ah::Current	46
Ah::Impedance	57
Ah::FreqImpedance	55
Ah::Capacitor	39
Ah::Inductor	60
Ah::Resistor	133
Ah::Power	130
Ah::Voltage	163
Ah::PhysUnitDbl	110
Ah::PickUp	112
Ah::PickUpClearaudioAlphaWood	115
Ah::PickUpClearaudioDiscovery	116
Ah::PickUpClearaudioGammaSGold	117
Ah::PickUpClearaudioVirtuosoWood	118
Ah::PickUpGradoMaster	119
Ah::PickUpGradoPlatinum	120
Ah::PickUpGradoReference	121
Ah::PickUpGradoStatement	122
Ah::PickUpOrtofon540mk2	123
Ah::PickUpVdhBlackBeautySPX	124
Ah::PickUpVdhColibriXGP	125
Ah::PickUpVdhDDT2Special	126
Ah::PickUpVdhFrogCopper	127
Ah::PickUpVdhMc1Special	128
Ah::PickUpVdhMm1	129
Ah::Speaker	138
Ah::Kef_B110_A	70
Ah::Kef_B110_B	71
Ah::Kef_B139_B	72
Ah::Kef_B200_A	73
Ah::Kef_B200_B	74
Ah::Kef_B300_B	75
Ah::Monacor_SP45_4	82
Ah::Monacor_SP45_8	83
Ah::Monacor_SP60_4	84
Ah::Monacor_SP60_8	85
Ah::Monacor_SPP110_4	86
Ah::Monacor_SPP110_8	87
Ah::Visaton_SC_10	161
Ah::SpeakerCreator	142
Ah::Transistor	143
Ah::BPT	36
Ah::FET	50
Ah::TransistorFollower	145
Ah::VierPol	147
Ah::VierPolA	149

Ah::VierPolH	152
Ah::VierPolY	155
Ah::VierPolZ	158

Chapter 4

libAHetech Class Index

4.1 libAHetech Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Ah::AD711 (Operational amplifier AD711)	27
Ah::AD745 (Ultra low noise operational amplifier AD745)	28
Ah::AD797 (Ultra low noise operational amplifier AD797)	29
Ah::AD845 (Fast precision operational amplifier AD845)	30
Ah::Amplifier (Pure virtual generic amplifier base class)	31
Ah::AnalogDevOpAmpCreator (Creator class for Analog Devices operational amplifiers)	33
Ah::Box (Loudspeaker box)	34
Ah::BPT (Bipolar transistor)	36
Ah::BurrBrownOpAmpCreator (Creator class for Burr-Brown operational amplifiers)	38
Ah::Capacitor (Ideal electrical capacitor)	39
Ah::ClosedBox (Closed loudspeaker box)	42
Ah::Cplx (Declaration of a complex number)	45
Ah::Cplx (Declaration of a complex number)	45
Ah::Current (Electrical current)	46
Ah::Divider (Voltage divider)	49
Ah::FET (FET)	50
Ah::Filter2PoleInv (2-pole filter using multiple feedback on inverting amplifier)	53
Ah::FreqImpedance (Frequency dependent electrical impedance base class)	55
Ah::Impedance (Electrical impedance)	57
Ah::Inductor (Ideal electrical inductor)	60
Ah::IntPol (Pure virtual interpolation class)	63
Ah::IntPolData	65
Ah::IntPolLin1Lin2 (Linear interpolation class)	66
Ah::IntPolLog1Lin2 (Logarithmic interpolation class)	67
Ah::InvAmp (Inverting amplifier)	68
Ah::Kef_B110_A (Speaker KEF B 110 SP 1003)	70
Ah::Kef_B110_B (Speaker KEF B 110 SP 1057)	71
Ah::Kef_B139_B (Speaker KEF B 139 SP 1044)	72
Ah::Kef_B200_A (Speaker KEF B 200 SP 1014)	73
Ah::Kef_B200_B (Speaker KEF B 200 SP 1039)	74
Ah::Kef_B300_B (Speaker KEF B 300 SP 1071)	75
Ah::LinTechOpAmpCreator (Creator class for Linear Technology operational amplifiers)	76
Ah::LM833 (Audio operational amplifier LM833)	77

Ah::LT1028 (Very fast high precision operational amplifier LT1028)	78
Ah::LT1115 (Audio operational amplifier LT1115)	79
Ah::MathConst (Some mathical constants)	80
Ah::MiscOpAmpCreator (OpAmp creator class)	81
Ah::Monacor_SP45_4 (Speaker Monacor SP-45/4)	82
Ah::Monacor_SP45_8 (Speaker Monacor SP-45/8)	83
Ah::Monacor_SP60_4 (Speaker Monacor SP-60/4)	84
Ah::Monacor_SP60_8 (Speaker Monacor SP-60/8)	85
Ah::Monacor_SPP110_4 (Speaker Monacor SPP-110/4)	86
Ah::Monacor_SPP110_8 (Speaker Monacor SPP-110/8)	87
Ah::NE5532 (Audio operational amplifier NE5532)	89
Ah::NoneInvAmp (None inverting amplifier)	90
Ah::OP07 (Industry standard precision operational amplifier OP07)	92
Ah::OP177 (Industry standard high precision operational amplifier OP177)	93
Ah::OP27 (Fast industry standard precision operational amplifier OP27)	94
Ah::OP37 (Very fast industry standard precision operational amplifier OP37)	95
Ah::OP77 (Industry standard high precision operational amplifier OP77)	96
Ah::OPA134 (Sound-Plus operational amplifier OPA134)	97
Ah::OPA604 (Audio operational amplifier OPA604)	98
Ah::OpAmp (Operational amplifier)	99
Ah::OpAmpCreator (OpAmp creator class)	103
Ah::PhysConst (Some physical constants)	104
Ah::PhysUnitCplx (Complex physical unit)	105
Ah::PhysUnitDbI (Scalar physical unit)	110
Ah::PickUp (Pickup of a record player)	112
Ah::PickUpClearaudioAlphaWood (Clearaudio Alpha Wood (data supplied by Frank Klemm))	115
Ah::PickUpClearaudioDiscovery (Clearaudio Discovery (data supplied by Frank Klemm))	116
Ah::PickUpClearaudioGammaSGold (Clearaudio Gamma S Gold (data supplied by Frank Klemm))	117
Ah::PickUpClearaudioVirtuosoWood (Clearaudio Virtuoso Wood (data supplied by Frank Klemm))	118
Ah::PickUpGradoMaster (Grado Master (data supplied by Frank Klemm))	119
Ah::PickUpGradoPlatinum (Grado Platinum (data supplied by Frank Klemm))	120
Ah::PickUpGradoReference (Grado Reference (data supplied by Frank Klemm))	121
Ah::PickUpGradoStatement (Grado The Statement (data supplied by Frank Klemm))	122
Ah::PickUpOrtofon540mk2 (Ortofon 540 MK2 (example supplied by Frank Klemm in drmh))	123
Ah::PickUpVdhBlackBeautySPX (Van den Hul Black Beauty SPX (data supplied by Frank Klemm))	124
Ah::PickUpVdhColibriXGP (Van den Hul The Colibri XGP (data supplied by Frank Klemm))	125
Ah::PickUpVdhDDT2Special (Van den Hul DDT II Special (data supplied by Frank Klemm))	126
Ah::PickUpVdhFrogCopper (Van den Hul Frog Copper (data supplied by Frank Klemm))	127
Ah::PickUpVdhMc1Special (Van den Hul MC-1 Special)	128
Ah::PickUpVdhMm1 (Van den Hul MM-1)	129
Ah::Power (Electrical power)	130
Ah::Resistor (Ideal electrical resistor)	133
Ah::SalKey (Sallen-Key Filter)	136
Ah::Speaker (Pure speaker system)	138
Ah::SpeakerCreator (Speaker creator class)	142
Ah::Transistor (Generic transistor)	143
Ah::TransistorFollower (Generic transistor follower)	145
Ah::VierPol	147
Ah::VierPolA	149
Ah::VierPolH	152
Ah::VierPolY	155

Ah::VierPolZ	158
Ah::Visaton_SC_10 (Speaker VISATON SC 10)	161
Ah::Voltage (Electrical voltage)	163

Chapter 5

libAHetech File Index

5.1 libAHetech File List

Here is a list of all files with brief descriptions:

AH/Etech/aaa.h	167
AH/Etech/Algo.h	168
AH/Etech/Amplifier.h	169
AH/Etech/AnalogDevOpAmps.h	170
AH/Etech/AnalogDevOpCreat.h	171
AH/Etech/Box.h	172
AH/Etech/BPT.h	173
AH/Etech/BurrBrownOpAmps.h	174
AH/Etech/BurrBrownOpCreat.h	175
AH/Etech/Capacitor.h	176
AH/Etech/Cd.h	177
AH/Etech/ClosedBox.h	178
AH/Etech/Current.h	179
AH/Etech/Divider.h	180
AH/Etech/Etech.h	181
AH/Etech/FET.h	182
AH/Etech/Filter2PoleInv.h	183
AH/Etech/FreqImp.h	184
AH/Etech/Impedance.h	185
AH/Etech/Inductor.h	186
AH/Etech/IntPol.h	187
AH/Etech/InvAmp.h	188
AH/Etech/KefSpeakers.h	189
AH/Etech/LinTechOpAmps.h	190
AH/Etech/LinTechOpCreat.h	191
AH/Etech/MathConst.h	192
AH/Etech/MiscOpAmps.h	193
AH/Etech/MiscOpCreat.h	194
AH/Etech/MonacorSpeakers.h	195
AH/Etech/NoneInvAmp.h	196
AH/Etech/OpAmp.h	197
AH/Etech/OpAmpCreator.h	198
AH/Etech/PhysConst.h	199

AH/Etech/PhysUnit.h	200
AH/Etech/PickUp.h	201
AH/Etech/Power.h	203
AH/Etech/Resistor.h	204
AH/Etech/Riaa.h	205
AH/Etech/SalKey.h	206
AH/Etech/Speaker.h	207
AH/Etech/SpeakerCreator.h	208
AH/Etech/TransFollow.h	209
AH/Etech/Transistor.h	210
AH/Etech/Types.h	211
AH/Etech/VdHPickUps.h	212
AH/Etech/Version.h	213
AH/Etech/VierPol.h	214
AH/Etech/VisatonSpeakers.h	215
AH/Etech/Voltage.h	216

Chapter 6

libAHetech Directory Documentation

6.1 AH/ Directory Reference

Directories

- directory [Etech](#)

6.2 AH/Etech/ Directory Reference

Files

- file [aaa.h](#)
- file [Algo.h](#)
- file [Amplifier.h](#)
- file [AnalogDevOpAmps.h](#)
- file [AnalogDevOpCreat.h](#)
- file [Box.h](#)
- file [BPT.h](#)
- file [BurrBrownOpAmps.h](#)
- file [BurrBrownOpCreat.h](#)
- file [Capacitor.h](#)
- file [Cd.h](#)
- file [ClosedBox.h](#)
- file [Current.h](#)
- file [Divider.h](#)
- file [Etech.h](#)
- file [FET.h](#)
- file [Filter2PoleInv.h](#)
- file [FreqImp.h](#)
- file [Impedance.h](#)
- file [Inductor.h](#)
- file [IntPol.h](#)
- file [InvAmp.h](#)
- file [KefSpeakers.h](#)
- file [LinTechOpAmps.h](#)
- file [LinTechOpCreat.h](#)
- file [MathConst.h](#)
- file [MiscOpAmps.h](#)
- file [MiscOpCreat.h](#)
- file [MonacorSpeakers.h](#)
- file [NoneInvAmp.h](#)
- file [OpAmp.h](#)
- file [OpAmpCreator.h](#)
- file [PhysConst.h](#)
- file [PhysUnit.h](#)
- file [PickUp.h](#)
- file [Power.h](#)
- file [Resistor.h](#)
- file [Riaa.h](#)
- file [SalKey.h](#)
- file [Speaker.h](#)
- file [SpeakerCreator.h](#)
- file [TransFollow.h](#)
- file [Transistor.h](#)
- file [Types.h](#)
- file [VdHPickUps.h](#)
- file [Version.h](#)
- file [VierPol.h](#)
- file [VisatonSpeakers.h](#)
- file [Voltage.h](#)

Chapter 7

libAHetech Namespace Documentation

7.1 AH Namespace Reference

7.2 Ah Namespace Reference

Classes

- class [Cplx](#)
declaration of a complex number
- class [Amplifier](#)
pure virtual generic amplifier base class
- class [AD711](#)
represents the operational amplifier [AD711](#)
- class [AD745](#)
represents the ultra low noise operational amplifier [AD745](#)
- class [AD797](#)
represents the ultra low noise operational amplifier [AD797](#)
- class [AD845](#)
represents the fast precision operational amplifier [AD845](#)
- class [AnalogDevOpAmpCreator](#)
represents the creator class for Analog Devices operational amplifiers
- class [Box](#)
represents a loudspeaker box
- class [BPT](#)
represents a bipolar transistor
- class [OPA134](#)
represents the Sound-Plus operational amplifier [OPA134](#)
- class [OPA604](#)
represents the audio operational amplifier [OPA604](#)
- class [BurrBrownOpAmpCreator](#)
represents the creator class for Burr-Brown operational amplifiers
- class [Capacitor](#)
represents an ideal electrical capacitor
- class [ClosedBox](#)
represents a closed loudspeaker box
- class [Current](#)
represents an electrical current
- class [Divider](#)

represents a voltage divider

- class [FET](#)
represents a [FET](#)
- class [Filter2PoleInv](#)
2-pole filter using multiple feedback on inverting amplifier
- class [FreqImpedance](#)
represents an frequency dependent electrical impedance base class
- class [Impedance](#)
represents an electrical impedance
- class [Inductor](#)
represents an ideal electrical inductor
- struct [IntPolData](#)
- class [IntPol](#)
represents a pure virtual interpolation class
- class [IntPolLin1Lin2](#)
represents a linear interpolation class
- class [IntPolLog1Lin2](#)
represents a logarithmic interpolation class
- class [InvAmp](#)
represents an inverting amplifier
- class [Kef_B110_A](#)
represents the speaker KEF B 110 SP 1003
- class [Kef_B110_B](#)
represents the speaker KEF B 110 SP 1057
- class [Kef_B200_A](#)
represents the speaker KEF B 200 SP 1014
- class [Kef_B200_B](#)
represents the speaker KEF B 200 SP 1039
- class [Kef_B139_B](#)
represents the speaker KEF B 139 SP 1044
- class [Kef_B300_B](#)
represents the speaker KEF B 300 SP 1071
- class [LT1028](#)
represents the very fast high precision operational amplifier [LT1028](#)

- class [LT1115](#)
represents the audio operational amplifier [LT1115](#)
- class [LinTechOpAmpCreator](#)
represents the creator class for Linear Technology operational amplifiers
- class [MathConst](#)
represents some mathical constants
- class [LM833](#)
represents the audio operational amplifier [LM833](#)
- class [NE5532](#)
represents the audio operational amplifier [NE5532](#)
- class [OP07](#)
represents the industry standard precision operational amplifier [OP07](#)
- class [OP27](#)
represents the fast industry standard precision operational amplifier [OP27](#)
- class [OP37](#)
represents the very fast industry standard precision operational amplifier [OP37](#)
- class [OP77](#)
represents the industry standard high precision operational amplifier [OP77](#)
- class [OP177](#)
represents the industry standard high precision operational amplifier [OP177](#)
- class [MiscOpAmpCreator](#)
represents the [OpAmp](#) creator class
- class [Monacor_SP45_4](#)
represents the speaker Monacor SP-45/4
- class [Monacor_SP45_8](#)
represents the speaker Monacor SP-45/8
- class [Monacor_SP60_4](#)
represents the speaker Monacor SP-60/4
- class [Monacor_SP60_8](#)
represents the speaker Monacor SP-60/8
- class [Monacor_SPP110_4](#)
represents the speaker Monacor SPP-110/4
- class [Monacor_SPP110_8](#)

represents the speaker Monacor SPP-110/8

- class [NoneInvAmp](#)
represents a none inverting amplifier
- class [OpAmp](#)
represents an operational amplifier
- class [OpAmpCreator](#)
represents the [OpAmp](#) creator class
- class [PhysConst](#)
represents some physical constants
- class [PhysUnitDbl](#)
represents a scalar physical unit
- class [PhysUnitCplx](#)
represents a complex physical unit
- class [PickUp](#)
represents a pickup of a record player
- class [PickUpOrtofon540mk2](#)
Ortofon 540 MK2 (example supplied by Frank Klemm in drmh).
- class [PickUpVdhDDT2Special](#)
Van den Hul DDT II Special (data supplied by Frank Klemm).
- class [PickUpVdhFrogCopper](#)
Van den Hul Frog Copper (data supplied by Frank Klemm).
- class [PickUpVdhBlackBeautySPX](#)
Van den Hul Black Beauty SPX (data supplied by Frank Klemm).
- class [PickUpVdhColibriXGP](#)
Van den Hul The Colibri XGP (data supplied by Frank Klemm).
- class [PickUpClearaudioAlphaWood](#)
Clearaudio Alpha Wood (data supplied by Frank Klemm).
- class [PickUpClearaudioVirtuosoWood](#)
Clearaudio Virtuoso Wood (data supplied by Frank Klemm).
- class [PickUpClearaudioGammaSGold](#)
Clearaudio Gamma S Gold (data supplied by Frank Klemm).
- class [PickUpClearaudioDiscovery](#)
Clearaudio Discovery (data supplied by Frank Klemm).

- class [PickUpGradoPlatinum](#)
Grado Platinum (data supplied by Frank Klemm).
- class [PickUpGradoMaster](#)
Grado Master (data supplied by Frank Klemm).
- class [PickUpGradoReference](#)
Grado Reference (data supplied by Frank Klemm).
- class [PickUpGradoStatement](#)
Grado The Statement (data supplied by Frank Klemm).
- class [Power](#)
represents an electrical power
- class [Resistor](#)
represents an ideal electrical resistor
- class [SalKey](#)
Sallen-Key Filter.
- class [Speaker](#)
represents a pure speaker system
- class [SpeakerCreator](#)
represents the speaker creator class
- class [TransistorFollower](#)
represents a generic transistor follower
- class [Transistor](#)
represents a generic transistor
- class [PickUpVdhMc1Special](#)
Van den Hul MC-1 Special.
- class [PickUpVdhMm1](#)
Van den Hul MM-1.
- class [VierPol](#)
- class [VierPolZ](#)
- class [VierPolY](#)
- class [VierPolH](#)
- class [VierPolA](#)
- class [Visaton_SC_10](#)
represents the speaker VISATON SC 10
- class [Voltage](#)
represents an electrical voltage

Typedefs

- typedef double [Dbl](#)
declaration of a scalar number
- typedef std::complex< double > [double_complex](#)

Functions

- [Impedance operator/](#) (const [Voltage](#) &u, const [Current](#) &i)
overloaded operator / for division of { [Voltage](#) } / { [Current](#) }
- [Voltage operator *](#) (const [Impedance](#) &r, const [Current](#) &i)
*overloaded operator * for multiplication of { [Impedance](#) } * { [Current](#) }*
- [Voltage operator *](#) (const [Current](#) &i, const [Impedance](#) &r)
*overloaded operator * for multiplication of { [Current](#) } * { [Impedance](#) }*
- [Voltage operator/](#) (const [Power](#) &p, const [Current](#) &i)
overloaded operator / for division of { [Power](#) } / { [Current](#) }
- [Current operator/](#) (const [Voltage](#) &u, const [Impedance](#) &r)
overloaded operator / for division of { [Voltage](#) } / { [Impedance](#) }
- [Current operator *](#) (const [Power](#) &p, const [Voltage](#) &u)
overloaded operator / for division of { [Power](#) } / { [Voltage](#) }
- [Power operator *](#) (const [Voltage](#) &u, const [Current](#) &i)
*overloaded operator * for multiplication of { [Voltage](#) } * { [Current](#) }*
- [Power operator *](#) (const [Current](#) &i, const [Voltage](#) &u)
*overloaded operator * for multiplication of { [Current](#) } * { [Voltage](#) }*
- [Dbl inVdB](#) ([Dbl](#) x)
return value in voltage decibel
- [Dbl inPdB](#) ([Dbl](#) x)
return value in power decibel
- [Dbl inDeg](#) ([Dbl](#) x)
return phase argument in degree
- [Cplx CdPreEmphasis](#) ([Dbl](#) f)
return CD preemphasis value at frequency { f }
- [Cplx CdDeEmphasis](#) ([Dbl](#) f)
return CD deemphasis value at frequency { f }
- [Cplx R1aaPreEmphasis](#) ([Dbl](#) f)
return R1AA preemphasis value at frequency { f }

- [Cplx RiaaDeEmphasis \(Dbl f\)](#)

return RIAA deemphasis value at frequency { f}

7.2.1 Typedef Documentation

7.2.1.1 typedef double [Ah::Dbl](#)

declaration of a scalar number

{ Dbl} is a scalar number, which is almost everywhere used in the {*Etech*} library. The reason for using this definition is to make it easier using floating point types other than { double} simply by changing this definition.

7.2.1.2 typedef std::complex<double> [Ah::double_complex](#)

7.2.2 Function Documentation

7.2.2.1 [Cplx Ah::CdDeEmphasis \(Dbl f\)](#)

return CD deemphasis value at frequency { f}

7.2.2.2 [Cplx Ah::CdPreEmphasis \(Dbl f\)](#)

return CD preemphasis value at frequency { f}

7.2.2.3 [Dbl Ah::inDeg \(Dbl x\)](#) [inline]

return phase argument in degree

7.2.2.4 [Dbl Ah::inPdB \(Dbl x\)](#) [inline]

return value in power decibel

7.2.2.5 [Dbl Ah::inVdB \(Dbl x\)](#) [inline]

return value in voltage decibel

7.2.2.6 [Power Ah::operator * \(const \[Current\]\(#\) & i, const \[Voltage\]\(#\) & u\)](#) [inline]

overloaded operator * for multiplication of { [Current](#) } * { [Voltage](#) }

7.2.2.7 [Power Ah::operator * \(const \[Voltage\]\(#\) & u, const \[Current\]\(#\) & i\)](#) [inline]

overloaded operator * for multiplication of { [Voltage](#) } * { [Current](#) }

7.2.2.8 **Current** Ah::operator * (const **Power** & *p*, const **Voltage** & *u*) [inline]

overloaded operator / for division of { **Power** } / { **Voltage** }

7.2.2.9 **Voltage** Ah::operator * (const **Current** & *i*, const **Impedance** & *r*) [inline]

overloaded operator * for multiplication of { **Current** } * { **Impedance** }

7.2.2.10 **Voltage** Ah::operator * (const **Impedance** & *r*, const **Current** & *i*) [inline]

overloaded operator * for multiplication of { **Impedance** } * { **Current** }

7.2.2.11 **Current** Ah::operator/ (const **Voltage** & *u*, const **Impedance** & *r*) [inline]

overloaded operator / for division of { **Voltage** } / { **Impedance** }

7.2.2.12 **Voltage** Ah::operator/ (const **Power** & *p*, const **Current** & *i*) [inline]

overloaded operator / for division of { **Power** } / { **Current** }

7.2.2.13 **Impedance** Ah::operator/ (const **Voltage** & *u*, const **Current** & *i*) [inline]

overloaded operator / for division of { **Voltage** } / { **Current** }

7.2.2.14 **Cplx** Ah::RiaaDeEmphasis (**Dbl** *f*)

return RIAA deemphasis value at frequency { *f* }

7.2.2.15 **Cplx** Ah::RiaaPreEmphasis (**Dbl** *f*)

return RIAA preemphasis value at frequency { *f* }

Chapter 8

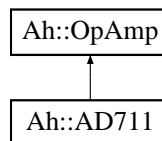
libAHetech Class Documentation

8.1 Ah::AD711 Class Reference

represents the operational amplifier [AD711](#)

```
#include <AnalogDevOpAmps.h>
```

Inheritance diagram for Ah::AD711::



Public Member Functions

- [AD711](#) ()
default constructor

8.1.1 Detailed Description

represents the operational amplifier [AD711](#)

{ [AD711](#) } represents the operational amplifier [AD711](#) which can be used both in DC (direct current) and in AC (alternate current) applications.

8.1.2 Constructor & Destructor Documentation

8.1.2.1 Ah::AD711::AD711 ()

default constructor

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

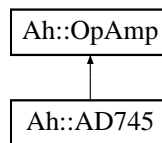
- AH/Etech/[AnalogDevOpAmps.h](#)

8.2 Ah::AD745 Class Reference

represents the ultra low noise operational amplifier [AD745](#)

```
#include <AnalogDevOpAmps.h>
```

Inheritance diagram for Ah::AD745::



Public Member Functions

- [AD745](#) ()
default constructor

8.2.1 Detailed Description

represents the ultra low noise operational amplifier [AD745](#)

{ [AD745](#) } represents the ultra low noise operational amplifier [AD745](#) which can be used both in DC (direct current) and in AC (alternate current) applications.

8.2.2 Constructor & Destructor Documentation

8.2.2.1 Ah::AD745::AD745 ()

default constructor

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

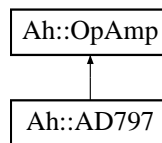
- AH/Etech/[AnalogDevOpAmps.h](#)

8.3 Ah::AD797 Class Reference

represents the ultra low noise operational amplifier [AD797](#)

```
#include <AnalogDevOpAmps.h>
```

Inheritance diagram for Ah::AD797::



Public Member Functions

- [AD797](#) ()
default constructor

8.3.1 Detailed Description

represents the ultra low noise operational amplifier [AD797](#)

{ [AD797](#) } represents the ultra low noise operational amplifier [AD797](#) which can be used both in DC (direct current) and in AC (alternate current) applications.

8.3.2 Constructor & Destructor Documentation

8.3.2.1 Ah::AD797::AD797 ()

default constructor

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

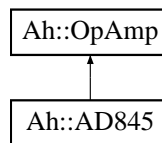
- AH/Etech/[AnalogDevOpAmps.h](#)

8.4 Ah::AD845 Class Reference

represents the fast precision operational amplifier [AD845](#)

```
#include <AnalogDevOpAmps.h>
```

Inheritance diagram for Ah::AD845::



Public Member Functions

- [AD845](#) ()
default constructor

8.4.1 Detailed Description

represents the fast precision operational amplifier [AD845](#)

{ [AD845](#) } represents the fast precision operational amplifier [AD845](#) which can be used both in DC (direct current) and in AC (alternate current) applications.

8.4.2 Constructor & Destructor Documentation

8.4.2.1 Ah::AD845::AD845 ()

default constructor

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

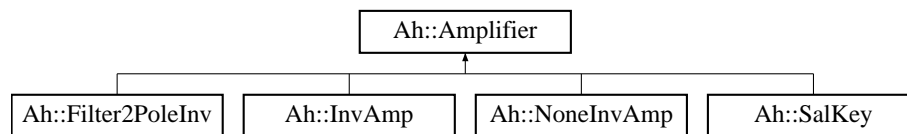
- AH/Etech/[AnalogDevOpAmps.h](#)

8.5 Ah::Amplifier Class Reference

pure virtual generic amplifier base class

```
#include <Amplifier.h>
```

Inheritance diagram for Ah::Amplifier::



Public Member Functions

- virtual [~Amplifier](#) ()
destructor:
- virtual [Cplx getA](#) ([Dbl freq](#))=0
get amplification at frequency { freq}
- virtual [Cplx getA](#) ([Dbl freq](#), const [Impedance](#) &load)=0
get amplification at frequency { freq} with { load}

8.5.1 Detailed Description

pure virtual generic amplifier base class

{ [Amplifier](#) } is a pure virtual amplifier base class which uses an operational amplifier. It can be used both in DC (direct current) and in AC (alternate current) applications.

8.5.2 Constructor & Destructor Documentation

8.5.2.1 virtual Ah::Amplifier::~~Amplifier () [inline, virtual]

destructor:

8.5.3 Member Function Documentation

8.5.3.1 virtual [Cplx](#) Ah::Amplifier::getA ([Dbl freq](#), const [Impedance](#) &load) [pure virtual]

get amplification at frequency { freq} with { load}

Implemented in [Ah::Filter2PoleInv](#), [Ah::InvAmp](#), [Ah::NoneInvAmp](#), and [Ah::SalKey](#).

8.5.3.2 virtual [Cplx](#) Ah::Amplifier::getA ([Dbl freq](#)) [pure virtual]

get amplification at frequency { freq}

Implemented in [Ah::Filter2PoleInv](#), [Ah::InvAmp](#), [Ah::NoneInvAmp](#), and [Ah::SalKey](#).

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

- AH/Etech/[Amplifier.h](#)

8.6 Ah::AnalogDevOpAmpCreator Class Reference

represents the creator class for Analog Devices operational amplifiers

```
#include <AnalogDevOpCreat.h>
```

Public Member Functions

- [OpAmp](#) * [create](#) (const char *model) const
create instance of specified Analog Devices [OpAmp](#)

8.6.1 Detailed Description

represents the creator class for Analog Devices operational amplifiers

{ [AnalogDevOpAmpCreator](#) } represents a creator class for all operational amplifiers made by Analog Devices within { *libEtech* }.

8.6.2 Member Function Documentation

8.6.2.1 [OpAmp](#)* Ah::AnalogDevOpAmpCreator::create (const char * *model*) const

create instance of specified Analog Devices [OpAmp](#)

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

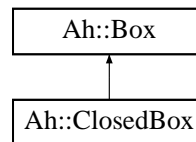
- AH/Etech/[AnalogDevOpCreat.h](#)

8.7 Ah::Box Class Reference

represents a loudspeaker box

```
#include <Box.h>
```

Inheritance diagram for Ah::Box::



Public Member Functions

- [Box](#) ([Speaker](#) &s, const char *n="")
constructor
- virtual [~Box](#) ()
virtual destructor
- const char * [getName](#) () const
get name of box
- const [Speaker](#) & [getSpeaker](#) () const
get used speaker

Protected Attributes

- [Speaker](#) & sp

8.7.1 Detailed Description

represents a loudspeaker box

{ [Box](#) } represents a loudspeaker box.

8.7.2 Constructor & Destructor Documentation

8.7.2.1 [Ah::Box::Box](#) ([Speaker](#) & s, const char * n = "") [inline]

constructor

8.7.2.2 [virtual Ah::Box::~~Box](#) () [inline, virtual]

virtual destructor

8.7.3 Member Function Documentation

8.7.3.1 `const char* Ah::Box::getName () const` [inline]

get name of box

8.7.3.2 `const Speaker& Ah::Box::getSpeaker () const` [inline]

get used speaker

8.7.4 Member Data Documentation

8.7.4.1 `Speaker& Ah::Box::sp` [protected]

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

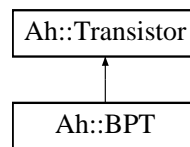
- [AH/Etech/Box.h](#)

8.8 Ah::BPT Class Reference

represents a bipolar transistor

```
#include <BPT.h>
```

Inheritance diagram for Ah::BPT:



Public Member Functions

- [BPT](#) ()
default constructor
- virtual [~BPT](#) ()
destructor
- virtual [Cplx getS](#) (const [Voltage](#) &Ube, [Dbl](#) t=PhysConst_T25) const
get steilheit
- virtual [Cplx getS](#) (const [Current](#) &Ic, [Dbl](#) t=PhysConst_T25) const
get steilheit
- virtual [Current getIout](#) (const [Voltage](#) &Uin, [Dbl](#) t=PhysConst_T25) const
return output current depending on input voltage

Static Public Member Functions

- static [Voltage getUt](#) ([Dbl](#) t=PhysConst_T25)
get Ut at specified temperature

Static Protected Attributes

- static const [Current Ics](#)

8.8.1 Detailed Description

represents a bipolar transistor

{ [BPT](#) } represents a bipolar transistor. It can be used both in DC (direct current) and in AC (alternate current) applications.

8.8.2 Constructor & Destructor Documentation

8.8.2.1 Ah::BPT::BPT () [inline]

default constructor

8.8.2.2 virtual Ah::BPT::~~BPT () [inline, virtual]

destructor

8.8.3 Member Function Documentation

8.8.3.1 virtual Current Ah::BPT::getIout (const Voltage & Uin, Dbl t = PhysConst_T25) const [virtual]

return output current depending on input voltage

Implements [Ah::Transistor](#).

8.8.3.2 virtual Cplx Ah::BPT::getS (const Current & Ic, Dbl t = PhysConst_T25) const [virtual]

get steilheit

Implements [Ah::Transistor](#).

8.8.3.3 virtual Cplx Ah::BPT::getS (const Voltage & Ube, Dbl t = PhysConst_T25) const [virtual]

get steilheit

Implements [Ah::Transistor](#).

8.8.3.4 static Voltage Ah::BPT::getUt (Dbl t = PhysConst_T25) [static]

get Ut at specified temperature

8.8.4 Member Data Documentation

8.8.4.1 const Current Ah::BPT::Ics [static, protected]

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

- [AH/Etech/BPT.h](#)

8.9 Ah::BurrBrownOpAmpCreator Class Reference

represents the creator class for Burr-Brown operational amplifiers

```
#include <BurrBrownOpCreat.h>
```

Public Member Functions

- [OpAmp](#) * [create](#) (const char *model) const
create instance of specified Burr-Brown [OpAmp](#)

8.9.1 Detailed Description

represents the creator class for Burr-Brown operational amplifiers

{ BurrBrownDevOpAmpCreator } represents a creator class for all operational amplifiers made by Burr-Brown within { *libEtech* }.

8.9.2 Member Function Documentation

8.9.2.1 [OpAmp](#)* Ah::BurrBrownOpAmpCreator::create (const char * *model*) const

create instance of specified Burr-Brown [OpAmp](#)

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

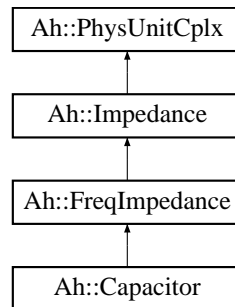
- AH/Etech/[BurrBrownOpCreat.h](#)

8.10 Ah::Capacitor Class Reference

represents an ideal electrical capacitor

```
#include <Capacitor.h>
```

Inheritance diagram for Ah::Capacitor::



Public Member Functions

- [Capacitor](#) ([Dbl](#) c=0.0)
constructor
- [Capacitor](#) (const [Capacitor](#) &c)
copy constructor
- [Dbl](#) [getCap](#) () const
return capacity

Protected Member Functions

- virtual void [calcVal](#) ([Dbl](#) f)

Friends

- [Capacitor](#) [operator+](#) (const [Capacitor](#) &c1, const [Capacitor](#) &c2)
overloaded operator + for serial circuit of two capacitors
- [Capacitor](#) [operator||](#) (const [Capacitor](#) &c1, const [Capacitor](#) &c2)
overloaded operator || for parallel circuit of two capacitors
- [Capacitor](#) [operator *](#) (const [Capacitor](#) &c, [Dbl](#) k)
*overloaded operator * for multiplication of { [Capacitor](#) } * { [Dbl](#) }*
- [Capacitor](#) [operator *](#) ([Dbl](#) k, const [Capacitor](#) &c)
*overloaded operator * for multiplication of { [Dbl](#) } * { [Capacitor](#) }*
- [Capacitor](#) [operator/](#) (const [Capacitor](#) &c, [Dbl](#) k)

overloaded operator / for division of { [Capacitor](#) } / { [Dbl](#) }

- `std::ostream & operator<< (std::ostream &out, const Capacitor &c)`

overloaded operator for output into streams

8.10.1 Detailed Description

represents an ideal electrical capacitor

{ [Capacitor](#) } represents an ideal electrical capacitor which can be used both in DC (direct current) and in AC (alternate current) applications.

8.10.2 Constructor & Destructor Documentation

8.10.2.1 [Ah::Capacitor::Capacitor](#) ([Dbl](#) *c* = 0.0) [[inline](#)]

constructor

8.10.2.2 [Ah::Capacitor::Capacitor](#) (const [Capacitor](#) &*c*) [[inline](#)]

copy constructor

8.10.3 Member Function Documentation

8.10.3.1 `virtual void Ah::Capacitor::calcVal (Dbl f)` [[protected](#), [virtual](#)]

Implements [Ah::FreqImpedance](#).

8.10.3.2 [Dbl](#) [Ah::Capacitor::getCap](#) () const [[inline](#)]

return capacity

8.10.4 Friends And Related Function Documentation

8.10.4.1 [Capacitor](#) operator * ([Dbl](#) *k*, const [Capacitor](#) &*c*) [[friend](#)]

overloaded operator * for multiplication of { [Dbl](#) } * { [Capacitor](#) }

8.10.4.2 [Capacitor](#) operator * (const [Capacitor](#) &*c*, [Dbl](#) *k*) [[friend](#)]

overloaded operator * for multiplication of { [Capacitor](#) } * { [Dbl](#) }

8.10.4.3 [Capacitor](#) operator+ (const [Capacitor](#) &*c1*, const [Capacitor](#) &*c2*) [[friend](#)]

overloaded operator + for serial circuit of two capacitors

8.10.4.4 [Capacitor](#) operator/ (const [Capacitor](#) & *c*, [Dbl](#) *k*) [friend]

overloaded operator / for division of { [Capacitor](#) } / { [Dbl](#) }

8.10.4.5 [std::ostream&](#) operator<< ([std::ostream](#) & *out*, const [Capacitor](#) & *c*) [friend]

overloaded operator for output into streams

8.10.4.6 [Capacitor](#) operator|| (const [Capacitor](#) & *c1*, const [Capacitor](#) & *c2*) [friend]

overloaded operator || for parallel circuit of two capacitors

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

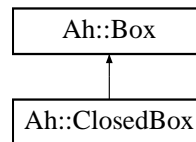
- [AH/Etech/Capacitor.h](#)

8.11 Ah::ClosedBox Class Reference

represents a closed loudspeaker box

```
#include <ClosedBox.h>
```

Inheritance diagram for Ah::ClosedBox::



Public Member Functions

- **ClosedBox** (**Speaker** &s, **Db1** q=0.7, bool d=true, const char *n="")
constructor
- virtual **~ClosedBox** ()
virtual destructor
- **Db1** getQt () const
return damping factor of box-speaker system
- void **calc** (**Db1** &Vb, **Db1** &Fr, **Db1** &F3) const
calculate parameters: box volume, resonance frequency, lower limit

Protected Attributes

- **Db1** Qt
- bool **damped**

8.11.1 Detailed Description

represents a closed loudspeaker box

{ **ClosedBox** } represents a closed loudspeaker box.

8.11.2 Constructor & Destructor Documentation

8.11.2.1 Ah::ClosedBox::ClosedBox (**Speaker** &s, **Db1** q = 0.7, bool d = true, const char * n = "")

constructor

8.11.2.2 virtual Ah::ClosedBox::~~ClosedBox () [inline, virtual]

virtual destructor

8.11.3 Member Function Documentation

8.11.3.1 void Ah::ClosedBox::calc ([Dbl & Vb](#), [Dbl & Fr](#), [Dbl & F3](#)) const

calculate parameters: box volume, resonance frequency, lower limit

8.11.3.2 [Dbl](#) Ah::ClosedBox::getQt () const [inline]

return damping factor of box-speaker system

8.11.4 Member Data Documentation

8.11.4.1 bool [Ah::ClosedBox::damped](#) [protected]

8.11.4.2 [Dbl](#) [Ah::ClosedBox::Qt](#) [protected]

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

- AH/Etech/[ClosedBox.h](#)

8.12 Ah::Cplx Class Reference

declaration of a complex number

```
#include <aaa.h>
```

8.12.1 Detailed Description

declaration of a complex number

{ [Cplx](#) } is a complex number, which is almost everywhere used in the {*Etech*} library. The reason for using this definition is to make it easier using complex types other than { double } simply by changing this definition.

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

- AH/Etech/[aaa.h](#)

8.13 Ah::Cplx Class Reference

declaration of a complex number

```
#include <aaa.h>
```

8.13.1 Detailed Description

declaration of a complex number

{ [Cplx](#) } is a complex number, which is almost everywhere used in the {*Etech*} library. The reason for using this definition is to make it easier using complex types other than { double } simply by changing this definition.

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

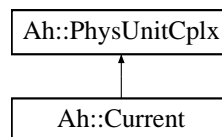
- AH/Etech/[aaa.h](#)

8.14 Ah::Current Class Reference

represents an electrical current

```
#include <Current.h>
```

Inheritance diagram for Ah::Current::



Public Member Functions

- **Current** (Dbl r=0.0, Dbl i=0.0)
constructor
- **Current** (const Cplx &val)
constructor
- **Current** (const Current &i)
copy constructor

Friends

- **Current operator+** (const Current &i1, const Current &i2)
overloaded operator + for addition of two currents
- **Current operator-** (const Current &i1, const Current &i2)
overloaded operator - for subtraction of two currents
- **Current operator *** (const Current &i1, const Cplx &k)
*overloaded operator * for multiplication of { Current } * { Cplx }*
- **Current operator *** (const Cplx &k, const Current &i1)
*overloaded operator * for multiplication of { Cplx } * { Current }*
- **Current operator/** (const Current &i1, const Cplx &k)
overloaded operator / for division of { Current } / { Cplx }
- **bool operator==** (const Current &i1, const Current &i2)
overloaded operator == for comparison of two Currents
- **bool operator!=** (const Current &i1, const Current &i2)
overloaded operator != for comparison of two Currents
- **bool operator<** (const Current &i1, const Current &i2)

overloaded operator < for comparison of two Currents

- `bool operator> (const Current &i1, const Current &i2)`
overloaded operator > for comparison of two Currents

8.14.1 Detailed Description

represents an electrical current

{ `Current` } represents an electrical current which can be used both in DC (direct current) and in AC (alternate current) applications.

8.14.2 Constructor & Destructor Documentation

8.14.2.1 `Ah::Current::Current (Dbl r = 0.0, Dbl i = 0.0)` [inline]

constructor

8.14.2.2 `Ah::Current::Current (const Cplx &val)` [inline]

constructor

8.14.2.3 `Ah::Current::Current (const Current &i)` [inline]

copy constructor

8.14.3 Friends And Related Function Documentation

8.14.3.1 `Current operator * (const Cplx &k, const Current &i1)` [friend]

overloaded operator * for multiplication of { `Cplx` } * { `Current` }

8.14.3.2 `Current operator * (const Current &i1, const Cplx &k)` [friend]

overloaded operator * for multiplication of { `Current` } * { `Cplx` }

8.14.3.3 `bool operator!= (const Current &i1, const Current &i2)` [friend]

overloaded operator != for comparison of two Currents

8.14.3.4 `Current operator+ (const Current &i1, const Current &i2)` [friend]

overloaded operator + for addition of two currents

8.14.3.5 `Current operator- (const Current &i1, const Current &i2)` [friend]

overloaded operator - for subtraction of two currents

8.14.3.6 `Current` `operator/` (`const Current & i1`, `const Cplx & k`) [friend]

overloaded operator / for division of { `Current` } / { `Cplx` }

8.14.3.7 `bool operator<` (`const Current & i1`, `const Current & i2`) [friend]

overloaded operator < for comparison of two Currents

8.14.3.8 `bool operator==` (`const Current & i1`, `const Current & i2`) [friend]

overloaded operator == for comparison of two Currents

8.14.3.9 `bool operator>` (`const Current & i1`, `const Current & i2`) [friend]

overloaded operator > for comparison of two Currents

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

- AH/Etech/[Current.h](#)

8.15 Ah::Divider Class Reference

represents a voltage divider

```
#include <Divider.h>
```

Public Member Functions

- [Divider](#) (const [Impedance](#) &z1, const [Impedance](#) &z2)
constructor
- [Divider](#) (const [Divider](#) &d)
copy constructor
- [Cplx](#) getAtt ()
get attenuation
- [Impedance](#) getZin ()
get input impedance

8.15.1 Detailed Description

represents a voltage divider

{ [Divider](#) } represents a voltage divider existing of two impedances.

8.15.2 Constructor & Destructor Documentation

8.15.2.1 Ah::Divider::Divider (const [Impedance](#) & z1, const [Impedance](#) & z2) [inline]

constructor

8.15.2.2 Ah::Divider::Divider (const [Divider](#) & d) [inline]

copy constructor

8.15.3 Member Function Documentation

8.15.3.1 [Cplx](#) Ah::Divider::getAtt () [inline]

get attenuation

8.15.3.2 [Impedance](#) Ah::Divider::getZin () [inline]

get input impedance

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

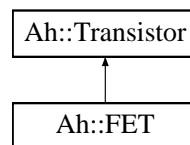
- AH/Etech/[Divider.h](#)

8.16 Ah::FET Class Reference

represents a [FET](#)

```
#include <FET.h>
```

Inheritance diagram for Ah::FET:



Public Member Functions

- [FET](#) (const [Current](#) &i_ds)
default constructor
- [FET](#) (const [FET](#) &t)
copy constructor
- virtual [~FET](#) ()
destructor:
- virtual [Cplx](#) getS (const [Voltage](#) &Ugs, [Dbl](#) t=PhysConst_T25) const
get steilheit
- virtual [Cplx](#) getS (const [Current](#) &Id, [Dbl](#) t=PhysConst_T25) const
get steilheit
- virtual [Current](#) getIout (const [Voltage](#) &Uin, [Dbl](#) t=PhysConst_T25) const
return output current depending on input voltage

Static Public Member Functions

- static [Voltage](#) getUp ([Dbl](#) t=PhysConst_T25)
get Up at specified temperature

Protected Attributes

- [Current](#) Ids

Static Protected Attributes

- static const [Voltage](#) Up25

8.16.1 Detailed Description

represents a [FET](#)

{ [FET](#) } represents a field effect transistor. It can be used both in DC (direct current) and in AC (alternate current) applications.

8.16.2 Constructor & Destructor Documentation

8.16.2.1 Ah::FET::FET (const [Current](#) & *i_ds*) [inline]

default constructor

8.16.2.2 Ah::FET::FET (const [FET](#) & *t*) [inline]

copy constructor

8.16.2.3 virtual Ah::FET::~~FET () [inline, virtual]

destructor:

8.16.3 Member Function Documentation

8.16.3.1 virtual [Current](#) Ah::FET::getIout (const [Voltage](#) & *Uin*, [Dbl](#) *t* = PhysConst_T25) const [virtual]

return output current depending on input voltage

Implements [Ah::Transistor](#).

8.16.3.2 virtual [Cplx](#) Ah::FET::getS (const [Current](#) & *Id*, [Dbl](#) *t* = PhysConst_T25) const [virtual]

get steilheit

Implements [Ah::Transistor](#).

8.16.3.3 virtual [Cplx](#) Ah::FET::getS (const [Voltage](#) & *Ugs*, [Dbl](#) *t* = PhysConst_T25) const [virtual]

get steilheit

Implements [Ah::Transistor](#).

8.16.3.4 static [Voltage](#) Ah::FET::getUp ([Dbl](#) *t* = PhysConst_T25) [static]

get Up at specified temperature

8.16.4 Member Data Documentation

8.16.4.1 [Current Ah::FET::Ids](#) [protected]

8.16.4.2 `const` [Voltage Ah::FET::Up25](#) [static, protected]

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

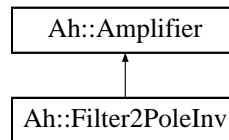
- AH/Etech/[FET.h](#)

8.17 Ah::Filter2PoleInv Class Reference

2-pole filter using multiple feedback on inverting amplifier

```
#include <Filter2PoleInv.h>
```

Inheritance diagram for Ah::Filter2PoleInv::



Public Member Functions

- **Filter2PoleInv** (const **OpAmp** &opamp, const **Impedance** &z1, const **Impedance** &z2, const **Impedance** &z3, const **Impedance** &z4, const **Impedance** &z5)

constructors:

- virtual **~Filter2PoleInv** ()

destructor:

- virtual **Cplx getA** (**Dbf** freq)

get amplification at frequency { freq}:

- virtual **Cplx getA** (**Dbf** freq, const **Impedance** &)

get amplification at frequency { freq} with output { load}:

Static Public Member Functions

- static bool **calcLowpass** (**Dbf** fg, **Dbf** A0, **Dbf** a1, **Dbf** b1, const **Capacitor** &C1, const **Capacitor** &C2, **Resistor** &R1, **Resistor** &R2, **Resistor** &R3)

calculate components

8.17.1 Detailed Description

2-pole filter using multiple feedback on inverting amplifier

{ **Filter2PoleInv** } is a 2 pole filter realized with an inverting (operational) amplifier having multiple feedback paths. {verbatim} _____ ||| Z2 Z5 ||||-\ | In: -Z1-*--Z24-*--|- \ ||| — Out Z3 -|+ / |||/_ Gnd Gnd {verbatim} Typical applications are as follows: {description} [Lowpass:] Z1, Z2 and Z4 are resistors, Z3 and Z5 are capacitors, e.g:\ Z1=R1, Z2=R2, Z3=C2, Z4=R3, Z5=C1\ \$R_1=-R_2/A_0\$ [Highpass:] Z3 and Z5 are resistors, Z1, Z2 and Z4 are capacitors. [Bandpass:] Z1, Z3 and Z5 are resistors, Z2 and Z4 are capacitors. {description}

8.17.2 Constructor & Destructor Documentation

8.17.2.1 `Ah::Filter2PoleInv::Filter2PoleInv (const OpAmp & opamp, const Impedance & z1, const Impedance & z2, const Impedance & z3, const Impedance & z4, const Impedance & z5)`

constructors:

8.17.2.2 `virtual Ah::Filter2PoleInv::~~Filter2PoleInv ()` `[inline, virtual]`

destructor:

8.17.3 Member Function Documentation

8.17.3.1 `static bool Ah::Filter2PoleInv::calcLowpass (Dbl fg, Dbl A0, Dbl a1, Dbl b1, const Capacitor & C1, const Capacitor & C2, Resistor & R1, Resistor & R2, Resistor & R3)` `[static]`

calculate components

8.17.3.2 `virtual Cplx Ah::Filter2PoleInv::getA (Dbl freq, const Impedance &)` `[inline, virtual]`

get amplification at frequency { *freq*} with output { *load*}:

Implements [Ah::Amplifier](#).

8.17.3.3 `virtual Cplx Ah::Filter2PoleInv::getA (Dbl freq)` `[virtual]`

get amplification at frequency { *freq*}:

Implements [Ah::Amplifier](#).

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

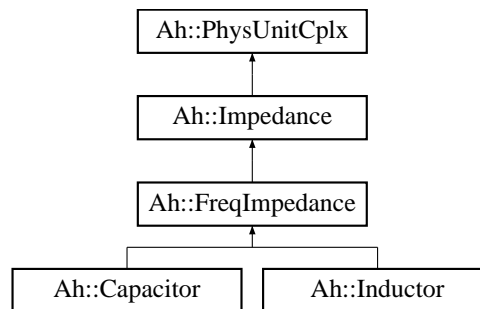
- AH/Etech/[Filter2PoleInv.h](#)

8.18 Ah::FreqImpedance Class Reference

represents an frequency dependent electrical impedance base class

```
#include <FreqImp.h>
```

Inheritance diagram for Ah::FreqImpedance::



Public Member Functions

- [FreqImpedance](#) ([Dbl](#) r=0.0, [Dbl](#) i=0.0)
constructor
- [FreqImpedance](#) (const [Cplx](#) &val)
constructor
- [FreqImpedance](#) (const [FreqImpedance](#) &f)
copy constructor
- virtual [~FreqImpedance](#) ()
destructor
- virtual const [Impedance](#) & [operator\(\)](#) ([Dbl](#) f)
overloaded operator () for returning the impedance at frequency { f}

Protected Member Functions

- virtual void [calcVal](#) ([Dbl](#) f)=0

8.18.1 Detailed Description

represents an frequency dependent electrical impedance base class

{ [FreqImpedance](#) } represents an electrical impedance which is depending on the frequency. It is a pure virtual base class serving the implementation of capacitors and coils.

8.18.2 Constructor & Destructor Documentation

8.18.2.1 `Ah::FreqImpedance::FreqImpedance (Dbl r = 0.0, Dbl i = 0.0)` [inline]

constructor

8.18.2.2 `Ah::FreqImpedance::FreqImpedance (const Cplx & val)` [inline]

constructor

8.18.2.3 `Ah::FreqImpedance::FreqImpedance (const FreqImpedance & f)` [inline]

copy constructor

8.18.2.4 `virtual Ah::FreqImpedance::~~FreqImpedance ()` [inline, virtual]

destructor

8.18.3 Member Function Documentation

8.18.3.1 `virtual void Ah::FreqImpedance::calcVal (Dbl f)` [protected, pure virtual]

Implemented in [Ah::Capacitor](#), and [Ah::Inductor](#).

8.18.3.2 `virtual const Impedance& Ah::FreqImpedance::operator() (Dbl f)` [inline, virtual]

overloaded operator () for returning the impedance at frequency { *f* }

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

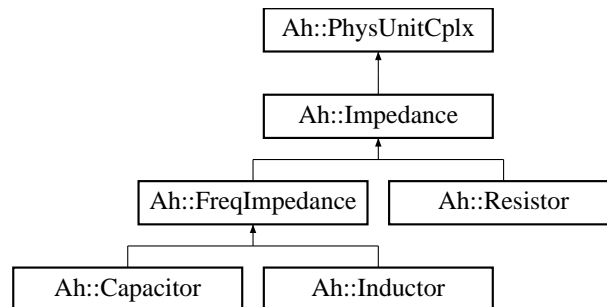
- [AH/Etech/FreqImp.h](#)

8.19 Ah::Impedance Class Reference

represents an electrical impedance

```
#include <Impedance.h>
```

Inheritance diagram for Ah::Impedance::



Public Member Functions

- [Impedance](#) ([Dbl](#) r=0.0, [Dbl](#) i=0.0)
constructor
- [Impedance](#) (const [Cplx](#) &val)
constructor
- [Impedance](#) (const [Impedance](#) &i)
copy constructor

Friends

- [Impedance operator+](#) (const [Impedance](#) &z1, const [Impedance](#) &z2)
overloaded operator + for serial circuit of two impedances
- [Impedance operator||](#) (const [Impedance](#) &z1, const [Impedance](#) &z2)
overloaded operator || for parallel circuit of two impedances
- [Impedance operator-](#) (const [Impedance](#) &z1, const [Impedance](#) &z2)
overloaded operator - for subtraction of two impedances
- [Impedance operator *](#) (const [Impedance](#) &z, [Dbl](#) k)
*overloaded operator * for multiplication of { [Impedance](#) } * { [Dbl](#) }*
- [Impedance operator *](#) ([Dbl](#) k, const [Impedance](#) &z)
*overloaded operator * for multiplication of { [Dbl](#) } * { [Impedance](#) }*
- [Impedance operator *](#) (const [Impedance](#) &z, const [Cplx](#) &k)
*overloaded operator * for multiplication of { [Impedance](#) } * { [Cplx](#) }*

- **Impedance operator *** (const **Cplx** &k, const **Impedance** &z)
*overloaded operator * for multiplication of { Cplx } * { Impedance }*
- **Impedance operator/** (const **Impedance** &z1, const **Cplx** &k)
overloaded operator / for division of { Impedance } / { Cplx }
- **bool operator==** (const **Impedance** &z1, const **Impedance** &z2)
overloaded operator == for comparison of two Impedances
- **bool operator!=** (const **Impedance** &z1, const **Impedance** &z2)
overloaded operator != for comparison of two Impedances
- **bool operator<** (const **Impedance** &z1, const **Impedance** &z2)
overloaded operator < for comparison of two Impedances
- **bool operator>** (const **Impedance** &z1, const **Impedance** &z2)
overloaded operator > for comparison of two Impedances

8.19.1 Detailed Description

represents an electrical impedance

{ **Impedance** } represents an electrical impedance which can be used both in DC (direct current) and in AC (alternate current) applications.

8.19.2 Constructor & Destructor Documentation

8.19.2.1 **Ah::Impedance::Impedance** (**Dbl** *r* = 0.0, **Dbl** *i* = 0.0) [inline]

constructor

8.19.2.2 **Ah::Impedance::Impedance** (const **Cplx** &*val*) [inline]

constructor

8.19.2.3 **Ah::Impedance::Impedance** (const **Impedance** &*i*) [inline]

copy constructor

8.19.3 Friends And Related Function Documentation

8.19.3.1 **Impedance operator *** (const **Cplx** &*k*, const **Impedance** &*z*) [friend]

overloaded operator * for multiplication of { **Cplx** } * { **Impedance** }

8.19.3.2 Impedance operator * (const Impedance & z, const Cplx & k) [friend]

overloaded operator * for multiplication of { Impedance } * { Cplx }

8.19.3.3 Impedance operator * (Dbl k, const Impedance & z) [friend]

overloaded operator * for multiplication of { Dbl } * { Impedance }

8.19.3.4 Impedance operator * (const Impedance & z, Dbl k) [friend]

overloaded operator * for multiplication of { Impedance } * { Dbl }

8.19.3.5 bool operator!= (const Impedance & z1, const Impedance & z2) [friend]

overloaded operator != for comparison of two Impedances

8.19.3.6 Impedance operator+ (const Impedance & z1, const Impedance & z2) [friend]

overloaded operator + for serial circuit of two impedances

8.19.3.7 Impedance operator- (const Impedance & z1, const Impedance & z2) [friend]

overloaded operator - for subtraction of two impedances

8.19.3.8 Impedance operator/ (const Impedance & z1, const Cplx & k) [friend]

overloaded operator / for division of { Impedance } / { Cplx }

8.19.3.9 bool operator< (const Impedance & z1, const Impedance & z2) [friend]

overloaded operator < for comparison of two Impedances

8.19.3.10 bool operator== (const Impedance & z1, const Impedance & z2) [friend]

overloaded operator == for comparison of two Impedances

8.19.3.11 bool operator> (const Impedance & z1, const Impedance & z2) [friend]

overloaded operator > for comparison of two Impedances

8.19.3.12 Impedance operator|| (const Impedance & z1, const Impedance & z2) [friend]

overloaded operator || for parallel circuit of two impedances

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

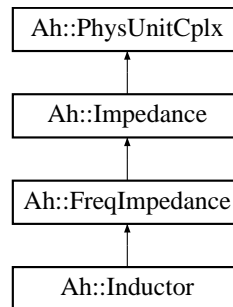
- AH/Etech/[Impedance.h](#)

8.20 Ah::Inductor Class Reference

represents an ideal electrical inductor

```
#include <Inductor.h>
```

Inheritance diagram for Ah::Inductor::



Public Member Functions

- **Inductor** (**Dbl** l=0.0)
constructor
- **Inductor** (const **Inductor** &l)
copy constructor
- **Dbl** getInd () const
return inductivity

Protected Member Functions

- virtual void **calcVal** (**Dbl** f)

Friends

- **Inductor operator+** (const **Inductor** &l1, const **Inductor** &l2)
overloaded operator + for serial circuit of two inductors
- **Inductor operator||** (const **Inductor** &l1, const **Inductor** &l2)
overloaded operator || for parallel circuit of two inductors
- **Inductor operator *** (const **Inductor** &l, **Dbl** k)
*overloaded operator * for multiplication of { Inductor } * { Dbl }*
- **Inductor operator *** (**Dbl** k, const **Inductor** &l)
*overloaded operator * for multiplication of { Dbl } * { Inductor }*
- **Inductor operator/** (const **Inductor** &l, **Dbl** k)

overloaded operator / for division of { [Inductor](#) } / { [Dbl](#) }

- `std::ostream & operator<< (std::ostream &out, const Inductor &i)`

overloaded operator for output into streams

8.20.1 Detailed Description

represents an ideal electrical inductor

{ [Inductor](#) } represents an ideal electrical inductor which can be used both in DC (direct current) and in AC (alternate current) applications.

8.20.2 Constructor & Destructor Documentation

8.20.2.1 [Ah::Inductor::Inductor](#) ([Dbl](#) *l* = 0.0) [`inline`]

constructor

8.20.2.2 [Ah::Inductor::Inductor](#) (const [Inductor](#) &*l*) [`inline`]

copy constructor

8.20.3 Member Function Documentation

8.20.3.1 `virtual void Ah::Inductor::calcVal (Dbl f)` [`protected`, `virtual`]

Implements [Ah::FreqImpedance](#).

8.20.3.2 `Dbl Ah::Inductor::getInd () const` [`inline`]

return inductivity

8.20.4 Friends And Related Function Documentation

8.20.4.1 `Inductor operator * (Dbl k, const Inductor &l)` [`friend`]

overloaded operator * for multiplication of { [Dbl](#) } * { [Inductor](#) }

8.20.4.2 `Inductor operator * (const Inductor &l, Dbl k)` [`friend`]

overloaded operator * for multiplication of { [Inductor](#) } * { [Dbl](#) }

8.20.4.3 `Inductor operator+ (const Inductor &l1, const Inductor &l2)` [`friend`]

overloaded operator + for serial circuit of two inductors

8.20.4.4 Inductor operator/ (const Inductor & *l*, Dbl *k*) [friend]

overloaded operator / for division of { Inductor } / { Dbl }

8.20.4.5 std::ostream& operator<< (std::ostream & *out*, const Inductor & *i*) [friend]

overloaded operator for output into streams

8.20.4.6 Inductor operator|| (const Inductor & *l1*, const Inductor & *l2*) [friend]

overloaded operator || for parallel circuit of two inductors

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

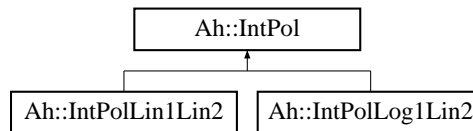
- AH/Etech/[Inductor.h](#)

8.21 Ah::IntPol Class Reference

represents a pure virtual interpolation class

```
#include <IntPol.h>
```

Inheritance diagram for Ah::IntPol::



Public Member Functions

- [IntPol](#) ([IntPolData](#) *data, unsigned int size)
constructor
- virtual [~IntPol](#) ()
virtual destructor
- virtual [Dbl](#) [getY](#) ([Dbl](#) x) const
return interpolated value

Protected Member Functions

- virtual [Dbl](#) [interpolate](#) ([Dbl](#) x, const [IntPolData](#) &p, const [IntPolData](#) &q) const =0

Protected Attributes

- std::vector< [IntPolData](#) > [table](#)

8.21.1 Detailed Description

represents a pure virtual interpolation class

{ [IntPol](#) } represents an pure interpolation class.

8.21.2 Constructor & Destructor Documentation

8.21.2.1 Ah::IntPol::IntPol ([IntPolData](#) * data, unsigned int size)

constructor

8.21.2.2 virtual Ah::IntPol::~~IntPol () [virtual]

virtual destructor

8.21.3 Member Function Documentation

8.21.3.1 `virtual Dbl Ah::IntPol::getY (Dbl x) const` `[virtual]`

return interpolated value

8.21.3.2 `virtual Dbl Ah::IntPol::interpolate (Dbl x, const IntPolData & p, const IntPolData & q) const` `[protected, pure virtual]`

Implemented in [Ah::IntPolLin1Lin2](#), and [Ah::IntPolLog1Lin2](#).

8.21.4 Member Data Documentation

8.21.4.1 `std::vector<IntPolData> Ah::IntPol::table` `[protected]`

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

- [AH/Etech/IntPol.h](#)

8.22 Ah::IntPolData Struct Reference

```
#include <IntPol.h>
```

Public Attributes

- [Dbl x](#)
- [Dbl y](#)

8.22.1 Member Data Documentation

8.22.1.1 [Dbl Ah::IntPolData::x](#)

8.22.1.2 [Dbl Ah::IntPolData::y](#)

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

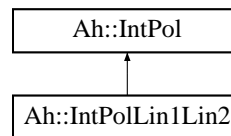
- AH/Etech/[IntPol.h](#)

8.23 Ah::IntPolLin1Lin2 Class Reference

represents a linear interpolation class

```
#include <IntPol.h>
```

Inheritance diagram for Ah::IntPolLin1Lin2::



Public Member Functions

- [IntPolLin1Lin2](#) ([IntPolData](#) *data, unsigned int size)
constructor

Protected Member Functions

- virtual [Dbl](#) [interpolate](#) ([Dbl](#) x, const [IntPolData](#) &p, const [IntPolData](#) &q) const

8.23.1 Detailed Description

represents a linear interpolation class

{ [IntPolLin1Lin2](#) } represents a linear interpolation class.

8.23.2 Constructor & Destructor Documentation

8.23.2.1 Ah::IntPolLin1Lin2::IntPolLin1Lin2 ([IntPolData](#) * data, unsigned int size) [inline]

constructor

8.23.3 Member Function Documentation

8.23.3.1 virtual [Dbl](#) Ah::IntPolLin1Lin2::interpolate ([Dbl](#) x, const [IntPolData](#) & p, const [IntPolData](#) & q) const [protected, virtual]

Implements [Ah::IntPol](#).

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

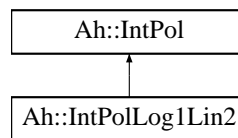
- AH/Etech/[IntPol.h](#)

8.24 Ah::IntPolLog1Lin2 Class Reference

represents a logarithmic interpolation class

```
#include <IntPol.h>
```

Inheritance diagram for Ah::IntPolLog1Lin2::



Public Member Functions

- [IntPolLog1Lin2](#) ([IntPolData](#) *data, unsigned int size)
constructor

Protected Member Functions

- virtual [Dbl](#) [interpolate](#) ([Dbl](#) x, const [IntPolData](#) &p, const [IntPolData](#) &q) const

8.24.1 Detailed Description

represents a logarithmic interpolation class

{ [IntPolLog1Lin2](#) } represents a single logarithmic interpolation class.

8.24.2 Constructor & Destructor Documentation

8.24.2.1 Ah::IntPolLog1Lin2::IntPolLog1Lin2 ([IntPolData](#) * data, unsigned int size) [inline]

constructor

8.24.3 Member Function Documentation

8.24.3.1 virtual [Dbl](#) Ah::IntPolLog1Lin2::interpolate ([Dbl](#) x, const [IntPolData](#) & p, const [IntPolData](#) & q) const [protected, virtual]

Implements [Ah::IntPol](#).

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

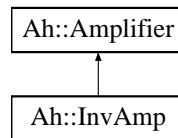
- AH/Etech/[IntPol.h](#)

8.25 Ah::InvAmp Class Reference

represents an inverting amplifier

```
#include <InvAmp.h>
```

Inheritance diagram for Ah::InvAmp::



Public Member Functions

- **InvAmp** (const **Impedance** &z1, const **Impedance** &z2)
constructor
- **InvAmp** (const **OpAmp** &opamp, const **Impedance** &z1, const **Impedance** &z2)
constructor
- virtual **~InvAmp** ()
destructor:
- virtual **Cplx** **getA** (**Dbf** freq)
get amplification at frequency { freq}
- virtual **Cplx** **getA** (**Dbf** freq, const **Impedance** &load)
get amplification at frequency { freq} with { load}

8.25.1 Detailed Description

represents an inverting amplifier

{ **NoneInvAmp** } is an inverting amplifier which uses an operational amplifier. It can be used both in DC (direct current) and in AC (alternate current) applications.

8.25.2 Constructor & Destructor Documentation

8.25.2.1 Ah::InvAmp::InvAmp (const **Impedance** &z1, const **Impedance** &z2) [inline]

constructor

8.25.2.2 Ah::InvAmp::InvAmp (const **OpAmp** &opamp, const **Impedance** &z1, const **Impedance** &z2) [inline]

constructor

8.25.2.3 virtual Ah::InvAmp::~InvAmp () [inline, virtual]

destructor:

8.25.3 Member Function Documentation

8.25.3.1 virtual **Cplx** Ah::InvAmp::getA (**Dbl** *freq*, const **Impedance** & *load*) [virtual]

get amplification at frequency { *freq*} with { *load*}

Implements [Ah::Amplifier](#).

8.25.3.2 virtual **Cplx** Ah::InvAmp::getA (**Dbl** *freq*) [virtual]

get amplification at frequency { *freq*}

Implements [Ah::Amplifier](#).

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

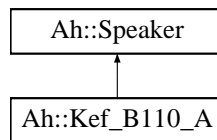
- AH/Etech/[InvAmp.h](#)

8.26 Ah::Kef_B110_A Class Reference

represents the speaker KEF B 110 SP 1003

```
#include <KefSpeakers.h>
```

Inheritance diagram for Ah::Kef_B110_A::



Public Member Functions

- [Kef_B110_A \(\)](#)
default constructor
- virtual [~Kef_B110_A \(\)](#)
destructor

8.26.1 Detailed Description

represents the speaker KEF B 110 SP 1003

{ Kef} represents the small bass/midrange speaker B 110 SP 1003 (B110-A) which is made by KEF.

8.26.2 Constructor & Destructor Documentation

8.26.2.1 Ah::Kef_B110_A::Kef_B110_A ()

default constructor

8.26.2.2 virtual Ah::Kef_B110_A::~~Kef_B110_A () [inline, virtual]

destructor

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

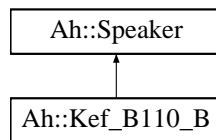
- AH/Etech/[KefSpeakers.h](#)

8.27 Ah::Kef_B110_B Class Reference

represents the speaker KEF B 110 SP 1057

```
#include <KefSpeakers.h>
```

Inheritance diagram for Ah::Kef_B110_B::



Public Member Functions

- [Kef_B110_B \(\)](#)
default constructor
- virtual [~Kef_B110_B \(\)](#)
destructor

8.27.1 Detailed Description

represents the speaker KEF B 110 SP 1057

{ Kef} represents the small bass/midrange speaker B 110 SP 1057 (B110-B) which is made by KEF.

8.27.2 Constructor & Destructor Documentation

8.27.2.1 Ah::Kef_B110_B::Kef_B110_B ()

default constructor

8.27.2.2 virtual Ah::Kef_B110_B::~~Kef_B110_B () [inline, virtual]

destructor

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

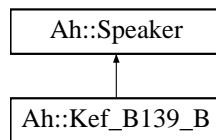
- AH/Etech/[KefSpeakers.h](#)

8.28 Ah::Kef_B139_B Class Reference

represents the speaker KEF B 139 SP 1044

```
#include <KefSpeakers.h>
```

Inheritance diagram for Ah::Kef_B139_B::



Public Member Functions

- [Kef_B139_B \(\)](#)
default constructor
- virtual [~Kef_B139_B \(\)](#)
destructor

8.28.1 Detailed Description

represents the speaker KEF B 139 SP 1044

{ Kef} represents the flat oval bass speaker B 139 SP 1044 (B139-B) which is made by KEF.

8.28.2 Constructor & Destructor Documentation

8.28.2.1 Ah::Kef_B139_B::Kef_B139_B ()

default constructor

8.28.2.2 virtual Ah::Kef_B139_B::~~Kef_B139_B () [inline, virtual]

destructor

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

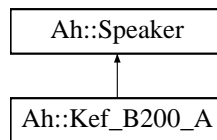
- AH/Etech/[KefSpeakers.h](#)

8.29 Ah::Kef_B200_A Class Reference

represents the speaker KEF B 200 SP 1014

```
#include <KefSpeakers.h>
```

Inheritance diagram for Ah::Kef_B200_A::



Public Member Functions

- [Kef_B200_A \(\)](#)
default constructor
- virtual [~Kef_B200_A \(\)](#)
destructor

8.29.1 Detailed Description

represents the speaker KEF B 200 SP 1014

{ Kef} represents the bass/midrange speaker B 200 SP 1014 (B200-A) which is made by KEF.

8.29.2 Constructor & Destructor Documentation

8.29.2.1 Ah::Kef_B200_A::Kef_B200_A ()

default constructor

8.29.2.2 virtual Ah::Kef_B200_A::~~Kef_B200_A () [inline, virtual]

destructor

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

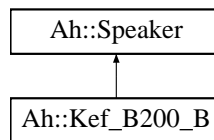
- AH/Etech/[KefSpeakers.h](#)

8.30 Ah::Kef_B200_B Class Reference

represents the speaker KEF B 200 SP 1039

```
#include <KefSpeakers.h>
```

Inheritance diagram for Ah::Kef_B200_B::



Public Member Functions

- [Kef_B200_B \(\)](#)
default constructor
- virtual [~Kef_B200_B \(\)](#)
destructor

8.30.1 Detailed Description

represents the speaker KEF B 200 SP 1039

{ Kef} represents the bass/midrange speaker B 200 SP 1039 (B200-B) which is made by KEF.

8.30.2 Constructor & Destructor Documentation

8.30.2.1 Ah::Kef_B200_B::Kef_B200_B ()

default constructor

8.30.2.2 virtual Ah::Kef_B200_B::~~Kef_B200_B () [inline, virtual]

destructor

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

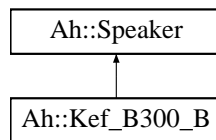
- AH/Etech/[KefSpeakers.h](#)

8.31 Ah::Kef_B300_B Class Reference

represents the speaker KEF B 300 SP 1071

```
#include <KefSpeakers.h>
```

Inheritance diagram for Ah::Kef_B300_B::



Public Member Functions

- [Kef_B300_B \(\)](#)
default constructor
- virtual [~Kef_B300_B \(\)](#)
destructor

8.31.1 Detailed Description

represents the speaker KEF B 300 SP 1071

{ Kef} represents the flat oval bass speaker B 300 SP 1071 (B300-B) which is made by KEF.

8.31.2 Constructor & Destructor Documentation

8.31.2.1 Ah::Kef_B300_B::Kef_B300_B ()

default constructor

8.31.2.2 virtual Ah::Kef_B300_B::~~Kef_B300_B () [inline, virtual]

destructor

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

- AH/Etech/[KefSpeakers.h](#)

8.32 Ah::LinTechOpAmpCreator Class Reference

represents the creator class for Linear Technology operational amplifiers

```
#include <LinTechOpCreat.h>
```

Public Member Functions

- [OpAmp](#) * [create](#) (const char *model) const
create instance of specified Linear Technology [OpAmp](#)

8.32.1 Detailed Description

represents the creator class for Linear Technology operational amplifiers

{ BurrBrownDevOpAmpCreator } represents a creator class for all operational amplifiers made by Linear Technology within { *libEtech* }.

8.32.2 Member Function Documentation

8.32.2.1 [OpAmp](#)* Ah::LinTechOpAmpCreator::create (const char * *model*) const

create instance of specified Linear Technology [OpAmp](#)

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

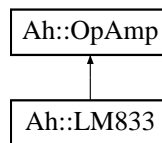
- AH/Etech/[LinTechOpCreat.h](#)

8.33 Ah::LM833 Class Reference

represents the audio operational amplifier [LM833](#)

```
#include <MiscOpAmps.h>
```

Inheritance diagram for Ah::LM833:



Public Member Functions

- [LM833](#) ()
default constructor

8.33.1 Detailed Description

represents the audio operational amplifier [LM833](#)

{ [LM833](#) } represents the audio operational amplifier [LM833](#) which can be used both in DC (direct current) and in AC (alternate current) applications.

8.33.2 Constructor & Destructor Documentation

8.33.2.1 Ah::LM833::LM833 ()

default constructor

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

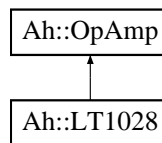
- AH/Etech/[MiscOpAmps.h](#)

8.34 Ah::LT1028 Class Reference

represents the very fast high precision operational amplifier [LT1028](#)

```
#include <LinTechOpAmps.h>
```

Inheritance diagram for Ah::LT1028::



Public Member Functions

- [LT1028](#) ()
default constructor

8.34.1 Detailed Description

represents the very fast high precision operational amplifier [LT1028](#)

{ [LT1028](#) } represents the very fast high precision operational amplifier [LT1028](#) made by Linear Technology which can be used both in DC (direct current) and in AC (alternate current) applications.

8.34.2 Constructor & Destructor Documentation

8.34.2.1 Ah::LT1028::LT1028 ()

default constructor

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

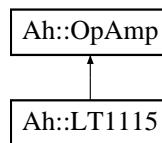
- AH/Etech/[LinTechOpAmps.h](#)

8.35 Ah::LT1115 Class Reference

represents the audio operational amplifier [LT1115](#)

```
#include <LinTechOpAmps.h>
```

Inheritance diagram for Ah::LT1115::



Public Member Functions

- [LT1115](#) ()
default constructor

8.35.1 Detailed Description

represents the audio operational amplifier [LT1115](#)

{ [LT1115](#) } represents the audio operational amplifier [LT1115](#) made by Linear Technology which can be used both in DC (direct current) and in AC (alternate current) applications.

8.35.2 Constructor & Destructor Documentation

8.35.2.1 Ah::LT1115::LT1115 ()

default constructor

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

- AH/Etech/[LinTechOpAmps.h](#)

8.36 Ah::MathConst Class Reference

represents some mathical constants

```
#include <MathConst.h>
```

Static Public Attributes

- static const [Dbl Pi](#) = 3.14159265359
Pi.

8.36.1 Detailed Description

represents some mathical constants

{ [MathConst](#) } is a class containing some mathematical constants.

8.36.2 Member Data Documentation

8.36.2.1 `const Dbl Ah::MathConst::Pi = 3.14159265359` `[static]`

Pi.

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

- AH/Etech/[MathConst.h](#)

8.37 Ah::MiscOpAmpCreator Class Reference

represents the [OpAmp](#) creator class

```
#include <MiscOpCreat.h>
```

Public Member Functions

- [OpAmp](#) * [create](#) (const char *model) const
create instance of specified [OpAmp](#)

8.37.1 Detailed Description

represents the [OpAmp](#) creator class

{ [OpAmpCreator](#) } represents a creator class for all miscellaneous operational amplifiers (no company specified) within { *libEtech* }.

8.37.2 Member Function Documentation

8.37.2.1 [OpAmp](#)* Ah::MiscOpAmpCreator::create (const char * *model*) const

create instance of specified [OpAmp](#)

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

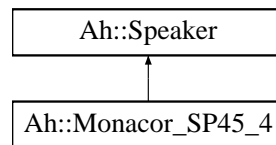
- AH/Etech/[MiscOpCreat.h](#)

8.38 Ah::Monacor_SP45_4 Class Reference

represents the speaker Monacor SP-45/4

```
#include <MonacorSpeakers.h>
```

Inheritance diagram for Ah::Monacor_SP45_4::



Public Member Functions

- [Monacor_SP45_4\(\)](#)
default constructor
- virtual [~Monacor_SP45_4\(\)](#)
destructor

8.38.1 Detailed Description

represents the speaker Monacor SP-45/4

{ Monacor } represents the small bass/midrange speaker SP-45 in 4 Ohm version which is made by Monacor.

8.38.2 Constructor & Destructor Documentation

8.38.2.1 Ah::Monacor_SP45_4::Monacor_SP45_4()

default constructor

8.38.2.2 virtual Ah::Monacor_SP45_4::~~Monacor_SP45_4() [inline, virtual]

destructor

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

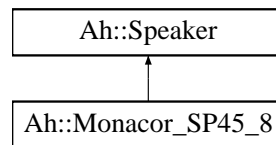
- AH/Etech/[MonacorSpeakers.h](#)

8.39 Ah::Monacor_SP45_8 Class Reference

represents the speaker Monacor SP-45/8

```
#include <MonacorSpeakers.h>
```

Inheritance diagram for Ah::Monacor_SP45_8::



Public Member Functions

- [Monacor_SP45_8 \(\)](#)
default constructor
- virtual [~Monacor_SP45_8 \(\)](#)
destructor

8.39.1 Detailed Description

represents the speaker Monacor SP-45/8

{ Monacor } represents the small bass/midrange speaker SP-45 in 8 Ohm version which is made by Monacor.

8.39.2 Constructor & Destructor Documentation

8.39.2.1 Ah::Monacor_SP45_8::Monacor_SP45_8 ()

default constructor

8.39.2.2 virtual Ah::Monacor_SP45_8::~~Monacor_SP45_8 () [inline, virtual]

destructor

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

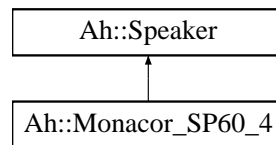
- AH/Etech/[MonacorSpeakers.h](#)

8.40 Ah::Monacor_SP60_4 Class Reference

represents the speaker Monacor SP-60/4

```
#include <MonacorSpeakers.h>
```

Inheritance diagram for Ah::Monacor_SP60_4::



Public Member Functions

- [Monacor_SP60_4\(\)](#)
default constructor
- virtual [~Monacor_SP60_4\(\)](#)
destructor

8.40.1 Detailed Description

represents the speaker Monacor SP-60/4

{ Monacor } represents the small bass/midrange speaker SP-60 in 4 Ohm version which is made by Monacor.

8.40.2 Constructor & Destructor Documentation

8.40.2.1 Ah::Monacor_SP60_4::Monacor_SP60_4()

default constructor

8.40.2.2 virtual Ah::Monacor_SP60_4::~~Monacor_SP60_4() [inline, virtual]

destructor

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

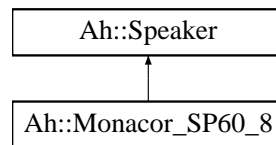
- AH/Etech/[MonacorSpeakers.h](#)

8.41 Ah::Monacor_SP60_8 Class Reference

represents the speaker Monacor SP-60/8

```
#include <MonacorSpeakers.h>
```

Inheritance diagram for Ah::Monacor_SP60_8::



Public Member Functions

- [Monacor_SP60_8 \(\)](#)
default constructor
- virtual [~Monacor_SP60_8 \(\)](#)
destructor

8.41.1 Detailed Description

represents the speaker Monacor SP-60/8

{ Monacor } represents the small bass/midrange speaker SP-60 in 8 Ohm version which is made by Monacor.

8.41.2 Constructor & Destructor Documentation

8.41.2.1 Ah::Monacor_SP60_8::Monacor_SP60_8 ()

default constructor

8.41.2.2 virtual Ah::Monacor_SP60_8::~~Monacor_SP60_8 () [inline, virtual]

destructor

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

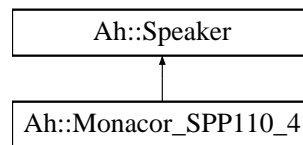
- AH/Etech/[MonacorSpeakers.h](#)

8.42 Ah::Monacor_SPP110_4 Class Reference

represents the speaker Monacor SPP-110/4

```
#include <MonacorSpeakers.h>
```

Inheritance diagram for Ah::Monacor_SPP110_4::



Public Member Functions

- [Monacor_SPP110_4 \(\)](#)
default constructor
- virtual [~Monacor_SPP110_4 \(\)](#)
destructor

8.42.1 Detailed Description

represents the speaker Monacor SPP-110/4

{ Monacor } represents the small bass/midrange speaker SP-110 in 4 Ohm version which is made by Monacor.

8.42.2 Constructor & Destructor Documentation

8.42.2.1 Ah::Monacor_SPP110_4::Monacor_SPP110_4 ()

default constructor

8.42.2.2 virtual Ah::Monacor_SPP110_4::~~Monacor_SPP110_4 () [inline, virtual]

destructor

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

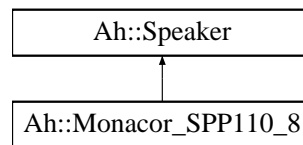
- AH/Etech/[MonacorSpeakers.h](#)

8.43 Ah::Monacor_SPP110_8 Class Reference

represents the speaker Monacor SPP-110/8

```
#include <MonacorSpeakers.h>
```

Inheritance diagram for Ah::Monacor_SPP110_8::



Public Member Functions

- [Monacor_SPP110_8](#) ()
default constructor
- virtual [~Monacor_SPP110_8](#) ()
destructor
- virtual [Dbl](#) [getSpl](#) ([Dbl](#) freq) const
get acoustic power level dB/W/m at specified frequency

8.43.1 Detailed Description

represents the speaker Monacor SPP-110/8

{ Monacor } represents the small bass/midrange speaker SP-110 in 8 Ohm version which is made by Monacor.

8.43.2 Constructor & Destructor Documentation

8.43.2.1 Ah::Monacor_SPP110_8::Monacor_SPP110_8 ()

default constructor

8.43.2.2 virtual Ah::Monacor_SPP110_8::~~Monacor_SPP110_8 () [inline, virtual]

destructor

8.43.3 Member Function Documentation

8.43.3.1 virtual [Dbl](#) Ah::Monacor_SPP110_8::getSpl ([Dbl](#) freq) const [virtual]

get acoustic power level dB/W/m at specified frequency

Reimplemented from [Ah::Speaker](#).

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

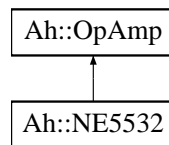
- AH/Etech/[MonacorSpeakers.h](#)

8.44 Ah::NE5532 Class Reference

represents the audio operational amplifier [NE5532](#)

```
#include <MiscOpAmps.h>
```

Inheritance diagram for Ah::NE5532::



Public Member Functions

- [NE5532](#) ()
default constructor

8.44.1 Detailed Description

represents the audio operational amplifier [NE5532](#)

{ [NE5532](#) } represents the audio operational amplifier [NE5532](#) which can be used both in DC (direct current) and in AC (alternate current) applications.

8.44.2 Constructor & Destructor Documentation

8.44.2.1 Ah::NE5532::NE5532 ()

default constructor

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

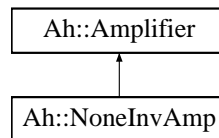
- AH/Etech/[MiscOpAmps.h](#)

8.45 Ah::NoneInvAmp Class Reference

represents a none inverting amplifier

```
#include <NoneInvAmp.h>
```

Inheritance diagram for Ah::NoneInvAmp::



Public Member Functions

- **NoneInvAmp** (const **Impedance** &z1, const **Impedance** &z2)
constructor:
- **NoneInvAmp** (const **OpAmp** &opamp, const **Impedance** &z1, const **Impedance** &z2)
constructor:
- **NoneInvAmp** (const **OpAmp** &opamp, const **Impedance** &z1, const **Impedance** &z2, const **Impedance** &zi)
constructor:
- virtual **~NoneInvAmp** ()
destructor:
- virtual **Cplx** **getA** (**Dbf** freq)
get amplification at frequency { freq}
- virtual **Cplx** **getA** (**Dbf** freq, const **Impedance** &load)
get amplification at frequency { freq} with { load}

Protected Member Functions

- virtual const **Cplx** * **getAd** (**Dbf** freq)
get open loop amplification at frequency { freq}

8.45.1 Detailed Description

represents a none inverting amplifier

{ **NoneInvAmp** } is a none inverting amplifier which uses an operational amplifier. It can be used both in DC (direct current) and in AC (alternate current) applications.

8.45.2 Constructor & Destructor Documentation

8.45.2.1 Ah::NoneInvAmp::NoneInvAmp (const [Impedance](#) & z1, const [Impedance](#) & z2) [\[inline\]](#)

constructor:

8.45.2.2 Ah::NoneInvAmp::NoneInvAmp (const [OpAmp](#) & opamp, const [Impedance](#) & z1, const [Impedance](#) & z2) [\[inline\]](#)

constructor:

8.45.2.3 Ah::NoneInvAmp::NoneInvAmp (const [OpAmp](#) & opamp, const [Impedance](#) & z1, const [Impedance](#) & z2, const [Impedance](#) & zi) [\[inline\]](#)

constructor:

8.45.2.4 virtual Ah::NoneInvAmp::~~NoneInvAmp () [\[inline, virtual\]](#)

destructor:

8.45.3 Member Function Documentation

8.45.3.1 virtual [Cplx](#) Ah::NoneInvAmp::getA ([Dbl](#) freq, const [Impedance](#) & load) [\[virtual\]](#)

get amplification at frequency { freq} with { load}

Implements [Ah::Amplifier](#).

8.45.3.2 virtual [Cplx](#) Ah::NoneInvAmp::getA ([Dbl](#) freq) [\[virtual\]](#)

get amplification at frequency { freq}

Implements [Ah::Amplifier](#).

8.45.3.3 virtual const [Cplx](#)* Ah::NoneInvAmp::getAd ([Dbl](#) freq) [\[protected, virtual\]](#)

get open loop amplification at frequency { freq}

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

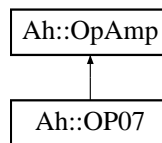
- AH/Etech/[NoneInvAmp.h](#)

8.46 Ah::OP07 Class Reference

represents the industry standard precision operational amplifier [OP07](#)

```
#include <MiscOpAmps.h>
```

Inheritance diagram for Ah::OP07::



Public Member Functions

- [OP07](#) ()
default constructor

8.46.1 Detailed Description

represents the industry standard precision operational amplifier [OP07](#)

{ [OP07](#) } represents the industry standard precision operational amplifier [OP07](#) which can be used both in DC (direct current) and in AC (alternate current) applications.

8.46.2 Constructor & Destructor Documentation

8.46.2.1 Ah::OP07::OP07 ()

default constructor

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

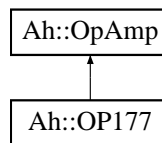
- AH/Etech/[MiscOpAmps.h](#)

8.47 Ah::OP177 Class Reference

represents the industry standard high precision operational amplifier [OP177](#)

```
#include <MiscOpAmps.h>
```

Inheritance diagram for Ah::OP177::



Public Member Functions

- [OP177](#) ()
default constructor

8.47.1 Detailed Description

represents the industry standard high precision operational amplifier [OP177](#)

{ [OP177](#) } represents the industry standard high precision operational amplifier [OP177](#) which can be used both in DC (direct current) and in AC (alternate current) applications.

8.47.2 Constructor & Destructor Documentation

8.47.2.1 Ah::OP177::OP177 ()

default constructor

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

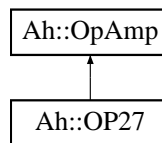
- AH/Etech/[MiscOpAmps.h](#)

8.48 Ah::OP27 Class Reference

represents the fast industry standard precision operational amplifier [OP27](#)

```
#include <MiscOpAmps.h>
```

Inheritance diagram for Ah::OP27::



Public Member Functions

- [OP27](#) ()
default constructor

8.48.1 Detailed Description

represents the fast industry standard precision operational amplifier [OP27](#)

{ [OP27](#) } represents the fast industry standard precision operational amplifier [OP27](#) which can be used both in DC (direct current) and in AC (alternate current) applications.

8.48.2 Constructor & Destructor Documentation

8.48.2.1 Ah::OP27::OP27 ()

default constructor

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

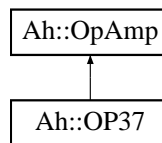
- AH/Etech/[MiscOpAmps.h](#)

8.49 Ah::OP37 Class Reference

represents the very fast industry standard precision operational amplifier [OP37](#)

```
#include <MiscOpAmps.h>
```

Inheritance diagram for Ah::OP37::



Public Member Functions

- [OP37](#) ()
default constructor

8.49.1 Detailed Description

represents the very fast industry standard precision operational amplifier [OP37](#)

{ [OP37](#) } represents the very fast industry standard precision operational amplifier [OP37](#) which can be used both in DC (direct current) and in AC (alternate current) applications.

8.49.2 Constructor & Destructor Documentation

8.49.2.1 Ah::OP37::OP37 ()

default constructor

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

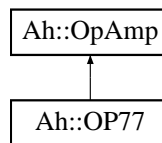
- AH/Etech/[MiscOpAmps.h](#)

8.50 Ah::OP77 Class Reference

represents the industry standard high precision operational amplifier [OP77](#)

```
#include <MiscOpAmps.h>
```

Inheritance diagram for Ah::OP77::



Public Member Functions

- [OP77](#) ()
default constructor

8.50.1 Detailed Description

represents the industry standard high precision operational amplifier [OP77](#)

{ [OP77](#) } represents the industry standard high precision operational amplifier [OP77](#) which can be used both in DC (direct current) and in AC (alternate current) applications.

8.50.2 Constructor & Destructor Documentation

8.50.2.1 Ah::OP77::OP77 ()

default constructor

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

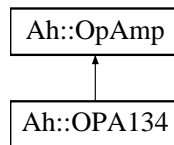
- AH/Etech/[MiscOpAmps.h](#)

8.51 Ah::OPA134 Class Reference

represents the Sound-Plus operational amplifier [OPA134](#)

```
#include <BurrBrownOpAmps.h>
```

Inheritance diagram for Ah::OPA134::



Public Member Functions

- [OPA134](#) ()
default constructor

8.51.1 Detailed Description

represents the Sound-Plus operational amplifier [OPA134](#)

{ [OPA134](#) } represents the Sound-Plus operational amplifier [OPA134](#) made by Burr-Brown, which can be used both in DC (direct current) and in AC (alternate current) applications.

8.51.2 Constructor & Destructor Documentation

8.51.2.1 Ah::OPA134::OPA134 ()

default constructor

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

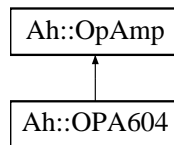
- AH/Etech/[BurrBrownOpAmps.h](#)

8.52 Ah::OPA604 Class Reference

represents the audio operational amplifier [OPA604](#)

```
#include <BurrBrownOpAmps.h>
```

Inheritance diagram for Ah::OPA604::



Public Member Functions

- [OPA604](#) ()
default constructor

8.52.1 Detailed Description

represents the audio operational amplifier [OPA604](#)

{ [OPA604](#) } represents the audio operational amplifier [OPA604](#) made by Burr-Brown, which can be used both in DC (direct current) and in AC (alternate current) applications.

8.52.2 Constructor & Destructor Documentation

8.52.2.1 Ah::OPA604::OPA604 ()

default constructor

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

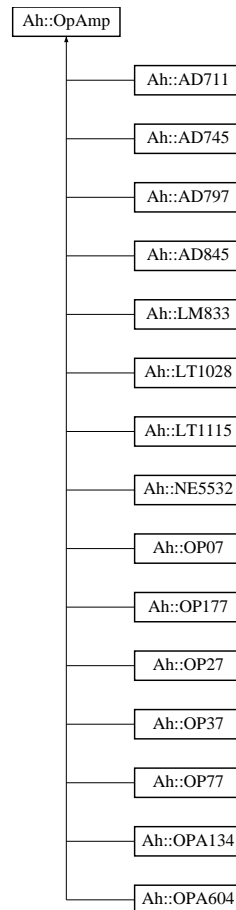
- AH/Etech/[BurrBrownOpAmps.h](#)

8.53 Ah::OpAmp Class Reference

represents an operational amplifier

```
#include <OpAmp.h>
```

Inheritance diagram for Ah::OpAmp::



Public Member Functions

- [OpAmp](#) ()
default constructor
- [OpAmp](#) (const char *name, [Dbl](#) a0, [Dbl](#) ft)
simple constructor
- [OpAmp](#) (const char *name, [Dbl](#) a0, [Dbl](#) ft, [Dbl](#) rout, [Dbl](#) en=0, [Dbl](#) fgen=0.001, [Dbl](#) in=0, [Dbl](#) fgin=0.001)
complete constructor
- [OpAmp](#) (const [OpAmp](#) &op)
copy constructor

- virtual `~OpAmp ()`
destructor:
- const char * `getName ()` const
return name of operational amplifier
- virtual const `Cplx * getAd (Dbl freq)`
get differential open loop amplification at frequency { freq}
- virtual const `Impedance * getZin (Dbl freq)`
get open loop input impedance at frequency { freq}
- virtual const `Impedance * getZout (Dbl freq)`
get open loop output impedance at frequency { freq}
- const `OpAmp & operator= (const OpAmp &op)`
assign operator

Protected Attributes

- `Dbl f`
- `Cplx Af`
- const char * `Name`
- `Dbl A0`
- `Dbl Fg`
- `Dbl En`
- `Dbl FgEn`
- `Dbl In`
- `Dbl FgIn`
- `Resistor * Rout`
- `Resistor * Rin`
- `Capacitor * Cin`

8.53.1 Detailed Description

represents an operational amplifier

{ `OpAmp` } represents a real operational amplifier which can be used both in DC (direct current) and in AC (alternate current) applications.

8.53.2 Constructor & Destructor Documentation

8.53.2.1 `Ah::OpAmp::OpAmp ()`

default constructor

8.53.2.2 Ah::OpAmp::OpAmp (const char * *name*, *Db*l *a0*, *Db*l *ft*)

simple constructor

8.53.2.3 Ah::OpAmp::OpAmp (const char * *name*, *Db*l *a0*, *Db*l *ft*, *Db*l *rout*, *Db*l *en* = 0, *Db*l *fgen* = 0.001, *Db*l *in* = 0, *Db*l *fgin* = 0.001)

complete constructor

8.53.2.4 Ah::OpAmp::OpAmp (const *OpAmp* & *op*)

copy constructor

8.53.2.5 virtual Ah::OpAmp::~~OpAmp () [virtual]

destructor:

8.53.3 Member Function Documentation**8.53.3.1 virtual const *Cplx** Ah::OpAmp::getAd (*Db*l *freq*) [virtual]**

get differential open loop amplification at frequency { *freq*}

8.53.3.2 const char* Ah::OpAmp::getName () const [inline]

return name of operational amplifier

8.53.3.3 virtual const *Impedance Ah::OpAmp::getZin (*Db*l *freq*) [virtual]**

get open loop input impedance at frequency { *freq*}

8.53.3.4 virtual const *Impedance Ah::OpAmp::getZout (*Db*l *freq*) [virtual]**

get open loop output impedance at frequency { *freq*}

8.53.3.5 const *OpAmp*& Ah::OpAmp::operator= (const *OpAmp* & *op*)

assign operator

8.53.4 Member Data Documentation

- 8.53.4.1 [Dbl Ah::OpAmp::A0](#) [protected]
- 8.53.4.2 [Cplx Ah::OpAmp::Af](#) [protected]
- 8.53.4.3 [Capacitor* Ah::OpAmp::Cin](#) [protected]
- 8.53.4.4 [Dbl Ah::OpAmp::En](#) [protected]
- 8.53.4.5 [Dbl Ah::OpAmp::f](#) [protected]
- 8.53.4.6 [Dbl Ah::OpAmp::Fg](#) [protected]
- 8.53.4.7 [Dbl Ah::OpAmp::FgEn](#) [protected]
- 8.53.4.8 [Dbl Ah::OpAmp::FgIn](#) [protected]
- 8.53.4.9 [Dbl Ah::OpAmp::In](#) [protected]
- 8.53.4.10 [const char* Ah::OpAmp::Name](#) [protected]
- 8.53.4.11 [Resistor* Ah::OpAmp::Rin](#) [protected]
- 8.53.4.12 [Resistor* Ah::OpAmp::Rout](#) [protected]

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

- [AH/Etech/OpAmp.h](#)

8.54 Ah::OpAmpCreator Class Reference

represents the [OpAmp](#) creator class

```
#include <OpAmpCreator.h>
```

Public Member Functions

- [OpAmpCreator](#) ()
default constructor
- virtual [~OpAmpCreator](#) ()
virtual destructor
- virtual [OpAmp](#) * [create](#) (const char *model=0, const char *company=0) const
create instance of specified opAmp

8.54.1 Detailed Description

represents the [OpAmp](#) creator class

{ [OpAmpCreator](#) } represents a creator class for all operational amplifiers within {*libEtech*}.

8.54.2 Constructor & Destructor Documentation

8.54.2.1 Ah::OpAmpCreator::OpAmpCreator () [inline]

default constructor

8.54.2.2 virtual Ah::OpAmpCreator::~~OpAmpCreator () [inline, virtual]

virtual destructor

8.54.3 Member Function Documentation

8.54.3.1 virtual [OpAmp](#)* Ah::OpAmpCreator::create (const char * *model* = 0, const char * *company* = 0) const [virtual]

create instance of specified opAmp

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

- AH/Etech/[OpAmpCreator.h](#)

8.55 Ah::PhysConst Class Reference

represents some physical constants

```
#include <PhysConst.h>
```

Static Public Attributes

- static const [Dbl Boltzmann](#) = 1.381e-23
Boltzmann constant.
- static const [Dbl eps0](#) = 1.6e-19
eps0
- static const [Dbl t25](#) = 298.13
temperature of 25 degree Celsius in Kelvin

8.55.1 Detailed Description

represents some physical constants

{ [PhysConst](#) } is a class containing some physical constants.

8.55.2 Member Data Documentation

8.55.2.1 `const Dbl Ah::PhysConst::Boltzmann = 1.381e-23` [static]

Boltzmann constant.

8.55.2.2 `const Dbl Ah::PhysConst::eps0 = 1.6e-19` [static]

eps0

8.55.2.3 `const Dbl Ah::PhysConst::t25 = 298.13` [static]

temperature of 25 degree Celsius in Kelvin

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

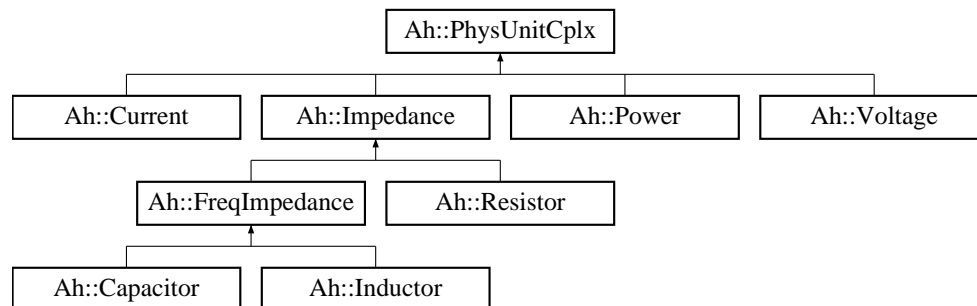
- AH/Etech/[PhysConst.h](#)

8.56 Ah::PhysUnitCplx Class Reference

represents a complex physical unit

```
#include <PhysUnit.h>
```

Inheritance diagram for Ah::PhysUnitCplx::



Public Member Functions

- **PhysUnitCplx** (const char *u, **Dbl** r=0.0, **Dbl** i=0.0)
constructor
- **PhysUnitCplx** (const char *u, const **Cplx** &val)
constructor
- **PhysUnitCplx** (const **PhysUnitCplx** &puc)
copy constructor
- bool **isInf** () const
return { true } if value is infinity
- bool **isNul** () const
return { true } if value is null
- bool **isValid** () const
return { true } if value is valid (neither null nor infinity)
- **Cplx** **getVal** () const
return value of unit

Protected Attributes

- **Cplx** Value
- const char * **Unit**

Friends

- `std::ostream & operator<< (std::ostream &out, const PhysUnitCplx &puc)`
overloaded operator for output into streams
- `Cplx operator+ (const PhysUnitCplx &puc1, const PhysUnitCplx &puc2)`
overloaded operator for addition of two { PhysUnitCplx } objects
- `Cplx operator+ (const PhysUnitCplx &puc, const Cplx &dc)`
overloaded operator for addition of { PhysUnitCplx } + { Cplx }
- `Cplx operator+ (const Cplx &dc, const PhysUnitCplx &puc)`
overloaded operator for addition of { Cplx } + { PhysUnitCplx }
- `Cplx operator- (const PhysUnitCplx &puc1, const PhysUnitCplx &puc2)`
overloaded operator for subtraction of two { PhysUnitCplx } objects
- `Cplx operator- (const PhysUnitCplx &puc, const Cplx &dc)`
overloaded operator for subtraction of { PhysUnitCplx } - { Cplx }
- `Cplx operator- (const Cplx &dc, const PhysUnitCplx &puc)`
overloaded operator for subtraction of { Cplx } - { PhysUnitCplx }
- `Cplx operator * (const PhysUnitCplx &puc1, const PhysUnitCplx &puc2)`
overloaded operator for multiplication of two { PhysUnitCplx } objects
- `Cplx operator * (const PhysUnitCplx &puc, const Cplx &dc)`
*overloaded operator for multiplication of { PhysUnitCplx } * { Cplx }*
- `Cplx operator * (const Cplx &dc, const PhysUnitCplx &puc)`
*overloaded operator for multiplication of { Cplx } * { PhysUnitCplx }*
- `Cplx operator/ (const PhysUnitCplx &puc1, const PhysUnitCplx &puc2)`
overloaded operator for division of two { PhysUnitCplx } objects
- `Cplx operator/ (const PhysUnitCplx &puc, const Cplx &dc)`
overloaded operator for division of { PhysUnitCplx } / { Cplx }
- `Cplx operator/ (const Cplx &dc, const PhysUnitCplx &puc)`
overloaded operator for division of { Cplx } / { PhysUnitCplx }
- `DbI abs (const PhysUnitCplx &puc)`
return absolute of value of a { PhysUnitCplx } object
- `DbI arg (const PhysUnitCplx &puc)`
return argument of value of a { PhysUnitCplx } object

8.56.1 Detailed Description

represents a complex physical unit

{ [PhysUnitCplx](#) } is the base class for all complex physical units, e.g voltage, current, impedances and so forth.

8.56.2 Constructor & Destructor Documentation

8.56.2.1 `Ah::PhysUnitCplx::PhysUnitCplx (const char * u, Dbl r = 0.0, Dbl i = 0.0)` [inline]

constructor

8.56.2.2 `Ah::PhysUnitCplx::PhysUnitCplx (const char * u, const Cplx & val)` [inline]

constructor

8.56.2.3 `Ah::PhysUnitCplx::PhysUnitCplx (const PhysUnitCplx & puc)` [inline]

copy constructor

8.56.3 Member Function Documentation

8.56.3.1 `Cplx Ah::PhysUnitCplx::getVal () const` [inline]

return value of unit

8.56.3.2 `bool Ah::PhysUnitCplx::isInf () const` [inline]

return { true } if value is infinity

8.56.3.3 `bool Ah::PhysUnitCplx::isNul () const` [inline]

return { true } if value is null

8.56.3.4 `bool Ah::PhysUnitCplx::isValid () const` [inline]

return { true } if value is valid (neither null nor infinity)

8.56.4 Friends And Related Function Documentation

8.56.4.1 `Dbl abs (const PhysUnitCplx & puc)` [friend]

return absolute of value of a { [PhysUnitCplx](#) } object

8.56.4.2 Dbl arg (const PhysUnitCplx & puc) [friend]

return argument of value of a { PhysUnitCplx } object

8.56.4.3 Cplx operator * (const Cplx & dc, const PhysUnitCplx & puc) [friend]

overloaded operator for multiplication of { Cplx } * { PhysUnitCplx }

8.56.4.4 Cplx operator * (const PhysUnitCplx & puc, const Cplx & dc) [friend]

overloaded operator for multiplication of { PhysUnitCplx } * { Cplx }

8.56.4.5 Cplx operator * (const PhysUnitCplx & puc1, const PhysUnitCplx & puc2) [friend]

overloaded operator for multiplication of two { PhysUnitCplx } objects

8.56.4.6 Cplx operator+ (const Cplx & dc, const PhysUnitCplx & puc) [friend]

overloaded operator for addition of { Cplx } + { PhysUnitCplx }

8.56.4.7 Cplx operator+ (const PhysUnitCplx & puc, const Cplx & dc) [friend]

overloaded operator for addition of { PhysUnitCplx } + { Cplx }

8.56.4.8 Cplx operator+ (const PhysUnitCplx & puc1, const PhysUnitCplx & puc2) [friend]

overloaded operator for addition of two { PhysUnitCplx } objects

8.56.4.9 Cplx operator- (const Cplx & dc, const PhysUnitCplx & puc) [friend]

overloaded operator for subtraction of { Cplx } - { PhysUnitCplx }

8.56.4.10 Cplx operator- (const PhysUnitCplx & puc, const Cplx & dc) [friend]

overloaded operator for subtraction of { PhysUnitCplx } - { Cplx }

8.56.4.11 Cplx operator- (const PhysUnitCplx & puc1, const PhysUnitCplx & puc2) [friend]

overloaded operator for subtraction of two { PhysUnitCplx } objects

8.56.4.12 Cplx operator/ (const Cplx & dc, const PhysUnitCplx & puc) [friend]

overloaded operator for division of { Cplx } / { PhysUnitCplx }

8.56.4.13 Cplx operator/ (const PhysUnitCplx & puc, const Cplx & dc) [friend]

overloaded operator for division of { PhysUnitCplx } / { Cplx }

8.56.4.14 Cplx operator/ (const PhysUnitCplx & puc1, const PhysUnitCplx & puc2) [friend]

overloaded operator for division of two { PhysUnitCplx } objects

8.56.4.15 std::ostream& operator<< (std::ostream & out, const PhysUnitCplx & puc) [friend]

overloaded operator for output into streams

8.56.5 Member Data Documentation**8.56.5.1 const char* Ah::PhysUnitCplx::Unit [protected]****8.56.5.2 Cplx Ah::PhysUnitCplx::Value [protected]**

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

- AH/Etech/PhysUnit.h

8.57 Ah::PhysUnitDbl Class Reference

represents a scalar physical unit

```
#include <PhysUnit.h>
```

Public Member Functions

- [PhysUnitDbl](#) (const char *u, [Dbl](#) v=0.0)
constructor
- [PhysUnitDbl](#) (const char *u, const [Dbl](#) &val)
constructor
- [PhysUnitDbl](#) (const [PhysUnitDbl](#) &pud)
copy constructor
- virtual [~PhysUnitDbl](#) ()
destructor
- virtual [Dbl](#) [getVal](#) () const
return value of unit

Protected Attributes

- [Dbl](#) [Value](#)
- const char * [Unit](#)

Friends

- std::ostream & [operator<<](#) (std::ostream &out, const [PhysUnitDbl](#) &pud)
overloaded operator for output into streams
- [Dbl](#) [abs](#) (const [PhysUnitDbl](#) &pud)
return absolute of value of a { [PhysUnitDbl](#) } object
- [Dbl](#) [arg](#) (const [PhysUnitDbl](#) &)
return argument of value of a { [PhysUnitDbl](#) } object (here always 0)

8.57.1 Detailed Description

represents a scalar physical unit

{ [PhysUnitDbl](#) } is the base class for all scalar physical units, e.g temperature, distances and so forth.

8.57.2 Constructor & Destructor Documentation

8.57.2.1 Ah::PhysUnitDbl::PhysUnitDbl (const char * *u*, **Dbl** *v* = 0.0) [inline]

constructor

8.57.2.2 Ah::PhysUnitDbl::PhysUnitDbl (const char * *u*, const **Dbl** & *val*) [inline]

constructor

8.57.2.3 Ah::PhysUnitDbl::PhysUnitDbl (const **PhysUnitDbl** & *pud*) [inline]

copy constructor

8.57.2.4 virtual Ah::PhysUnitDbl::~PhysUnitDbl () [inline, virtual]

destructor

8.57.3 Member Function Documentation

8.57.3.1 virtual **Dbl** Ah::PhysUnitDbl::getVal () const [inline, virtual]

return value of unit

8.57.4 Friends And Related Function Documentation

8.57.4.1 **Dbl** abs (const **PhysUnitDbl** & *pud*) [friend]

return absolute of value of a { **PhysUnitDbl** } object

8.57.4.2 **Dbl** arg (const **PhysUnitDbl** &) [friend]

return argument of value of a { **PhysUnitDbl** } object (here always 0)

8.57.4.3 std::ostream& operator<< (std::ostream & *out*, const **PhysUnitDbl** & *pud*) [friend]

overloaded operator for output into streams

8.57.5 Member Data Documentation

8.57.5.1 const char* Ah::PhysUnitDbl::Unit [protected]

8.57.5.2 **Dbl** Ah::PhysUnitDbl::Value [protected]

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

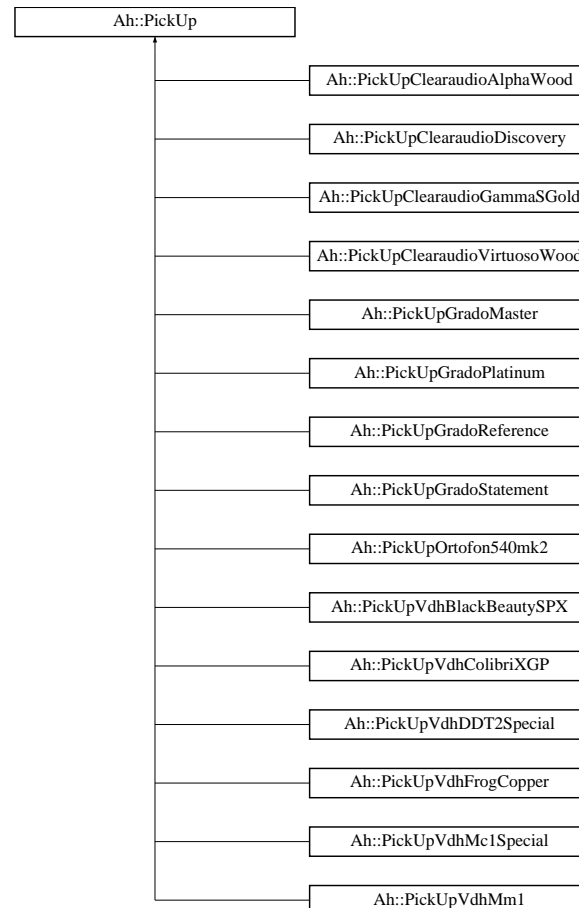
- AH/Etech/[PhysUnit.h](#)

8.58 Ah::PickUp Class Reference

represents a pickup of a record player

```
#include <PickUp.h>
```

Inheritance diagram for Ah::PickUp::



Public Member Functions

- [PickUp](#) ()
default constructor
- [PickUp](#) (const char *name, [Dbl](#) r, [Dbl](#) l, [Dbl](#) c)
constructor
- [PickUp](#) (const [PickUp](#) &op)
copy constructor
- virtual [~PickUp](#) ()
destructor:

- const char * [getName](#) () const
return name of pickup
- virtual [Cplx](#) [getAtt](#) ([Dbl](#) f, const [Impedance](#) &z)
get attenuation at frequency { f} for load impedanze { z}
- const [PickUp](#) & [operator=](#) (const [PickUp](#) &op)
assign operator

8.58.1 Detailed Description

represents a pickup of a record player

{ [PickUp](#) } represents a pickup of a record player.

8.58.2 Constructor & Destructor Documentation

8.58.2.1 Ah::PickUp::PickUp ()

default constructor

8.58.2.2 Ah::PickUp::PickUp (const char * name, [Dbl](#) r, [Dbl](#) l, [Dbl](#) c)

constructor

8.58.2.3 Ah::PickUp::PickUp (const [PickUp](#) & op)

copy constructor

8.58.2.4 virtual Ah::PickUp::~~PickUp () [virtual]

destructor:

8.58.3 Member Function Documentation

8.58.3.1 virtual [Cplx](#) Ah::PickUp::getAtt ([Dbl](#) f, const [Impedance](#) &z) [virtual]

get attenuation at frequency { f} for load impedanze { z}

8.58.3.2 const char* Ah::PickUp::getName () const [inline]

return name of pickup

8.58.3.3 `const PickUp& Ah::PickUp::operator= (const PickUp & op)`

assign operator

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

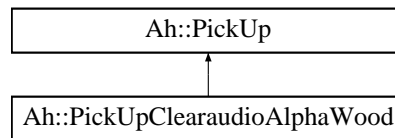
- AH/Etech/[PickUp.h](#)

8.59 Ah::PickUpClearaudioAlphaWood Class Reference

Clearaudio Alpha Wood (data supplied by Frank Klemm).

```
#include <PickUp.h>
```

Inheritance diagram for Ah::PickUpClearaudioAlphaWood::



Public Member Functions

- [PickUpClearaudioAlphaWood \(\)](#)
default constructor
- virtual [~PickUpClearaudioAlphaWood \(\)](#)
destructor:

8.59.1 Detailed Description

Clearaudio Alpha Wood (data supplied by Frank Klemm).

8.59.2 Constructor & Destructor Documentation

8.59.2.1 Ah::PickUpClearaudioAlphaWood::PickUpClearaudioAlphaWood () [inline]

default constructor

8.59.2.2 virtual Ah::PickUpClearaudioAlphaWood::~~PickUpClearaudioAlphaWood () [inline, virtual]

destructor:

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

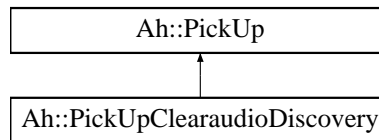
- AH/Etech/[PickUp.h](#)

8.60 Ah::PickUpClearaudioDiscovery Class Reference

Clearaudio Discovery (data supplied by Frank Klemm).

```
#include <PickUp.h>
```

Inheritance diagram for Ah::PickUpClearaudioDiscovery::



Public Member Functions

- [PickUpClearaudioDiscovery \(\)](#)
default constructor
- virtual [~PickUpClearaudioDiscovery \(\)](#)
destructor:

8.60.1 Detailed Description

Clearaudio Discovery (data supplied by Frank Klemm).

8.60.2 Constructor & Destructor Documentation

8.60.2.1 Ah::PickUpClearaudioDiscovery::PickUpClearaudioDiscovery () [inline]

default constructor

8.60.2.2 virtual Ah::PickUpClearaudioDiscovery::~~PickUpClearaudioDiscovery () [inline, virtual]

destructor:

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

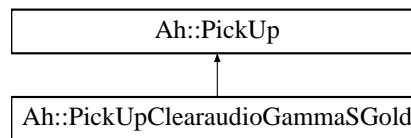
- AH/Etech/[PickUp.h](#)

8.61 Ah::PickUpClearaudioGammaSGold Class Reference

Clearaudio Gamma S Gold (data supplied by Frank Klemm).

```
#include <PickUp.h>
```

Inheritance diagram for Ah::PickUpClearaudioGammaSGold::



Public Member Functions

- [PickUpClearaudioGammaSGold \(\)](#)
default constructor
- virtual [~PickUpClearaudioGammaSGold \(\)](#)
destructor:

8.61.1 Detailed Description

Clearaudio Gamma S Gold (data supplied by Frank Klemm).

8.61.2 Constructor & Destructor Documentation

8.61.2.1 Ah::PickUpClearaudioGammaSGold::PickUpClearaudioGammaSGold () [inline]

default constructor

8.61.2.2 virtual Ah::PickUpClearaudioGammaSGold::~~PickUpClearaudioGammaSGold () [inline, virtual]

destructor:

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

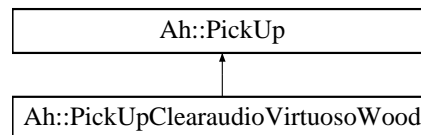
- AH/Etech/[PickUp.h](#)

8.62 Ah::PickUpClearaudioVirtuosoWood Class Reference

Clearaudio Virtuoso Wood (data supplied by Frank Klemm).

```
#include <PickUp.h>
```

Inheritance diagram for Ah::PickUpClearaudioVirtuosoWood::



Public Member Functions

- [PickUpClearaudioVirtuosoWood \(\)](#)
default constructor
- virtual [~PickUpClearaudioVirtuosoWood \(\)](#)
destructor:

8.62.1 Detailed Description

Clearaudio Virtuoso Wood (data supplied by Frank Klemm).

8.62.2 Constructor & Destructor Documentation

8.62.2.1 Ah::PickUpClearaudioVirtuosoWood::PickUpClearaudioVirtuosoWood () [inline]

default constructor

8.62.2.2 virtual Ah::PickUpClearaudioVirtuosoWood::~~PickUpClearaudioVirtuosoWood () [inline, virtual]

destructor:

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

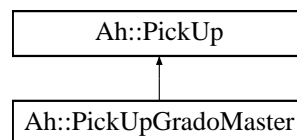
- AH/Etech/[PickUp.h](#)

8.63 Ah::PickUpGradoMaster Class Reference

Grado Master (data supplied by Frank Klemm).

```
#include <PickUp.h>
```

Inheritance diagram for Ah::PickUpGradoMaster::



Public Member Functions

- [PickUpGradoMaster \(\)](#)
default constructor
- virtual [~PickUpGradoMaster \(\)](#)
destructor:

8.63.1 Detailed Description

Grado Master (data supplied by Frank Klemm).

8.63.2 Constructor & Destructor Documentation

8.63.2.1 Ah::PickUpGradoMaster::PickUpGradoMaster () [inline]

default constructor

8.63.2.2 virtual Ah::PickUpGradoMaster::~~PickUpGradoMaster () [inline, virtual]

destructor:

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

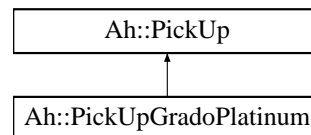
- AH/Etech/[PickUp.h](#)

8.64 Ah::PickUpGradoPlatinum Class Reference

Grado Platinum (data supplied by Frank Klemm).

```
#include <PickUp.h>
```

Inheritance diagram for Ah::PickUpGradoPlatinum::



Public Member Functions

- [PickUpGradoPlatinum \(\)](#)
default constructor
- virtual [~PickUpGradoPlatinum \(\)](#)
destructor:

8.64.1 Detailed Description

Grado Platinum (data supplied by Frank Klemm).

8.64.2 Constructor & Destructor Documentation

8.64.2.1 Ah::PickUpGradoPlatinum::PickUpGradoPlatinum () [inline]

default constructor

8.64.2.2 virtual Ah::PickUpGradoPlatinum::~~PickUpGradoPlatinum () [inline, virtual]

destructor:

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

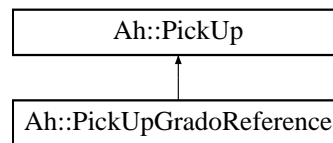
- AH/Etech/[PickUp.h](#)

8.65 Ah::PickUpGradoReference Class Reference

Grado Reference (data supplied by Frank Klemm).

```
#include <PickUp.h>
```

Inheritance diagram for Ah::PickUpGradoReference::



Public Member Functions

- [PickUpGradoReference](#) ()
default constructor
- virtual [~PickUpGradoReference](#) ()
destructor:

8.65.1 Detailed Description

Grado Reference (data supplied by Frank Klemm).

8.65.2 Constructor & Destructor Documentation

8.65.2.1 Ah::PickUpGradoReference::PickUpGradoReference () [inline]

default constructor

8.65.2.2 virtual Ah::PickUpGradoReference::~~PickUpGradoReference () [inline, virtual]

destructor:

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

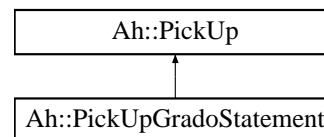
- AH/Etech/[PickUp.h](#)

8.66 Ah::PickUpGradoStatement Class Reference

Grado The Statement (data supplied by Frank Klemm).

```
#include <PickUp.h>
```

Inheritance diagram for Ah::PickUpGradoStatement::



Public Member Functions

- [PickUpGradoStatement \(\)](#)
default constructor
- virtual [~PickUpGradoStatement \(\)](#)
destructor:

8.66.1 Detailed Description

Grado The Statement (data supplied by Frank Klemm).

8.66.2 Constructor & Destructor Documentation

8.66.2.1 Ah::PickUpGradoStatement::PickUpGradoStatement () [inline]

default constructor

8.66.2.2 virtual Ah::PickUpGradoStatement::~~PickUpGradoStatement () [inline, virtual]

destructor:

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

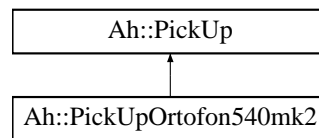
- AH/Etech/[PickUp.h](#)

8.67 Ah::PickUpOrtofon540mk2 Class Reference

Ortofon 540 MK2 (example supplied by Frank Klemm in drmh).

```
#include <PickUp.h>
```

Inheritance diagram for Ah::PickUpOrtofon540mk2::



Public Member Functions

- [PickUpOrtofon540mk2 \(\)](#)
default constructor
- virtual [~PickUpOrtofon540mk2 \(\)](#)
destructor:

8.67.1 Detailed Description

Ortofon 540 MK2 (example supplied by Frank Klemm in drmh).

8.67.2 Constructor & Destructor Documentation

8.67.2.1 Ah::PickUpOrtofon540mk2::PickUpOrtofon540mk2 () [inline]

default constructor

8.67.2.2 virtual Ah::PickUpOrtofon540mk2::~~PickUpOrtofon540mk2 () [inline, virtual]

destructor:

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

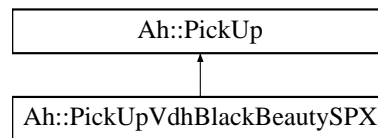
- AH/Etech/[PickUp.h](#)

8.68 Ah::PickUpVdhBlackBeautySPX Class Reference

Van den Hul Black Beauty SPX (data supplied by Frank Klemm).

```
#include <PickUp.h>
```

Inheritance diagram for Ah::PickUpVdhBlackBeautySPX::



Public Member Functions

- [PickUpVdhBlackBeautySPX \(\)](#)
default constructor
- virtual [~PickUpVdhBlackBeautySPX \(\)](#)
destructor:

8.68.1 Detailed Description

Van den Hul Black Beauty SPX (data supplied by Frank Klemm).

8.68.2 Constructor & Destructor Documentation

8.68.2.1 Ah::PickUpVdhBlackBeautySPX::PickUpVdhBlackBeautySPX () [inline]

default constructor

8.68.2.2 virtual Ah::PickUpVdhBlackBeautySPX::~~PickUpVdhBlackBeautySPX () [inline, virtual]

destructor:

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

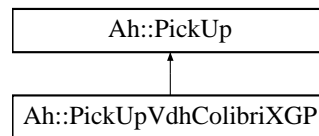
- AH/Etech/[PickUp.h](#)

8.69 Ah::PickUpVdhColibriXGP Class Reference

Van den Hul The Colibri XGP (data supplied by Frank Klemm).

```
#include <PickUp.h>
```

Inheritance diagram for Ah::PickUpVdhColibriXGP::



Public Member Functions

- [PickUpVdhColibriXGP \(\)](#)
default constructor
- virtual [~PickUpVdhColibriXGP \(\)](#)
destructor:

8.69.1 Detailed Description

Van den Hul The Colibri XGP (data supplied by Frank Klemm).

8.69.2 Constructor & Destructor Documentation

8.69.2.1 Ah::PickUpVdhColibriXGP::PickUpVdhColibriXGP () [inline]

default constructor

8.69.2.2 virtual Ah::PickUpVdhColibriXGP::~~PickUpVdhColibriXGP () [inline, virtual]

destructor:

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

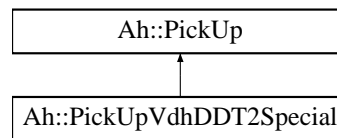
- AH/Etech/[PickUp.h](#)

8.70 Ah::PickUpVdhDDT2Special Class Reference

Van den Hul DDT II Special (data supplied by Frank Klemm).

```
#include <PickUp.h>
```

Inheritance diagram for Ah::PickUpVdhDDT2Special::



Public Member Functions

- [PickUpVdhDDT2Special \(\)](#)
default constructor
- virtual [~PickUpVdhDDT2Special \(\)](#)
destructor:

8.70.1 Detailed Description

Van den Hul DDT II Special (data supplied by Frank Klemm).

8.70.2 Constructor & Destructor Documentation

8.70.2.1 Ah::PickUpVdhDDT2Special::PickUpVdhDDT2Special () [inline]

default constructor

8.70.2.2 virtual Ah::PickUpVdhDDT2Special::~~PickUpVdhDDT2Special () [inline, virtual]

destructor:

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

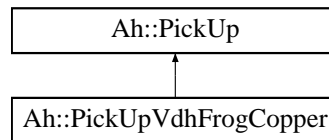
- AH/Etech/[PickUp.h](#)

8.71 Ah::PickUpVdhFrogCopper Class Reference

Van den Hul Frog Copper (data supplied by Frank Klemm).

```
#include <PickUp.h>
```

Inheritance diagram for Ah::PickUpVdhFrogCopper::



Public Member Functions

- [PickUpVdhFrogCopper \(\)](#)
default constructor
- virtual [~PickUpVdhFrogCopper \(\)](#)
destructor:

8.71.1 Detailed Description

Van den Hul Frog Copper (data supplied by Frank Klemm).

8.71.2 Constructor & Destructor Documentation

8.71.2.1 Ah::PickUpVdhFrogCopper::PickUpVdhFrogCopper () [inline]

default constructor

8.71.2.2 virtual Ah::PickUpVdhFrogCopper::~~PickUpVdhFrogCopper () [inline, virtual]

destructor:

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

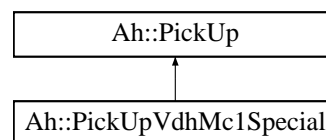
- AH/Etech/[PickUp.h](#)

8.72 Ah::PickUpVdhMc1Special Class Reference

Van den Hul MC-1 Special.

```
#include <VdHPickUps.h>
```

Inheritance diagram for Ah::PickUpVdhMc1Special::



Public Member Functions

- [PickUpVdhMc1Special \(\)](#)
default constructor
- virtual [~PickUpVdhMc1Special \(\)](#)
destructor:

8.72.1 Detailed Description

Van den Hul MC-1 Special.

8.72.2 Constructor & Destructor Documentation

8.72.2.1 Ah::PickUpVdhMc1Special::PickUpVdhMc1Special () [inline]

default constructor

8.72.2.2 virtual Ah::PickUpVdhMc1Special::~~PickUpVdhMc1Special () [inline, virtual]

destructor:

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

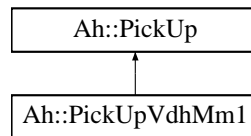
- AH/Etech/[VdHPickUps.h](#)

8.73 Ah::PickUpVdhMm1 Class Reference

Van den Hul MM-1.

```
#include <VdHPickUps.h>
```

Inheritance diagram for Ah::PickUpVdhMm1::



Public Member Functions

- [PickUpVdhMm1 \(\)](#)
default constructor
- virtual [~PickUpVdhMm1 \(\)](#)
destructor:

8.73.1 Detailed Description

Van den Hul MM-1.

8.73.2 Constructor & Destructor Documentation

8.73.2.1 Ah::PickUpVdhMm1::PickUpVdhMm1 () [inline]

default constructor

8.73.2.2 virtual Ah::PickUpVdhMm1::~~PickUpVdhMm1 () [inline, virtual]

destructor:

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

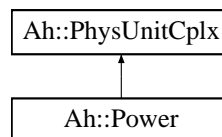
- AH/Etech/[VdHPickUps.h](#)

8.74 Ah::Power Class Reference

represents an electrical power

```
#include <Power.h>
```

Inheritance diagram for Ah::Power::



Public Member Functions

- **Power** (Dbl r=0.0, Dbl i=0.0)
constructor
- **Power** (const Cplx &val)
constructor
- **Power** (const Power &i)
copy constructor

Friends

- **Power operator+** (const Power &p1, const Power &p2)
overloaded operator + for addition of two powers
- **Power operator-** (const Power &p1, const Power &p2)
overloaded operator - for subtraction of two powers
- **Power operator *** (const Power &p1, const Cplx &k)
*overloaded operator * for multiplication of { Power } * { Cplx }*
- **Power operator *** (const Cplx &k, const Power &p1)
*overloaded operator * for multiplication of { Cplx } * { Power }*
- **Cplx operator/** (const Power &p1, const Power &p2)
overloaded operator / for division of { Power } / { Power }
- **Power operator/** (const Power &p1, const Cplx &k)
overloaded operator / for division of { Power } / { Cplx }
- bool **operator==** (const Power &p1, const Power &p2)
overloaded operator == for comparison of two Powers
- bool **operator!=** (const Power &p1, const Power &p2)

overloaded operator != for comparison of two Powers

- `bool operator< (const Power &p1, const Power &p2)`
overloaded operator < for comparison of two Powers
- `bool operator> (const Power &p1, const Power &p2)`
overloaded operator > for comparison of two Powers

8.74.1 Detailed Description

represents an electrical power

{ `Power` } represents an electrical power which can be used both in DC (direct current) and in AC (alternate current) applications.

8.74.2 Constructor & Destructor Documentation

8.74.2.1 `Ah::Power::Power (Dbl r = 0.0, Dbl i = 0.0)` `[inline]`

constructor

8.74.2.2 `Ah::Power::Power (const Cplx & val)` `[inline]`

constructor

8.74.2.3 `Ah::Power::Power (const Power & i)` `[inline]`

copy constructor

8.74.3 Friends And Related Function Documentation

8.74.3.1 `Power operator * (const Cplx & k, const Power & p1)` `[friend]`

overloaded operator * for multiplication of { `Cplx` } * { `Power` }

8.74.3.2 `Power operator * (const Power & p1, const Cplx & k)` `[friend]`

overloaded operator * for multiplication of { `Power` } * { `Cplx` }

8.74.3.3 `bool operator!= (const Power & p1, const Power & p2)` `[friend]`

overloaded operator != for comparison of two Powers

8.74.3.4 `Power operator+ (const Power & p1, const Power & p2)` `[friend]`

overloaded operator + for addition of two powers

8.74.3.5 Power operator- (const **Power** & *p1*, const **Power** & *p2*) [friend]

overloaded operator - for subtraction of two powers

8.74.3.6 Power operator/ (const **Power** & *p1*, const **Cplx** & *k*) [friend]

overloaded operator / for division of { **Power** } / { **Cplx** }

8.74.3.7 Cplx operator/ (const **Power** & *p1*, const **Power** & *p2*) [friend]

overloaded operator / for division of { **Power** } / { **Power** }

8.74.3.8 bool operator< (const **Power** & *p1*, const **Power** & *p2*) [friend]

overloaded operator < for comparison of two Powers

8.74.3.9 bool operator== (const **Power** & *p1*, const **Power** & *p2*) [friend]

overloaded operator == for comparison of two Powers

8.74.3.10 bool operator> (const **Power** & *p1*, const **Power** & *p2*) [friend]

overloaded operator > for comparison of two Powers

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

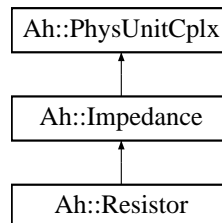
- AH/Etech/[Power.h](#)

8.75 Ah::Resistor Class Reference

represents an ideal electrical resistor

```
#include <Resistor.h>
```

Inheritance diagram for Ah::Resistor::



Public Member Functions

- **Resistor** (**Db**l r=0.0)
constructor
- **Resistor** (const **Resistor** &r)
copy constructor
- **Db**l getRes () const
return resistance

Friends

- **Resistor operator+** (const **Resistor** &r1, const **Resistor** &r2)
overloaded operator + for serial circuit of two resistors
- **Resistor operator||** (const **Resistor** &r1, const **Resistor** &r2)
overloaded operator || for parallel circuit of two resistors
- **Resistor operator *** (const **Resistor** &r, **Db**l k)
*overloaded operator * for multiplication of { **Resistor** } * { **Db**l }*
- **Resistor operator *** (**Db**l k, const **Resistor** &r)
*overloaded operator * for multiplication of { **Db**l } * { **Resistor** }*
- **Resistor operator/** (const **Resistor** &r, **Db**l k)
*overloaded operator / for division of { **Resistor** } / { **Db**l }*
- std::ostream & **operator<<** (std::ostream &out, const **Resistor** &r)
overloaded operator for output into streams

8.75.1 Detailed Description

represents an ideal electrical resistor

{ [Resistor](#) } represents an ideal electrical resistor which can be used both in DC (direct current) and in AC (alternate current) applications.

8.75.2 Constructor & Destructor Documentation

8.75.2.1 `Ah::Resistor::Resistor (Dbl r = 0.0)` `[inline]`

constructor

8.75.2.2 `Ah::Resistor::Resistor (const Resistor & r)` `[inline]`

copy constructor

8.75.3 Member Function Documentation

8.75.3.1 `Dbl Ah::Resistor::getRes () const` `[inline]`

return resistance

8.75.4 Friends And Related Function Documentation

8.75.4.1 `Resistor operator * (Dbl k, const Resistor & r)` `[friend]`

overloaded operator * for multiplication of { Dbl } * { [Resistor](#) }

8.75.4.2 `Resistor operator * (const Resistor & r, Dbl k)` `[friend]`

overloaded operator * for multiplication of { [Resistor](#) } * { Dbl }

8.75.4.3 `Resistor operator+ (const Resistor & r1, const Resistor & r2)` `[friend]`

overloaded operator + for serial circuit of two resistors

8.75.4.4 `Resistor operator/ (const Resistor & r, Dbl k)` `[friend]`

overloaded operator / for division of { [Resistor](#) } / { Dbl }

8.75.4.5 `std::ostream& operator<< (std::ostream & out, const Resistor & r)` `[friend]`

overloaded operator for output into streams

8.75.4.6 Resistor operator|| (const Resistor & r1, const Resistor & r2) [friend]

overloaded operator || for parallel circuit of two resistors

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

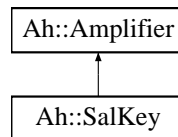
- AH/Etech/[Resistor.h](#)

8.76 Ah::SalKey Class Reference

Sallen-Key Filter.

```
#include <SalKey.h>
```

Inheritance diagram for Ah::SalKey::



Public Member Functions

- [SalKey](#) (const [OpAmp](#) &opamp, const [Impedance](#) &z1, const [Impedance](#) &z2, const [Impedance](#) &z3, const [Impedance](#) &z4, const [Impedance](#) &z5, const [Impedance](#) &z6)

constructors:

- virtual [~SalKey](#) ()

destructor:

- virtual [Cplx](#) [getA](#) ([Dbl](#) freq)

get amplification at frequency { freq}:

- virtual [Cplx](#) [getA](#) ([Dbl](#) freq, const [Impedance](#) &)

get amplification at frequency { freq} with output { load}:

8.76.1 Detailed Description

Sallen-Key Filter.

{ [SalKey](#) } is a Sallen-Key Filter: a 2-pole filter using an operational amplifier.

8.76.2 Constructor & Destructor Documentation

- 8.76.2.1** [Ah::SalKey::SalKey](#) (const [OpAmp](#) &opamp, const [Impedance](#) &z1, const [Impedance](#) &z2, const [Impedance](#) &z3, const [Impedance](#) &z4, const [Impedance](#) &z5, const [Impedance](#) &z6) [\[inline\]](#)

constructors:

- 8.76.2.2** virtual [Ah::SalKey::~~SalKey](#) () [\[inline, virtual\]](#)

destructor:

8.76.3 Member Function Documentation

8.76.3.1 virtual **Cplx** Ah::SalKey::getA (**Dbl** *freq*, const **Impedance** &) [inline, virtual]

get amplification at frequency { freq} with output { load}:

Implements [Ah::Amplifier](#).

8.76.3.2 virtual **Cplx** Ah::SalKey::getA (**Dbl** *freq*) [virtual]

get amplification at frequency { freq}:

Implements [Ah::Amplifier](#).

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

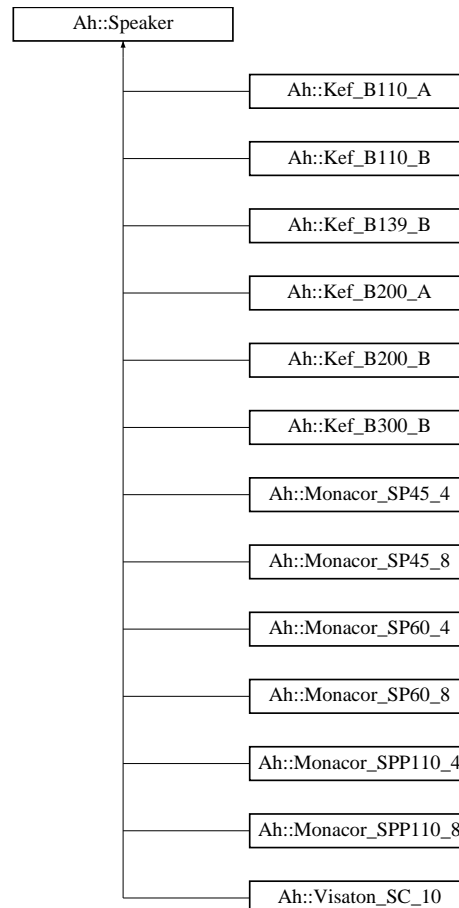
- AH/Etech/[SalKey.h](#)

8.77 Ah::Speaker Class Reference

represents a pure speaker system

```
#include <Speaker.h>
```

Inheritance diagram for Ah::Speaker::



Public Member Functions

- [Speaker](#) ()
default constructor
- [Speaker](#) (const [Speaker](#) &s)
copy constructor
- [Speaker](#) (const char *n)
constructor
- virtual [~Speaker](#) ()
virtual destructor

- `const char * getName () const`
get name of speaker
- `Dbt getQt () const`
get overall damping factor
- `Dbt getQe () const`
get electrical damping factor
- `Dbt getQm () const`
get mechanical damping factor
- `Dbt getFres () const`
get resonance frequency
- `Dbt getVa () const`
get equivalent volume
- `virtual Dbt getSpl (Dbt freq) const`
get acoustic power level dB/W/m at specified frequency
- `virtual Impedance getImp (Dbt freq) const`
get impedance of voice coil at specified frequency
- `const Speaker & operator= (const Speaker &s)`
assignment operator

Protected Attributes

- `Dbt Rdc`
- `Dbt Ls`
- `Dbt Fr`
- `Dbt Qm`
- `Dbt Qe`
- `Dbt Qt`
- `Dbt Va`
- `Dbt Spl`

8.77.1 Detailed Description

represents a pure speaker system

{ `Speaker` } represents a real speaker system.

8.77.2 Constructor & Destructor Documentation

8.77.2.1 Ah::Speaker::Speaker ()

default constructor

8.77.2.2 `Ah::Speaker::Speaker (const Speaker & s)`

copy constructor

8.77.2.3 `Ah::Speaker::Speaker (const char * n)`

constructor

8.77.2.4 `virtual Ah::Speaker::~~Speaker () [inline, virtual]`

virtual destructor

8.77.3 Member Function Documentation**8.77.3.1** `Dbl Ah::Speaker::getFres () const [inline]`

get resonance frequency

8.77.3.2 `virtual Impedance Ah::Speaker::getImp (Dbl freq) const [virtual]`

get impedance of voice coil at specified frequency

8.77.3.3 `const char* Ah::Speaker::getName () const [inline]`

get name of speaker

8.77.3.4 `Dbl Ah::Speaker::getQe () const [inline]`

get electrical damping factor

8.77.3.5 `Dbl Ah::Speaker::getQm () const [inline]`

get mechanical damping factor

8.77.3.6 `Dbl Ah::Speaker::getQt () const [inline]`

get overall damping factor

8.77.3.7 `virtual Dbl Ah::Speaker::getSpl (Dbl freq) const [virtual]`

get acoustic power level dB/W/m at specified frequency

Reimplemented in [Ah::Monacor_SPP110_8](#), and [Ah::Visaton_SC_10](#).

8.77.3.8 `Dbl Ah::Speaker::getVa () const [inline]`

get equivalent volume

8.77.3.9 `const Speaker& Ah::Speaker::operator= (const Speaker & s)`

assignment operator

8.77.4 Member Data Documentation

8.77.4.1 `Dbl Ah::Speaker::Fr` [protected]

8.77.4.2 `Dbl Ah::Speaker::Ls` [protected]

8.77.4.3 `Dbl Ah::Speaker::Qe` [protected]

8.77.4.4 `Dbl Ah::Speaker::Qm` [protected]

8.77.4.5 `Dbl Ah::Speaker::Qt` [protected]

8.77.4.6 `Dbl Ah::Speaker::Rdc` [protected]

8.77.4.7 `Dbl Ah::Speaker::Spl` [protected]

8.77.4.8 `Dbl Ah::Speaker::Va` [protected]

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

- [AH/Etech/Speaker.h](#)

8.78 Ah::SpeakerCreator Class Reference

represents the speaker creator class

```
#include <SpeakerCreator.h>
```

Public Member Functions

- [SpeakerCreator](#) ()
default constructor
- virtual [~SpeakerCreator](#) ()
virtual destructor
- virtual [Speaker](#) * [create](#) (const char *company, const char *model) const
create instance of specified speaker

8.78.1 Detailed Description

represents the speaker creator class

{ [SpeakerCreator](#) } represents a creator class for all speakers within {*libEtech*}.

8.78.2 Constructor & Destructor Documentation

8.78.2.1 Ah::SpeakerCreator::SpeakerCreator () [inline]

default constructor

8.78.2.2 virtual Ah::SpeakerCreator::~~SpeakerCreator () [inline, virtual]

virtual destructor

8.78.3 Member Function Documentation

8.78.3.1 virtual [Speaker](#)* Ah::SpeakerCreator::create (const char * *company*, const char * *model*) const [virtual]

create instance of specified speaker

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

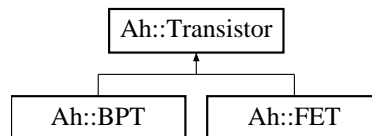
- AH/Etech/[SpeakerCreator.h](#)

8.79 Ah::Transistor Class Reference

represents a generic transistor

```
#include <Transistor.h>
```

Inheritance diagram for Ah::Transistor::



Public Member Functions

- virtual [~Transistor](#) ()
destructor
- virtual [Cplx getS](#) (const [Voltage](#) &u, [Dbl](#) t=PhysConst_T25) const =0
return steilheit
- virtual [Cplx getS](#) (const [Current](#) &i, [Dbl](#) t=PhysConst_T25) const =0
return steilheit
- virtual [Current getIout](#) (const [Voltage](#) &Uin, [Dbl](#) t=PhysConst_T25) const =0
return output current depending on input voltage

8.79.1 Detailed Description

represents a generic transistor

{ [Transistor](#) } is a pure virtual transistor base class, which just defines the common interface to all Transistors.

8.79.2 Constructor & Destructor Documentation

8.79.2.1 virtual Ah::Transistor::~~Transistor () [inline, virtual]

destructor

8.79.3 Member Function Documentation

8.79.3.1 virtual [Current](#) Ah::Transistor::getIout (const [Voltage](#) &Uin, [Dbl](#) t = PhysConst_T25) const [pure virtual]

return output current depending on input voltage

Implemented in [Ah::BPT](#), and [Ah::FET](#).

8.79.3.2 `virtual Cplx Ah::Transistor::getS (const Current & i, Dbl t = PhysConst_T25) const`
[pure virtual]

return steilheit

Implemented in [Ah::BPT](#), and [Ah::FET](#).

8.79.3.3 `virtual Cplx Ah::Transistor::getS (const Voltage & u, Dbl t = PhysConst_T25) const`
[pure virtual]

return steilheit

Implemented in [Ah::BPT](#), and [Ah::FET](#).

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

- [AH/Etech/Transistor.h](#)

8.80 Ah::TransistorFollower Class Reference

represents a generic transistor follower

```
#include <TransFollow.h>
```

Public Member Functions

- [TransistorFollower](#) ([Transistor](#) &t, const [Impedance](#) &load)
constructor
- [~TransistorFollower](#) ()
destructor
- [Current](#) [getIload](#) (const [Voltage](#) &Us)
return current through load
- [Voltage](#) [getUload](#) (const [Voltage](#) &Us)
return voltage at load

8.80.1 Detailed Description

represents a generic transistor follower

{ [TransistorFollower](#) } is an impedance buffer using a transistor in follower mode.

8.80.2 Constructor & Destructor Documentation

8.80.2.1 [Ah::TransistorFollower::TransistorFollower](#) ([Transistor](#) & t, const [Impedance](#) & load)
[inline]

constructor

8.80.2.2 [Ah::TransistorFollower::~~TransistorFollower](#) () [inline]

destructor

8.80.3 Member Function Documentation

8.80.3.1 [Current](#) [Ah::TransistorFollower::getIload](#) (const [Voltage](#) & Us)

return current through load

8.80.3.2 [Voltage](#) [Ah::TransistorFollower::getUload](#) (const [Voltage](#) & Us)

return voltage at load

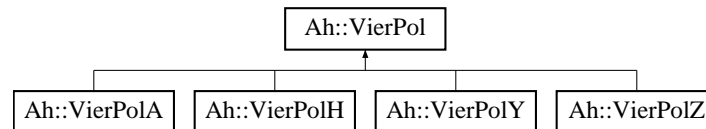
The documentation for this class was generated from the following file:

- [AH/Etech/TransFollow.h](#)

8.81 Ah::VierPol Class Reference

```
#include <VierPol.h>
```

Inheritance diagram for Ah::VierPol::



Public Member Functions

- [Cplx det](#) () const
calculate and return determinante
- bool [isValid](#) () const
return true if { [VierPol](#) } is valid

Protected Member Functions

- [VierPol](#) ()
default constructor
- [VierPol](#) ([Cplx](#) C11, [Cplx](#) C12, [Cplx](#) C21, [Cplx](#) C22)
constructor
- [~VierPol](#) ()
destructor
- bool [equal](#) (const [VierPol](#) &vp) const
return true if equal

Protected Attributes

- [Cplx](#) c11
- [Cplx](#) c12
- [Cplx](#) c21
- [Cplx](#) c22

Friends

- std::ostream & [operator<<](#) (std::ostream &out, const [VierPol](#) &vp)
overloaded operator for output into streams

8.81.1 Constructor & Destructor Documentation

8.81.1.1 `Ah::VierPol::VierPol ()` [inline, protected]

default constructor

8.81.1.2 `Ah::VierPol::VierPol (Cplx C11, Cplx C12, Cplx C21, Cplx C22)` [inline, protected]

constructor

8.81.1.3 `Ah::VierPol::~~VierPol ()` [inline, protected]

destructor

8.81.2 Member Function Documentation

8.81.2.1 `Cplx Ah::VierPol::det () const` [inline]

calculate and return determinante

8.81.2.2 `bool Ah::VierPol::equal (const VierPol & vp) const` [inline, protected]

return true if equal

8.81.2.3 `bool Ah::VierPol::isValid () const` [inline]

return true if { `VierPol` } is valid

8.81.3 Friends And Related Function Documentation

8.81.3.1 `std::ostream& operator<< (std::ostream & out, const VierPol & vp)` [friend]

overloaded operator for output into streams

8.81.4 Member Data Documentation

8.81.4.1 `Cplx Ah::VierPol::c11` [protected]

8.81.4.2 `Cplx Ah::VierPol::c12` [protected]

8.81.4.3 `Cplx Ah::VierPol::c21` [protected]

8.81.4.4 `Cplx Ah::VierPol::c22` [protected]

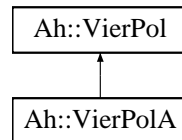
The documentation for this class was generated from the following file:

- [AH/Etech/VierPol.h](#)

8.82 Ah::VierPolA Class Reference

```
#include <VierPol.h>
```

Inheritance diagram for Ah::VierPolA::



Public Member Functions

- [VierPolA](#) ([Cplx](#) a11, [Cplx](#) a12, [Cplx](#) a21, [Cplx](#) a22)
constructor
- [VierPolA](#) (const [VierPolA](#) &vp)
copy constructor
- [VierPolA](#) (const [VierPolY](#) &vpy)
convert constructor
- [VierPolA](#) (const [VierPolH](#) &vph)
convert constructor
- [VierPolA](#) (const [VierPolZ](#) &vpz)
convert constructor
- [~VierPolA](#) ()
destructor
- [Cplx](#) a11 () const
accessors:
- [Cplx](#) a12 () const
- [Cplx](#) a21 () const
- [Cplx](#) a22 () const
- const [VierPolA](#) & [operator=](#) (const [VierPolZ](#) &vpz)
convert [VierPolZ](#) into [VierPolA](#)
- const [VierPolA](#) & [operator=](#) (const [VierPolH](#) &vph)
convert [VierPolH](#) into [VierPolA](#)
- const [VierPolA](#) & [operator=](#) (const [VierPolY](#) &vpy)
convert [VierPolY](#) into [VierPolA](#)

Friends

- bool `operator==` (const [VierPolA](#) &vp1, const [VierPolA](#) &vp2)
overloaded operator == for comparison of two { [VierPolA](#) }
- bool `operator!=` (const [VierPolA](#) &vp1, const [VierPolA](#) &vp2)
overloaded operator != for comparison of two { [VierPolA](#) }

8.82.1 Constructor & Destructor Documentation

8.82.1.1 `Ah::VierPolA::VierPolA (Cplx a11, Cplx a12, Cplx a21, Cplx a22)` [inline]

constructor

8.82.1.2 `Ah::VierPolA::VierPolA (const VierPolA &vp)` [inline]

copy constructor

8.82.1.3 `Ah::VierPolA::VierPolA (const VierPolY &vpy)`

convert constructor

8.82.1.4 `Ah::VierPolA::VierPolA (const VierPolH &vph)`

convert constructor

8.82.1.5 `Ah::VierPolA::VierPolA (const VierPolZ &vpz)`

convert constructor

8.82.1.6 `Ah::VierPolA::~~VierPolA ()` [inline]

destructor

8.82.2 Member Function Documentation

8.82.2.1 `Cplx Ah::VierPolA::a11 () const` [inline]

accessors:

8.82.2.2 [Cplx](#) Ah::VierPolA::a12 () const [inline]

8.82.2.3 [Cplx](#) Ah::VierPolA::a21 () const [inline]

8.82.2.4 [Cplx](#) Ah::VierPolA::a22 () const [inline]

8.82.2.5 const [VierPolA](#)& Ah::VierPolA::operator= (const [VierPolY](#) & vpy)

convert [VierPolY](#) into [VierPolA](#)

8.82.2.6 const [VierPolA](#)& Ah::VierPolA::operator= (const [VierPolH](#) & vph)

convert [VierPolH](#) into [VierPolA](#)

8.82.2.7 const [VierPolA](#)& Ah::VierPolA::operator= (const [VierPolZ](#) & vpz)

convert [VierPolZ](#) into [VierPolA](#)

8.82.3 Friends And Related Function Documentation

8.82.3.1 bool operator!= (const [VierPolA](#) & vp1, const [VierPolA](#) & vp2) [friend]

overloaded operator != for comparison of two { [VierPolA](#) }

8.82.3.2 bool operator== (const [VierPolA](#) & vp1, const [VierPolA](#) & vp2) [friend]

overloaded operator == for comparison of two { [VierPolA](#) }

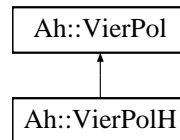
The documentation for this class was generated from the following file:

- AH/Etech/[VierPol.h](#)

8.83 Ah::VierPolH Class Reference

```
#include <VierPol.h>
```

Inheritance diagram for Ah::VierPolH::



Public Member Functions

- [VierPolH](#) ([Cplx](#) h11, [Cplx](#) h12, [Cplx](#) h21, [Cplx](#) h22)
constructor
- [VierPolH](#) (const [VierPolH](#) &vp)
copy constructor
- [VierPolH](#) (const [VierPolY](#) &vpy)
convert constructor
- [VierPolH](#) (const [VierPolZ](#) &vpz)
convert constructor
- [VierPolH](#) (const [VierPolA](#) &vpa)
convert constructor
- [~VierPolH](#) ()
destructor
- [Cplx](#) h11 () const
accessors:
- [Cplx](#) h12 () const
- [Cplx](#) h21 () const
- [Cplx](#) h22 () const
- const [VierPolH](#) & [operator=](#) (const [VierPolZ](#) &vpz)
convert [VierPolZ](#) into [VierPolH](#)
- const [VierPolH](#) & [operator=](#) (const [VierPolY](#) &vpy)
convert [VierPolY](#) into [VierPolH](#)
- const [VierPolH](#) & [operator=](#) (const [VierPolA](#) &vpa)
convert [VierPolA](#) into [VierPolH](#)

Friends

- bool **operator==** (const [VierPolH](#) &vp1, const [VierPolH](#) &vp2)
overloaded operator == for comparison of two { [VierPolH](#)}
- bool **operator!=** (const [VierPolH](#) &vp1, const [VierPolH](#) &vp2)
overloaded operator != for comparison of two { [VierPolH](#)}

8.83.1 Constructor & Destructor Documentation

8.83.1.1 Ah::VierPolH::VierPolH ([Cplx](#) h11, [Cplx](#) h12, [Cplx](#) h21, [Cplx](#) h22) [inline]

constructor

8.83.1.2 Ah::VierPolH::VierPolH (const [VierPolH](#) & vp) [inline]

copy constructor

8.83.1.3 Ah::VierPolH::VierPolH (const [VierPolY](#) & vpy)

convert constructor

8.83.1.4 Ah::VierPolH::VierPolH (const [VierPolZ](#) & vpz)

convert constructor

8.83.1.5 Ah::VierPolH::VierPolH (const [VierPolA](#) & vpa)

convert constructor

8.83.1.6 Ah::VierPolH::~~VierPolH () [inline]

destructor

8.83.2 Member Function Documentation

8.83.2.1 [Cplx](#) Ah::VierPolH::h11 () const [inline]

accessors:

8.83.2.2 [Cplx](#) [Ah::VierPolH::h12 \(\) const](#) [\[inline\]](#)

8.83.2.3 [Cplx](#) [Ah::VierPolH::h21 \(\) const](#) [\[inline\]](#)

8.83.2.4 [Cplx](#) [Ah::VierPolH::h22 \(\) const](#) [\[inline\]](#)

8.83.2.5 [const VierPolH&](#) [Ah::VierPolH::operator= \(const VierPolA & vpa\)](#)

convert [VierPolA](#) into [VierPolH](#)

8.83.2.6 [const VierPolH&](#) [Ah::VierPolH::operator= \(const VierPolY & vpy\)](#)

convert [VierPolY](#) into [VierPolH](#)

8.83.2.7 [const VierPolH&](#) [Ah::VierPolH::operator= \(const VierPolZ & vpz\)](#)

convert [VierPolZ](#) into [VierPolH](#)

8.83.3 Friends And Related Function Documentation

8.83.3.1 [bool operator!= \(const VierPolH & vp1, const VierPolH & vp2\)](#) [\[friend\]](#)

overloaded operator != for comparison of two { [VierPolH](#) }

8.83.3.2 [bool operator== \(const VierPolH & vp1, const VierPolH & vp2\)](#) [\[friend\]](#)

overloaded operator == for comparison of two { [VierPolH](#) }

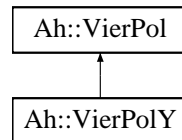
The documentation for this class was generated from the following file:

- [AH/Etech/VierPol.h](#)

8.84 Ah::VierPolY Class Reference

```
#include <VierPol.h>
```

Inheritance diagram for Ah::VierPolY::



Public Member Functions

- [VierPolY](#) ([Cplx](#) y11, [Cplx](#) y12, [Cplx](#) y21, [Cplx](#) y22)
constructor
- [VierPolY](#) (const [VierPolY](#) &vp)
copy constructor
- [VierPolY](#) (const [VierPolZ](#) &vpz)
convert constructor
- [VierPolY](#) (const [VierPolH](#) &vph)
convert constructor
- [VierPolY](#) (const [VierPolA](#) &vpa)
convert constructor
- [~VierPolY](#) ()
destructor
- [Cplx](#) y11 () const
accessors:
- [Cplx](#) y12 () const
- [Cplx](#) y21 () const
- [Cplx](#) y22 () const
- const [VierPolY](#) & [operator=](#) (const [VierPolZ](#) &vpz)
convert [VierPolZ](#) into [VierPolY](#)
- const [VierPolY](#) & [operator=](#) (const [VierPolH](#) &vph)
convert [VierPolH](#) into [VierPolY](#)
- const [VierPolY](#) & [operator=](#) (const [VierPolA](#) &vpa)
convert [VierPolA](#) into [VierPolY](#)

Friends

- bool `operator==` (const [VierPolY](#) &vp1, const [VierPolY](#) &vp2)
overloaded operator == for comparison of two { [VierPolY](#)}
- bool `operator!=` (const [VierPolY](#) &vp1, const [VierPolY](#) &vp2)
overloaded operator != for comparison of two { [VierPolY](#)}

8.84.1 Constructor & Destructor Documentation

8.84.1.1 `Ah::VierPolY::VierPolY (Cplx y1I, Cplx y12, Cplx y2I, Cplx y22)` `[inline]`

constructor

8.84.1.2 `Ah::VierPolY::VierPolY (const VierPolY &vp)` `[inline]`

copy constructor

8.84.1.3 `Ah::VierPolY::VierPolY (const VierPolZ &vpz)`

convert constructor

8.84.1.4 `Ah::VierPolY::VierPolY (const VierPolH &vph)`

convert constructor

8.84.1.5 `Ah::VierPolY::VierPolY (const VierPolA &vpa)`

convert constructor

8.84.1.6 `Ah::VierPolY::~~VierPolY ()` `[inline]`

destructor

8.84.2 Member Function Documentation

8.84.2.1 `const VierPolY & Ah::VierPolY::operator= (const VierPolA &vpa)`

convert [VierPolA](#) into [VierPolY](#)

8.84.2.2 `const VierPolY & Ah::VierPolY::operator= (const VierPolH &vph)`

convert [VierPolH](#) into [VierPolY](#)

8.84.2.3 `const VierPolY & Ah::VierPolY::operator= (const VierPolZ & vpz)`

convert VierPolZ into VierPolY

8.84.2.4 `Cplx Ah::VierPolY::y11 () const` [inline]

accessors:

8.84.2.5 `Cplx Ah::VierPolY::y12 () const` [inline]

8.84.2.6 `Cplx Ah::VierPolY::y21 () const` [inline]

8.84.2.7 `Cplx Ah::VierPolY::y22 () const` [inline]

8.84.3 Friends And Related Function Documentation

8.84.3.1 `bool operator!= (const VierPolY & vp1, const VierPolY & vp2)` [friend]

overloaded operator != for comparison of two { VierPolY }

8.84.3.2 `bool operator== (const VierPolY & vp1, const VierPolY & vp2)` [friend]

overloaded operator == for comparison of two { VierPolY }

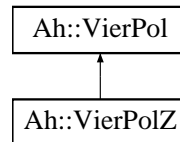
The documentation for this class was generated from the following file:

- AH/Etech/VierPol.h

8.85 Ah::VierPolZ Class Reference

```
#include <VierPol.h>
```

Inheritance diagram for Ah::VierPolZ::



Public Member Functions

- [VierPolZ](#) ([Cplx](#) z11, [Cplx](#) z12, [Cplx](#) z21, [Cplx](#) z22)
constructor
- [VierPolZ](#) (const [VierPolZ](#) &vp)
copy constructor
- [VierPolZ](#) (const [VierPolY](#) &vpy)
convert constructor
- [VierPolZ](#) (const [VierPolH](#) &vph)
convert constructor
- [VierPolZ](#) (const [VierPolA](#) &vpa)
convert constructor
- [~VierPolZ](#) ()
destructor
- [Cplx](#) z11 () const
accessors:
- [Cplx](#) z12 () const
- [Cplx](#) z21 () const
- [Cplx](#) z22 () const
- const [VierPolZ](#) & operator= (const [VierPolY](#) &vpy)
convert [VierPolY](#) into [VierPolZ](#)
- const [VierPolZ](#) & operator= (const [VierPolH](#) &vph)
convert [VierPolH](#) into [VierPolZ](#)
- const [VierPolZ](#) & operator= (const [VierPolA](#) &vpa)
convert [VierPolA](#) into [VierPolZ](#)

Friends

- `bool operator== (const VierPolZ &vp1, const VierPolZ &vp2)`
overloaded operator == for comparison of two { VierPolZ }
- `bool operator!= (const VierPolZ &vp1, const VierPolZ &vp2)`
overloaded operator != for comparison of two { VierPolZ }

8.85.1 Constructor & Destructor Documentation

8.85.1.1 Ah::VierPolZ::VierPolZ (Cplx z11, Cplx z12, Cplx z21, Cplx z22) [inline]

constructor

8.85.1.2 Ah::VierPolZ::VierPolZ (const VierPolZ &vp) [inline]

copy constructor

8.85.1.3 Ah::VierPolZ::VierPolZ (const VierPolY &vpy)

convert constructor

8.85.1.4 Ah::VierPolZ::VierPolZ (const VierPolH &vph)

convert constructor

8.85.1.5 Ah::VierPolZ::VierPolZ (const VierPolA &vpa)

convert constructor

8.85.1.6 Ah::VierPolZ::~~VierPolZ () [inline]

destructor

8.85.2 Member Function Documentation

8.85.2.1 const VierPolZ& Ah::VierPolZ::operator= (const VierPolA &vpa)

convert VierPolA into VierPolZ

8.85.2.2 const VierPolZ& Ah::VierPolZ::operator= (const VierPolH &vph)

convert VierPolH into VierPolZ

8.85.2.3 `const VierPolZ & Ah::VierPolZ::operator= (const VierPolY & vpy)`

convert [VierPolY](#) into [VierPolZ](#)

8.85.2.4 `Cplx Ah::VierPolZ::z11 () const` [inline]

accessors:

8.85.2.5 `Cplx Ah::VierPolZ::z12 () const` [inline]

8.85.2.6 `Cplx Ah::VierPolZ::z21 () const` [inline]

8.85.2.7 `Cplx Ah::VierPolZ::z22 () const` [inline]

8.85.3 Friends And Related Function Documentation

8.85.3.1 `bool operator!= (const VierPolZ & vp1, const VierPolZ & vp2)` [friend]

overloaded operator != for comparison of two { [VierPolZ](#) }

8.85.3.2 `bool operator== (const VierPolZ & vp1, const VierPolZ & vp2)` [friend]

overloaded operator == for comparison of two { [VierPolZ](#) }

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

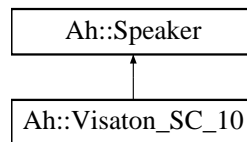
- [AH/Etech/VierPol.h](#)

8.86 Ah::Visaton_SC_10 Class Reference

represents the speaker VISATON SC 10

```
#include <VisatonSpeakers.h>
```

Inheritance diagram for Ah::Visaton_SC_10::



Public Member Functions

- [Visaton_SC_10 \(\)](#)
default constructor
- virtual [~Visaton_SC_10 \(\)](#)
destructor
- virtual [Dbl getSpl \(Dbl freq\) const](#)
get acoustic power level dB/W/m at specified frequency

8.86.1 Detailed Description

represents the speaker VISATON SC 10

{ Visaton } represents the 25 mm dome tweeter SC 10 which is made by Visaton.

8.86.2 Constructor & Destructor Documentation

8.86.2.1 Ah::Visaton_SC_10::Visaton_SC_10 ()

default constructor

8.86.2.2 virtual Ah::Visaton_SC_10::~~Visaton_SC_10 () [inline, virtual]

destructor

8.86.3 Member Function Documentation

8.86.3.1 virtual [Dbl](#) Ah::Visaton_SC_10::getSpl ([Dbl](#) freq) const [virtual]

get acoustic power level dB/W/m at specified frequency

Reimplemented from [Ah::Speaker](#).

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

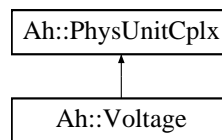
- AH/Etech/[VisatonSpeakers.h](#)

8.87 Ah::Voltage Class Reference

represents an electrical voltage

```
#include <Voltage.h>
```

Inheritance diagram for Ah::Voltage::



Public Member Functions

- [Voltage](#) ([Dbl](#) r=0.0, [Dbl](#) i=0.0)
constructor
- [Voltage](#) (const [Cplx](#) &val)
constructor
- [Voltage](#) (const [Voltage](#) &u)
copy constructor

Friends

- [Voltage operator+](#) (const [Voltage](#) &u1, const [Voltage](#) &u2)
overloaded operator + for addition of two voltages
- [Voltage operator-](#) (const [Voltage](#) &u1, const [Voltage](#) &u2)
overloaded operator - for subtraction of two voltages
- [Voltage operator *](#) (const [Voltage](#) &u1, const [Cplx](#) &k)
*overloaded operator * for multiplication of { [Voltage](#) } * { [Cplx](#) }*
- [Voltage operator *](#) (const [Cplx](#) &k, const [Voltage](#) &u1)
*overloaded operator * for multiplication of { [Cplx](#) } * { [Voltage](#) }*
- [Voltage operator/](#) (const [Voltage](#) &u1, const [Cplx](#) &k)
overloaded operator / for division of { [Voltage](#) } / { [Cplx](#) }
- [bool operator==](#) (const [Voltage](#) &u1, const [Voltage](#) &u2)
overloaded operator == for comparison of two Voltages
- [bool operator!=](#) (const [Voltage](#) &u1, const [Voltage](#) &u2)
overloaded operator != for comparison of two Voltages
- [bool operator<](#) (const [Voltage](#) &u1, const [Voltage](#) &u2)

overloaded operator < for comparison of two Voltages

- `bool operator> (const Voltage &u1, const Voltage &u2)`
overloaded operator > for comparison of two Voltages

8.87.1 Detailed Description

represents an electrical voltage

{ `Voltage` } represents an electrical voltage which can be used both in DC (direct current) and in AC (alternate current) applications.

8.87.2 Constructor & Destructor Documentation

8.87.2.1 `Ah::Voltage::Voltage (Dbl r = 0.0, Dbl i = 0.0)` [inline]

constructor

8.87.2.2 `Ah::Voltage::Voltage (const Cplx & val)` [inline]

constructor

8.87.2.3 `Ah::Voltage::Voltage (const Voltage & u)` [inline]

copy constructor

8.87.3 Friends And Related Function Documentation

8.87.3.1 `Voltage operator * (const Cplx & k, const Voltage & u1)` [friend]

overloaded operator * for multiplication of { `Cplx` } * { `Voltage` }

8.87.3.2 `Voltage operator * (const Voltage & u1, const Cplx & k)` [friend]

overloaded operator * for multiplication of { `Voltage` } * { `Cplx` }

8.87.3.3 `bool operator!= (const Voltage & u1, const Voltage & u2)` [friend]

overloaded operator != for comparison of two Voltages

8.87.3.4 `Voltage operator+ (const Voltage & u1, const Voltage & u2)` [friend]

overloaded operator + for addition of two voltages

8.87.3.5 `Voltage operator- (const Voltage & u1, const Voltage & u2)` [friend]

overloaded operator - for subtraction of two voltages

8.87.3.6 Voltage operator/ (const Voltage & u1, const Cplx & k) [friend]

overloaded operator / for division of { Voltage } / { Cplx }

8.87.3.7 bool operator< (const Voltage & u1, const Voltage & u2) [friend]

overloaded operator < for comparison of two Voltages

8.87.3.8 bool operator== (const Voltage & u1, const Voltage & u2) [friend]

overloaded operator == for comparison of two Voltages

8.87.3.9 bool operator> (const Voltage & u1, const Voltage & u2) [friend]

overloaded operator > for comparison of two Voltages

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

- AH/Etech/[Voltage.h](#)

Chapter 9

libAHetech File Documentation

9.1 AH/Etech/aaa.h File Reference

```
#include <AH/Config/Platform.h>
#include <complex>
```

Namespaces

- namespace [Ah](#)
- namespace [AH](#)

Classes

- class [Ah::Cplx](#)
declaration of a complex number

Typedefs

- typedef double [Ah::Dbl](#)
declaration of a scalar number

9.2 AH/Etech/Algo.h File Reference

```
#include <AH/Config/Platform.h>
#include <AH/Etech/Voltage.h>
#include <AH/Etech/Current.h>
#include <AH/Etech/Power.h>
#include <AH/Etech/Impedance.h>
```

Namespaces

- namespace [Ah](#)

Functions

- [Impedance Ah::operator/](#) (const [Voltage](#) &u, const [Current](#) &i)
overloaded operator / for division of { [Voltage](#) } / { [Current](#) }
- [Voltage Ah::operator *](#) (const [Impedance](#) &r, const [Current](#) &i)
*overloaded operator * for multiplication of { [Impedance](#) } * { [Current](#) }*
- [Voltage Ah::operator *](#) (const [Current](#) &i, const [Impedance](#) &r)
*overloaded operator * for multiplication of { [Current](#) } * { [Impedance](#) }*
- [Voltage Ah::operator/](#) (const [Power](#) &p, const [Current](#) &i)
overloaded operator / for division of { [Power](#) } / { [Current](#) }
- [Current Ah::operator/](#) (const [Voltage](#) &u, const [Impedance](#) &r)
overloaded operator / for division of { [Voltage](#) } / { [Impedance](#) }
- [Current Ah::operator *](#) (const [Power](#) &p, const [Voltage](#) &u)
overloaded operator / for division of { [Power](#) } / { [Voltage](#) }
- [Power Ah::operator *](#) (const [Voltage](#) &u, const [Current](#) &i)
*overloaded operator * for multiplication of { [Voltage](#) } * { [Current](#) }*
- [Power Ah::operator *](#) (const [Current](#) &i, const [Voltage](#) &u)
*overloaded operator * for multiplication of { [Current](#) } * { [Voltage](#) }*
- [Dbl Ah::inVdB](#) ([Dbl](#) x)
return value in voltage decibel
- [Dbl Ah::inPdB](#) ([Dbl](#) x)
return value in power decibel
- [Dbl Ah::inDeg](#) ([Dbl](#) x)
return phase argument in degree

9.3 AH/Etech/Amplifier.h File Reference

```
#include <AH/Config/Platform.h>
#include <AH/Etech/Types.h>
```

Namespaces

- namespace [Ah](#)

Classes

- class [Ah::Amplifier](#)
pure virtual generic amplifier base class

9.4 AH/Etech/AnalogDevOpAmps.h File Reference

```
#include <AH/Config/Platform.h>
#include <AH/Etech/OpAmp.h>
```

Namespaces

- namespace [Ah](#)

Classes

- class [Ah::AD711](#)
represents the operational amplifier [AD711](#)
- class [Ah::AD745](#)
represents the ultra low noise operational amplifier [AD745](#)
- class [Ah::AD797](#)
represents the ultra low noise operational amplifier [AD797](#)
- class [Ah::AD845](#)
represents the fast precision operational amplifier [AD845](#)

9.5 AH/Etech/AnalogDevOpCreat.h File Reference

```
#include <AH/Config/Platform.h>
```

Namespaces

- namespace [Ah](#)

Classes

- class [Ah::AnalogDevOpAmpCreator](#)
represents the creator class for Analog Devices operational amplifiers

9.6 AH/Etech/Box.h File Reference

```
#include <AH/Config/Platform.h>
```

Namespaces

- namespace [Ah](#)

Classes

- class [Ah::Box](#)
represents a loudspeaker box

9.7 AH/Etech/BPT.h File Reference

```
#include <AH/Config/Platform.h>
#include <AH/Etech/Transistor.h>
#include <AH/Etech/Voltage.h>
```

Namespaces

- namespace [Ah](#)

Classes

- class [Ah::BPT](#)
represents a bipolar transistor

9.8 AH/Etech/BurrBrownOpAmps.h File Reference

```
#include <AH/Config/Platform.h>
#include <AH/Etech/OpAmp.h>
```

Namespaces

- namespace [Ah](#)

Classes

- class [Ah::OPA134](#)
represents the Sound-Plus operational amplifier [OPA134](#)
- class [Ah::OPA604](#)
represents the audio operational amplifier [OPA604](#)

9.9 AH/Etech/BurrBrownOpCreat.h File Reference

```
#include <AH/Config/Platform.h>
```

Namespaces

- namespace [Ah](#)

Classes

- class [Ah::BurrBrownOpAmpCreator](#)
represents the creator class for Burr-Brown operational amplifiers

9.10 AH/Etech/Capacitor.h File Reference

```
#include <AH/Config/Platform.h>
#include <AH/Etech/FreqImp.h>
```

Namespaces

- namespace [Ah](#)

Classes

- class [Ah::Capacitor](#)
represents an ideal electrical capacitor

9.11 AH/Etech/Cd.h File Reference

```
#include <AH/Config/Platform.h>
#include <AH/Etech/Types.h>
```

Namespaces

- namespace [Ah](#)

Functions

- [Cplx Ah::CdPreEmphasis](#) (Dbl f)
return CD preemphasis value at frequency { f}
- [Cplx Ah::CdDeEmphasis](#) (Dbl f)
return CD deemphasis value at frequency { f}

9.12 AH/Etech/ClosedBox.h File Reference

```
#include <AH/Config/Platform.h>
#include <AH/Etech/Types.h>
#include <AH/Etech/Box.h>
```

Namespaces

- namespace [Ah](#)

Classes

- class [Ah::ClosedBox](#)
represents a closed loudspeaker box

9.13 AH/Etech/Current.h File Reference

```
#include <AH/Config/Platform.h>
#include <AH/Etech/PhysUnit.h>
```

Namespaces

- namespace [Ah](#)

Classes

- class [Ah::Current](#)
represents an electrical current

9.14 AH/Etech/Divider.h File Reference

```
#include <AH/Config/Platform.h>
#include <AH/Etech/Impedance.h>
```

Namespaces

- namespace [Ah](#)

Classes

- class [Ah::Divider](#)
represents a voltage divider

9.15 AH/Etech/Etech.h File Reference

```
#include <AH/Etech/Voltage.h>
#include <AH/Etech/Current.h>
#include <AH/Etech/Power.h>
#include <AH/Etech/Impedance.h>
#include <AH/Etech/Algo.h>
#include <AH/Etech/Resistor.h>
#include <AH/Etech/Capacitor.h>
#include <AH/Etech/Inductor.h>
```

9.16 AH/Etech/FET.h File Reference

```
#include <AH/Config/Platform.h>
#include <AH/Etech/Transistor.h>
#include <AH/Etech/Voltage.h>
```

Namespaces

- namespace [Ah](#)

Classes

- class [Ah::FET](#)
represents a [FET](#)

9.17 AH/Etech/Filter2PoleInv.h File Reference

```
#include <AH/Config/Platform.h>
#include <AH/Etech/OpAmp.h>
#include <AH/Etech/Amplifier.h>
```

Namespaces

- namespace [Ah](#)

Classes

- class [Ah::Filter2PoleInv](#)
2-pole filter using multiple feedback on inverting amplifier

9.18 AH/Etech/FreqImp.h File Reference

```
#include <AH/Config/Platform.h>
```

```
#include <AH/Etech/Impedance.h>
```

Namespaces

- namespace [Ah](#)

Classes

- class [Ah::FreqImpedance](#)
represents an frequency dependent electrical impedance base class

9.19 AH/Etech/Impedance.h File Reference

```
#include <AH/Config/Platform.h>
#include <AH/Etech/PhysUnit.h>
```

Namespaces

- namespace [Ah](#)

Classes

- class [Ah::Impedance](#)
represents an electrical impedance

9.20 AH/Etech/Inductor.h File Reference

```
#include <AH/Config/Platform.h>
#include <AH/Etech/FreqImp.h>
```

Namespaces

- namespace [Ah](#)

Classes

- class [Ah::Inductor](#)
represents an ideal electrical inductor

9.21 AH/Etech/IntPol.h File Reference

```
#include <AH/Config/Platform.h>
#include <AH/Etech/Types.h>
#include <vector>
```

Namespaces

- namespace [Ah](#)

Classes

- struct [Ah::IntPolData](#)
- class [Ah::IntPol](#)
represents a pure virtual interpolation class
- class [Ah::IntPolLin1Lin2](#)
represents a linear interpolation class
- class [Ah::IntPolLog1Lin2](#)
represents a logarithmic interpolation class

9.22 AH/Etech/InvAmp.h File Reference

```
#include <AH/Config/Platform.h>
#include <AH/Etech/Amplifier.h>
#include <AH/Etech/OpAmp.h>
```

Namespaces

- namespace [Ah](#)

Classes

- class [Ah::InvAmp](#)
represents an inverting amplifier

9.23 AH/Etech/KefSpeakers.h File Reference

```
#include <AH/Config/Platform.h>
#include <AH/Etech/Speaker.h>
```

Namespaces

- namespace [Ah](#)

Classes

- class [Ah::Kef_B110_A](#)
represents the speaker KEF B 110 SP 1003
- class [Ah::Kef_B110_B](#)
represents the speaker KEF B 110 SP 1057
- class [Ah::Kef_B200_A](#)
represents the speaker KEF B 200 SP 1014
- class [Ah::Kef_B200_B](#)
represents the speaker KEF B 200 SP 1039
- class [Ah::Kef_B139_B](#)
represents the speaker KEF B 139 SP 1044
- class [Ah::Kef_B300_B](#)
represents the speaker KEF B 300 SP 1071

9.24 AH/Etech/LinTechOpAmps.h File Reference

```
#include <AH/Config/Platform.h>
#include <AH/Etech/OpAmp.h>
```

Namespaces

- namespace [Ah](#)

Classes

- class [Ah::LT1028](#)
represents the very fast high precision operational amplifier [LT1028](#)
- class [Ah::LT1115](#)
represents the audio operational amplifier [LT1115](#)

9.25 AH/Etech/LinTechOpCreat.h File Reference

```
#include <AH/Config/Platform.h>
```

Namespaces

- namespace [Ah](#)

Classes

- class [Ah::LinTechOpAmpCreator](#)
represents the creator class for Linear Technology operational amplifiers

9.26 AH/Etech/MathConst.h File Reference

```
#include <AH/Config/Platform.h>
#include <AH/Etech/Types.h>
```

Namespaces

- namespace [Ah](#)

Classes

- class [Ah::MathConst](#)
represents some mathical constants

Defines

- #define [MathConst_PI](#) 3.14159265359

9.26.1 Define Documentation

9.26.1.1 #define MathConst_PI 3.14159265359

9.27 AH/Etech/MiscOpAmps.h File Reference

```
#include <AH/Config/Platform.h>
#include <AH/Etech/OpAmp.h>
```

Namespaces

- namespace [Ah](#)

Classes

- class [Ah::LM833](#)
represents the audio operational amplifier [LM833](#)
- class [Ah::NE5532](#)
represents the audio operational amplifier [NE5532](#)
- class [Ah::OP07](#)
represents the industry standard precision operational amplifier [OP07](#)
- class [Ah::OP27](#)
represents the fast industry standard precision operational amplifier [OP27](#)
- class [Ah::OP37](#)
represents the very fast industry standard precision operational amplifier [OP37](#)
- class [Ah::OP77](#)
represents the industry standard high precision operational amplifier [OP77](#)
- class [Ah::OP177](#)
represents the industry standard high precision operational amplifier [OP177](#)

9.28 AH/Etech/MiscOpCreat.h File Reference

```
#include <AH/Config/Platform.h>
```

Namespaces

- namespace [Ah](#)

Classes

- class [Ah::MiscOpAmpCreator](#)
represents the [OpAmp](#) creator class

9.29 AH/Etech/MonacorSpeakers.h File Reference

```
#include <AH/Config/Platform.h>
#include <AH/Etech/Speaker.h>
```

Namespaces

- namespace [Ah](#)

Classes

- class [Ah::Monacor_SP45_4](#)
represents the speaker Monacor SP-45/4
- class [Ah::Monacor_SP45_8](#)
represents the speaker Monacor SP-45/8
- class [Ah::Monacor_SP60_4](#)
represents the speaker Monacor SP-60/4
- class [Ah::Monacor_SP60_8](#)
represents the speaker Monacor SP-60/8
- class [Ah::Monacor_SPP110_4](#)
represents the speaker Monacor SPP-110/4
- class [Ah::Monacor_SPP110_8](#)
represents the speaker Monacor SPP-110/8

9.30 AH/Etech/NoneInvAmp.h File Reference

```
#include <AH/Config/Platform.h>
#include <AH/Etech/Amplifier.h>
#include <AH/Etech/OpAmp.h>
```

Namespaces

- namespace [Ah](#)

Classes

- class [Ah::NoneInvAmp](#)
represents a none inverting amplifier

9.31 AH/Etech/OpAmp.h File Reference

```
#include <AH/Config/Platform.h>
```

```
#include <AH/Etech/Impedance.h>
```

Namespaces

- namespace [Ah](#)

Classes

- class [Ah::OpAmp](#)
represents an operational amplifier

9.32 AH/Etech/OpAmpCreator.h File Reference

```
#include <AH/Config/Platform.h>
```

Namespaces

- namespace [Ah](#)

Classes

- class [Ah::OpAmpCreator](#)
represents the [OpAmp](#) creator class

9.33 AH/Etech/PhysConst.h File Reference

```
#include <AH/Config/Platform.h>
#include <AH/Etech/Types.h>
```

Namespaces

- namespace [Ah](#)

Classes

- class [Ah::PhysConst](#)
represents some physical constants

Defines

- #define [PhysConst_T25](#) 298.13
- #define [PhysConst_BOLTZMANN](#) 1.381e-23
- #define [PhysConst_EPS0](#) 1.6e-19

9.33.1 Define Documentation

9.33.1.1 #define [PhysConst_BOLTZMANN](#) 1.381e-23

9.33.1.2 #define [PhysConst_EPS0](#) 1.6e-19

9.33.1.3 #define [PhysConst_T25](#) 298.13

9.34 AH/Etech/PhysUnit.h File Reference

```
#include <AH/Config/Platform.h>
#include <cmath>
#include <AH/Etech/Types.h>
```

Namespaces

- namespace [Ah](#)

Classes

- class [Ah::PhysUnitDbl](#)
represents a scalar physical unit
- class [Ah::PhysUnitCplx](#)
represents a complex physical unit

9.35 AH/Etech/PickUp.h File Reference

```
#include <AH/Config/Platform.h>
#include <AH/Etech/Impedance.h>
#include <AH/Etech/Resistor.h>
#include <AH/Etech/Capacitor.h>
#include <AH/Etech/Inductor.h>
```

Namespaces

- namespace [Ah](#)

Classes

- class [Ah::PickUp](#)
represents a pickup of a record player
- class [Ah::PickUpOrtofon540mk2](#)
Ortofon 540 MK2 (example supplied by Frank Klemm in drmh).
- class [Ah::PickUpVdhDDT2Special](#)
Van den Hul DDT II Special (data supplied by Frank Klemm).
- class [Ah::PickUpVdhFrogCopper](#)
Van den Hul Frog Copper (data supplied by Frank Klemm).
- class [Ah::PickUpVdhBlackBeautySPX](#)
Van den Hul Black Beauty SPX (data supplied by Frank Klemm).
- class [Ah::PickUpVdhColibriXGP](#)
Van den Hul The Colibri XGP (data supplied by Frank Klemm).
- class [Ah::PickUpClearaudioAlphaWood](#)
Clearaudio Alpha Wood (data supplied by Frank Klemm).
- class [Ah::PickUpClearaudioVirtuosoWood](#)
Clearaudio Virtuoso Wood (data supplied by Frank Klemm).
- class [Ah::PickUpClearaudioGammaSGold](#)
Clearaudio Gamma S Gold (data supplied by Frank Klemm).
- class [Ah::PickUpClearaudioDiscovery](#)
Clearaudio Discovery (data supplied by Frank Klemm).
- class [Ah::PickUpGradoPlatinum](#)
Grado Platinum (data supplied by Frank Klemm).

- class [Ah::PickUpGradoMaster](#)
Grado Master (data supplied by Frank Klemm).
- class [Ah::PickUpGradoReference](#)
Grado Reference (data supplied by Frank Klemm).
- class [Ah::PickUpGradoStatement](#)
Grado The Statement (data supplied by Frank Klemm).

9.36 AH/Etech/Power.h File Reference

```
#include <AH/Config/Platform.h>
#include <AH/Etech/PhysUnit.h>
```

Namespaces

- namespace [Ah](#)

Classes

- class [Ah::Power](#)
represents an electrical power

9.37 AH/Etech/Resistor.h File Reference

```
#include <AH/Config/Platform.h>
#include <AH/Etech/Impedance.h>
```

Namespaces

- namespace [Ah](#)

Classes

- class [Ah::Resistor](#)
represents an ideal electrical resistor

9.38 AH/Etech/Riaa.h File Reference

```
#include <AH/Config/Platform.h>
#include <AH/Etech/Types.h>
```

Namespaces

- namespace [Ah](#)

Functions

- [Cplx Ah::RiaaPreEmphasis](#) (Dbl f)
return RIAA preemphasis value at frequency { f}
- [Cplx Ah::RiaaDeEmphasis](#) (Dbl f)
return RIAA deemphasis value at frequency { f}

9.39 AH/Etech/SalKey.h File Reference

```
#include <AH/Config/Platform.h>
#include <AH/Etech/Impedance.h>
#include <AH/Etech/OpAmp.h>
#include <AH/Etech/Amplifier.h>
```

Namespaces

- namespace [Ah](#)

Classes

- class [Ah::SalKey](#)
Sallen-Key Filter.

9.40 AH/Etech/Speaker.h File Reference

```
#include <AH/Config/Platform.h>
#include <AH/Etech/Types.h>
#include <AH/Etech/Impedance.h>
```

Namespaces

- namespace [Ah](#)

Classes

- class [Ah::Speaker](#)
represents a pure speaker system

9.41 AH/Etech/SpeakerCreator.h File Reference

```
#include <AH/Config/Platform.h>  
#include <AH/Etech/Speaker.h>
```

Namespaces

- namespace [Ah](#)

Classes

- class [Ah::SpeakerCreator](#)
represents the speaker creator class

9.42 AH/Etech/TransFollow.h File Reference

```
#include <AH/Config/Platform.h>
#include <AH/Etech/Voltage.h>
#include <AH/Etech/Current.h>
```

Namespaces

- namespace [Ah](#)

Classes

- class [Ah::TransistorFollower](#)
represents a generic transistor follower

9.43 AH/Etech/Transistor.h File Reference

```
#include <AH/Config/Platform.h>
#include <AH/Etech/PhysConst.h>
#include <AH/Etech/Current.h>
```

Namespaces

- namespace [Ah](#)

Classes

- class [Ah::Transistor](#)
represents a generic transistor

9.44 AH/Etech/Types.h File Reference

```
#include <AH/Config/Platform.h>
#include <complex>
```

Namespaces

- namespace [Ah](#)

Classes

- class [Ah::Cplx](#)
declaration of a complex number

Typedefs

- typedef std::complex< double > [Ah::double_complex](#)

9.45 AH/Etech/VdHPickUps.h File Reference

```
#include <AH/Config/Platform.h>
#include <AH/Etech/PickUp.h>
```

Namespaces

- namespace [Ah](#)

Classes

- class [Ah::PickUpVdhMc1Special](#)
Van den Hul MC-1 Special.
- class [Ah::PickUpVdhMm1](#)
Van den Hul MM-1.

9.46 AH/Etech/Version.h File Reference

Defines

- #define [AH_ETECH_VERSION](#) "1.0"

9.46.1 Define Documentation

9.46.1.1 #define AH_ETECH_VERSION "1.0"

9.47 AH/Etech/VierPol.h File Reference

```
#include <AH/Config/Platform.h>
#include <AH/Etech/Types.h>
```

Namespaces

- namespace [Ah](#)

Classes

- class [Ah::VierPol](#)
- class [Ah::VierPolZ](#)
- class [Ah::VierPolY](#)
- class [Ah::VierPolH](#)
- class [Ah::VierPolA](#)

9.48 AH/Etech/VisatonSpeakers.h File Reference

```
#include <AH/Config/Platform.h>
#include <AH/Etech/Speaker.h>
```

Namespaces

- namespace [Ah](#)

Classes

- class [Ah::Visaton_SC_10](#)
represents the speaker VISATON SC 10

9.49 AH/Etech/Voltage.h File Reference

```
#include <AH/Config/Platform.h>
#include <AH/Etech/PhysUnit.h>
```

Namespaces

- namespace [Ah](#)

Classes

- class [Ah::Voltage](#)
represents an electrical voltage

Index

- ~Amplifier
 - Ah::Amplifier, [31](#)
- ~BPT
 - Ah::BPT, [37](#)
- ~Box
 - Ah::Box, [34](#)
- ~ClosedBox
 - Ah::ClosedBox, [42](#)
- ~FET
 - Ah::FET, [51](#)
- ~Filter2PoleInv
 - Ah::Filter2PoleInv, [54](#)
- ~FreqImpedance
 - Ah::FreqImpedance, [56](#)
- ~IntPol
 - Ah::IntPol, [63](#)
- ~InvAmp
 - Ah::InvAmp, [68](#)
- ~Kef_B110_A
 - Ah::Kef_B110_A, [70](#)
- ~Kef_B110_B
 - Ah::Kef_B110_B, [71](#)
- ~Kef_B139_B
 - Ah::Kef_B139_B, [72](#)
- ~Kef_B200_A
 - Ah::Kef_B200_A, [73](#)
- ~Kef_B200_B
 - Ah::Kef_B200_B, [74](#)
- ~Kef_B300_B
 - Ah::Kef_B300_B, [75](#)
- ~Monacor_SP45_4
 - Ah::Monacor_SP45_4, [82](#)
- ~Monacor_SP45_8
 - Ah::Monacor_SP45_8, [83](#)
- ~Monacor_SP60_4
 - Ah::Monacor_SP60_4, [84](#)
- ~Monacor_SP60_8
 - Ah::Monacor_SP60_8, [85](#)
- ~Monacor_SPP110_4
 - Ah::Monacor_SPP110_4, [86](#)
- ~Monacor_SPP110_8
 - Ah::Monacor_SPP110_8, [87](#)
- ~NoneInvAmp
 - Ah::NoneInvAmp, [91](#)
- ~OpAmp
 - Ah::OpAmp, [101](#)
- ~OpAmpCreator
 - Ah::OpAmpCreator, [103](#)
- ~PhysUnitDbl
 - Ah::PhysUnitDbl, [111](#)
- ~PickUp
 - Ah::PickUp, [113](#)
- ~PickUpClearaudioAlphaWood
 - Ah::PickUpClearaudioAlphaWood, [115](#)
- ~PickUpClearaudioDiscovery
 - Ah::PickUpClearaudioDiscovery, [116](#)
- ~PickUpClearaudioGammaSGold
 - Ah::PickUpClearaudioGammaSGold, [117](#)
- ~PickUpClearaudioVirtuosoWood
 - Ah::PickUpClearaudioVirtuosoWood, [118](#)
- ~PickUpGradoMaster
 - Ah::PickUpGradoMaster, [119](#)
- ~PickUpGradoPlatinum
 - Ah::PickUpGradoPlatinum, [120](#)
- ~PickUpGradoReference
 - Ah::PickUpGradoReference, [121](#)
- ~PickUpGradoStatement
 - Ah::PickUpGradoStatement, [122](#)
- ~PickUpOrtofon540mk2
 - Ah::PickUpOrtofon540mk2, [123](#)
- ~PickUpVdhBlackBeautySPX
 - Ah::PickUpVdhBlackBeautySPX, [124](#)
- ~PickUpVdhColibriXGP
 - Ah::PickUpVdhColibriXGP, [125](#)
- ~PickUpVdhDDT2Special
 - Ah::PickUpVdhDDT2Special, [126](#)
- ~PickUpVdhFrogCopper
 - Ah::PickUpVdhFrogCopper, [127](#)
- ~PickUpVdhMc1Special
 - Ah::PickUpVdhMc1Special, [128](#)
- ~PickUpVdhMm1
 - Ah::PickUpVdhMm1, [129](#)
- ~SalKey
 - Ah::SalKey, [136](#)
- ~Speaker
 - Ah::Speaker, [140](#)
- ~SpeakerCreator
 - Ah::SpeakerCreator, [142](#)
- ~Transistor
 - Ah::Transistor, [143](#)

- ~TransistorFollower
 - Ah::TransistorFollower, 145
- ~VierPol
 - Ah::VierPol, 148
- ~VierPolA
 - Ah::VierPolA, 150
- ~VierPolH
 - Ah::VierPolH, 153
- ~VierPolY
 - Ah::VierPolY, 156
- ~VierPolZ
 - Ah::VierPolZ, 159
- ~Visaton_SC_10
 - Ah::Visaton_SC_10, 161
- A0
 - Ah::OpAmp, 102
- a11
 - Ah::VierPolA, 150
- a12
 - Ah::VierPolA, 150
- a21
 - Ah::VierPolA, 151
- a22
 - Ah::VierPolA, 151
- abs
 - Ah::PhysUnitCplx, 107
 - Ah::PhysUnitDbl, 111
- AD711
 - Ah::AD711, 27
- AD745
 - Ah::AD745, 28
- AD797
 - Ah::AD797, 29
- AD845
 - Ah::AD845, 30
- Af
 - Ah::OpAmp, 102
- AH, 17
- Ah, 18
 - CdDeEmphasis, 24
 - CdPreEmphasis, 24
 - Dbl, 24
 - double_complex, 24
 - inDeg, 24
 - inPdB, 24
 - inVdB, 24
 - operator *, 24, 25
 - operator/, 25
 - RiaaDeEmphasis, 25
 - RiaaPreEmphasis, 25
- AH/ Directory Reference, 15
- AH/Etech/ Directory Reference, 16
- AH/Etech/aaa.h, 167
- AH/Etech/Algo.h, 168
- AH/Etech/Amplifier.h, 169
- AH/Etech/AnalogDevOpAmps.h, 170
- AH/Etech/AnalogDevOpCreat.h, 171
- AH/Etech/Box.h, 172
- AH/Etech/BPT.h, 173
- AH/Etech/BurrBrownOpAmps.h, 174
- AH/Etech/BurrBrownOpCreat.h, 175
- AH/Etech/Capacitor.h, 176
- AH/Etech/Cd.h, 177
- AH/Etech/ClosedBox.h, 178
- AH/Etech/Current.h, 179
- AH/Etech/Divider.h, 180
- AH/Etech/Etech.h, 181
- AH/Etech/FET.h, 182
- AH/Etech/Filter2PoleInv.h, 183
- AH/Etech/FreqImp.h, 184
- AH/Etech/Impedance.h, 185
- AH/Etech/Inductor.h, 186
- AH/Etech/IntPol.h, 187
- AH/Etech/InvAmp.h, 188
- AH/Etech/KefSpeakers.h, 189
- AH/Etech/LinTechOpAmps.h, 190
- AH/Etech/LinTechOpCreat.h, 191
- AH/Etech/MathConst.h, 192
- AH/Etech/MiscOpAmps.h, 193
- AH/Etech/MiscOpCreat.h, 194
- AH/Etech/MonacorSpeakers.h, 195
- AH/Etech/NoneInvAmp.h, 196
- AH/Etech/OpAmp.h, 197
- AH/Etech/OpAmpCreator.h, 198
- AH/Etech/PhysConst.h, 199
- AH/Etech/PhysUnit.h, 200
- AH/Etech/PickUp.h, 201
- AH/Etech/Power.h, 203
- AH/Etech/Resistor.h, 204
- AH/Etech/Riaa.h, 205
- AH/Etech/SalKey.h, 206
- AH/Etech/Speaker.h, 207
- AH/Etech/SpeakerCreator.h, 208
- AH/Etech/TransFollow.h, 209
- AH/Etech/Transistor.h, 210
- AH/Etech/Types.h, 211
- AH/Etech/VdHPickUps.h, 212
- AH/Etech/Version.h, 213
- AH/Etech/VierPol.h, 214
- AH/Etech/VisatonSpeakers.h, 215
- AH/Etech/Voltage.h, 216
- Ah::AD711, 27
 - AD711, 27
- Ah::AD745, 28
 - AD745, 28
- Ah::AD797, 29
 - AD797, 29

- Ah::AD845, 30
 - AD845, 30
- Ah::Amplifier, 31
 - ~Amplifier, 31
 - getA, 31
- Ah::AnalogDevOpAmpCreator, 33
- Ah::AnalogDevOpAmpCreator
 - create, 33
- Ah::Box, 34
 - ~Box, 34
 - Box, 34
 - getName, 35
 - getSpeaker, 35
 - sp, 35
- Ah::BPT, 36
 - ~BPT, 37
 - BPT, 37
 - getIout, 37
 - getS, 37
 - getUt, 37
 - Ics, 37
- Ah::BurrBrownOpAmpCreator, 38
- Ah::BurrBrownOpAmpCreator
 - create, 38
- Ah::Capacitor, 39
 - calcVal, 40
 - Capacitor, 40
 - getCap, 40
 - operator *, 40
 - operator+, 40
 - operator/, 40
 - operator<=, 41
 - operator | |, 41
- Ah::ClosedBox, 42
- Ah::ClosedBox
 - ~ClosedBox, 42
 - calc, 43
 - ClosedBox, 42
 - damped, 43
 - getQt, 43
 - Qt, 43
- Ah::Cplx, 44, 45
- Ah::Current, 46
 - Current, 47
 - operator *, 47
 - operator!=, 47
 - operator+, 47
 - operator-, 47
 - operator/, 47
 - operator<, 48
 - operator==, 48
 - operator>, 48
- Ah::Divider, 49
 - Divider, 49
 - getAtt, 49
 - getZin, 49
- Ah::FET, 50
 - ~FET, 51
 - FET, 51
 - getIout, 51
 - getS, 51
 - getUp, 51
 - Ids, 52
 - Up25, 52
- Ah::Filter2PoleInv, 53
- Ah::Filter2PoleInv
 - ~Filter2PoleInv, 54
 - calcLowpass, 54
 - Filter2PoleInv, 54
 - getA, 54
- Ah::FreqImpedance, 55
- Ah::FreqImpedance
 - ~FreqImpedance, 56
 - calcVal, 56
 - FreqImpedance, 56
 - operator(), 56
- Ah::Impedance, 57
 - Impedance, 58
 - operator *, 58, 59
 - operator!=, 59
 - operator+, 59
 - operator-, 59
 - operator/, 59
 - operator<, 59
 - operator==, 59
 - operator>, 59
 - operator | |, 59
- Ah::Inductor, 60
 - calcVal, 61
 - getInd, 61
 - Inductor, 61
 - operator *, 61
 - operator+, 61
 - operator/, 61
 - operator<=, 62
 - operator | |, 62
- Ah::IntPol, 63
- Ah::IntPol
 - ~IntPol, 63
 - getY, 64
 - interpolate, 64
 - IntPol, 63
 - table, 64
- Ah::IntPolData, 65
- Ah::IntPolData
 - x, 65
 - y, 65
- Ah::IntPolLin1Lin2, 66

- Ah::IntPolLin1Lin2
 - interpolate, [66](#)
 - IntPolLin1Lin2, [66](#)
- Ah::IntPolLog1Lin2, [67](#)
- Ah::IntPolLog1Lin2
 - interpolate, [67](#)
 - IntPolLog1Lin2, [67](#)
- Ah::InvAmp, [68](#)
- Ah::InvAmp
 - ~InvAmp, [68](#)
 - getA, [69](#)
 - InvAmp, [68](#)
- Ah::Kef_B110_A, [70](#)
 - ~Kef_B110_A, [70](#)
 - Kef_B110_A, [70](#)
- Ah::Kef_B110_B, [71](#)
 - ~Kef_B110_B, [71](#)
 - Kef_B110_B, [71](#)
- Ah::Kef_B139_B, [72](#)
 - ~Kef_B139_B, [72](#)
 - Kef_B139_B, [72](#)
- Ah::Kef_B200_A, [73](#)
 - ~Kef_B200_A, [73](#)
 - Kef_B200_A, [73](#)
- Ah::Kef_B200_B, [74](#)
 - ~Kef_B200_B, [74](#)
 - Kef_B200_B, [74](#)
- Ah::Kef_B300_B, [75](#)
 - ~Kef_B300_B, [75](#)
 - Kef_B300_B, [75](#)
- Ah::LinTechOpAmpCreator, [76](#)
- Ah::LinTechOpAmpCreator
 - create, [76](#)
- Ah::LM833, [77](#)
 - LM833, [77](#)
- Ah::LT1028, [78](#)
 - LT1028, [78](#)
- Ah::LT1115, [79](#)
 - LT1115, [79](#)
- Ah::MathConst, [80](#)
- Ah::MathConst
 - Pi, [80](#)
- Ah::MiscOpAmpCreator, [81](#)
- Ah::MiscOpAmpCreator
 - create, [81](#)
- Ah::Monacor_SP45_4, [82](#)
 - ~Monacor_SP45_4, [82](#)
 - Monacor_SP45_4, [82](#)
- Ah::Monacor_SP45_8, [83](#)
 - ~Monacor_SP45_8, [83](#)
 - Monacor_SP45_8, [83](#)
- Ah::Monacor_SP60_4, [84](#)
 - ~Monacor_SP60_4, [84](#)
 - Monacor_SP60_4, [84](#)
- Ah::Monacor_SP60_8, [85](#)
 - ~Monacor_SP60_8, [85](#)
 - Monacor_SP60_8, [85](#)
- Ah::Monacor_SPP110_4, [86](#)
 - ~Monacor_SPP110_4, [86](#)
 - Monacor_SPP110_4, [86](#)
- Ah::Monacor_SPP110_8, [87](#)
 - ~Monacor_SPP110_8, [87](#)
 - getSpl, [87](#)
 - Monacor_SPP110_8, [87](#)
- Ah::NE5532, [89](#)
 - NE5532, [89](#)
- Ah::NoneInvAmp, [90](#)
- Ah::NoneInvAmp
 - ~NoneInvAmp, [91](#)
 - getA, [91](#)
 - getAd, [91](#)
 - NoneInvAmp, [91](#)
- Ah::OP07, [92](#)
 - OP07, [92](#)
- Ah::OP177, [93](#)
 - OP177, [93](#)
- Ah::OP27, [94](#)
 - OP27, [94](#)
- Ah::OP37, [95](#)
 - OP37, [95](#)
- Ah::OP77, [96](#)
 - OP77, [96](#)
- Ah::OPA134, [97](#)
 - OPA134, [97](#)
- Ah::OPA604, [98](#)
 - OPA604, [98](#)
- Ah::OpAmp, [99](#)
- Ah::OpAmp
 - ~OpAmp, [101](#)
 - A0, [102](#)
 - Af, [102](#)
 - Cin, [102](#)
 - En, [102](#)
 - f, [102](#)
 - Fg, [102](#)
 - FgEn, [102](#)
 - FgIn, [102](#)
 - getAd, [101](#)
 - getName, [101](#)
 - getZin, [101](#)
 - getZout, [101](#)
 - In, [102](#)
 - Name, [102](#)
 - OpAmp, [100](#), [101](#)
 - operator=, [101](#)
 - Rin, [102](#)
 - Rout, [102](#)
- Ah::OpAmpCreator, [103](#)

- Ah::OpAmpCreator
 - ~OpAmpCreator, [103](#)
 - create, [103](#)
 - OpAmpCreator, [103](#)
- Ah::PhysConst, [104](#)
- Ah::PhysConst
 - Boltzmann, [104](#)
 - eps0, [104](#)
 - t25, [104](#)
- Ah::PhysUnitCplx, [105](#)
- Ah::PhysUnitCplx
 - abs, [107](#)
 - arg, [107](#)
 - getVal, [107](#)
 - isInf, [107](#)
 - isNul, [107](#)
 - isValid, [107](#)
 - operator *, [108](#)
 - operator+, [108](#)
 - operator-, [108](#)
 - operator/, [108](#), [109](#)
 - operator<=, [109](#)
 - PhysUnitCplx, [107](#)
 - Unit, [109](#)
 - Value, [109](#)
- Ah::PhysUnitDbl, [110](#)
- Ah::PhysUnitDbl
 - ~PhysUnitDbl, [111](#)
 - abs, [111](#)
 - arg, [111](#)
 - getVal, [111](#)
 - operator<=, [111](#)
 - PhysUnitDbl, [111](#)
 - Unit, [111](#)
 - Value, [111](#)
- Ah::PickUp, [112](#)
- Ah::PickUp
 - ~PickUp, [113](#)
 - getAtt, [113](#)
 - getName, [113](#)
 - operator=, [113](#)
 - PickUp, [113](#)
- Ah::PickUpClearaudioAlphaWood, [115](#)
- Ah::PickUpClearaudioAlphaWood
 - ~PickUpClearaudioAlphaWood, [115](#)
 - PickUpClearaudioAlphaWood, [115](#)
- Ah::PickUpClearaudioDiscovery, [116](#)
- Ah::PickUpClearaudioDiscovery
 - ~PickUpClearaudioDiscovery, [116](#)
 - PickUpClearaudioDiscovery, [116](#)
- Ah::PickUpClearaudioGammaSGold, [117](#)
- Ah::PickUpClearaudioGammaSGold
 - ~PickUpClearaudioGammaSGold, [117](#)
 - PickUpClearaudioGammaSGold, [117](#)
- Ah::PickUpClearaudioVirtuosoWood, [118](#)
- Ah::PickUpClearaudioVirtuosoWood
 - ~PickUpClearaudioVirtuosoWood, [118](#)
 - PickUpClearaudioVirtuosoWood, [118](#)
- Ah::PickUpGradoMaster, [119](#)
- Ah::PickUpGradoMaster
 - ~PickUpGradoMaster, [119](#)
 - PickUpGradoMaster, [119](#)
- Ah::PickUpGradoPlatinum, [120](#)
- Ah::PickUpGradoPlatinum
 - ~PickUpGradoPlatinum, [120](#)
 - PickUpGradoPlatinum, [120](#)
- Ah::PickUpGradoReference, [121](#)
- Ah::PickUpGradoReference
 - ~PickUpGradoReference, [121](#)
 - PickUpGradoReference, [121](#)
- Ah::PickUpGradoStatement, [122](#)
- Ah::PickUpGradoStatement
 - ~PickUpGradoStatement, [122](#)
 - PickUpGradoStatement, [122](#)
- Ah::PickUpOrtofon540mk2, [123](#)
- Ah::PickUpOrtofon540mk2
 - ~PickUpOrtofon540mk2, [123](#)
 - PickUpOrtofon540mk2, [123](#)
- Ah::PickUpVdhBlackBeautySPX, [124](#)
- Ah::PickUpVdhBlackBeautySPX
 - ~PickUpVdhBlackBeautySPX, [124](#)
 - PickUpVdhBlackBeautySPX, [124](#)
- Ah::PickUpVdhColibriXGP, [125](#)
- Ah::PickUpVdhColibriXGP
 - ~PickUpVdhColibriXGP, [125](#)
 - PickUpVdhColibriXGP, [125](#)
- Ah::PickUpVdhDDT2Special, [126](#)
- Ah::PickUpVdhDDT2Special
 - ~PickUpVdhDDT2Special, [126](#)
 - PickUpVdhDDT2Special, [126](#)
- Ah::PickUpVdhFrogCopper, [127](#)
- Ah::PickUpVdhFrogCopper
 - ~PickUpVdhFrogCopper, [127](#)
 - PickUpVdhFrogCopper, [127](#)
- Ah::PickUpVdhMc1Special, [128](#)
- Ah::PickUpVdhMc1Special
 - ~PickUpVdhMc1Special, [128](#)
 - PickUpVdhMc1Special, [128](#)
- Ah::PickUpVdhMm1, [129](#)
- Ah::PickUpVdhMm1
 - ~PickUpVdhMm1, [129](#)
 - PickUpVdhMm1, [129](#)
- Ah::Power, [130](#)
 - operator *, [131](#)
 - operator!=, [131](#)
 - operator+, [131](#)
 - operator-, [131](#)
 - operator/, [132](#)

- operator<, 132
- operator==, 132
- operator>, 132
- Power, 131
- Ah::Resistor, 133
 - getRes, 134
 - operator *, 134
 - operator+, 134
 - operator/, 134
 - operator<=, 134
 - operator | |, 134
 - Resistor, 134
- Ah::SalKey, 136
- Ah::SalKey
 - ~SalKey, 136
 - getA, 137
 - SalKey, 136
- Ah::Speaker, 138
 - ~Speaker, 140
 - Fr, 141
 - getFres, 140
 - getImp, 140
 - getName, 140
 - getQe, 140
 - getQm, 140
 - getQt, 140
 - getSpl, 140
 - getVa, 140
 - Ls, 141
 - operator=, 140
 - Qe, 141
 - Qm, 141
 - Qt, 141
 - Rdc, 141
 - Speaker, 139, 140
 - Spl, 141
 - Va, 141
- Ah::SpeakerCreator, 142
- Ah::SpeakerCreator
 - ~SpeakerCreator, 142
 - create, 142
 - SpeakerCreator, 142
- Ah::Transistor, 143
 - ~Transistor, 143
 - getIout, 143
 - getS, 143, 144
- Ah::TransistorFollower, 145
- Ah::TransistorFollower
 - ~TransistorFollower, 145
 - getIload, 145
 - getUload, 145
 - TransistorFollower, 145
- Ah::VierPol, 147
- Ah::VierPol
 - ~VierPol, 148
 - c11, 148
 - c12, 148
 - c21, 148
 - c22, 148
 - det, 148
 - equal, 148
 - isValid, 148
 - operator<=, 148
 - VierPol, 148
- Ah::VierPolA, 149
- Ah::VierPolA
 - ~VierPolA, 150
 - a11, 150
 - a12, 150
 - a21, 151
 - a22, 151
 - operator!=, 151
 - operator=, 151
 - operator==, 151
 - VierPolA, 150
- Ah::VierPolH, 152
- Ah::VierPolH
 - ~VierPolH, 153
 - h11, 153
 - h12, 153
 - h21, 154
 - h22, 154
 - operator!=, 154
 - operator=, 154
 - operator==, 154
 - VierPolH, 153
- Ah::VierPolY, 155
- Ah::VierPolY
 - ~VierPolY, 156
 - operator!=, 157
 - operator=, 156
 - operator==, 157
 - VierPolY, 156
 - y11, 157
 - y12, 157
 - y21, 157
 - y22, 157
- Ah::VierPolZ, 158
- Ah::VierPolZ
 - ~VierPolZ, 159
 - operator!=, 160
 - operator=, 159
 - operator==, 160
 - VierPolZ, 159
 - z11, 160
 - z12, 160
 - z21, 160
 - z22, 160

- Ah::Visaton_SC_10, [161](#)
 - ~Visaton_SC_10, [161](#)
 - getSpl, [161](#)
 - Visaton_SC_10, [161](#)
- Ah::Voltage, [163](#)
 - operator *, [164](#)
 - operator !=, [164](#)
 - operator +, [164](#)
 - operator -, [164](#)
 - operator /, [164](#)
 - operator <, [165](#)
 - operator ==, [165](#)
 - operator >, [165](#)
 - Voltage, [164](#)
- AH_ETECH_VERSION
 - Version.h, [213](#)
- arg
 - Ah::PhysUnitCplx, [107](#)
 - Ah::PhysUnitDbl, [111](#)
- Boltzmann
 - Ah::PhysConst, [104](#)
- Box
 - Ah::Box, [34](#)
- BPT
 - Ah::BPT, [37](#)
- c11
 - Ah::VierPol, [148](#)
- c12
 - Ah::VierPol, [148](#)
- c21
 - Ah::VierPol, [148](#)
- c22
 - Ah::VierPol, [148](#)
- calc
 - Ah::ClosedBox, [43](#)
- calcLowpass
 - Ah::Filter2PoleInv, [54](#)
- calcVal
 - Ah::Capacitor, [40](#)
 - Ah::FreqImpedance, [56](#)
 - Ah::Inductor, [61](#)
- Capacitor
 - Ah::Capacitor, [40](#)
- CdDeEmphasis
 - Ah, [24](#)
- CdPreEmphasis
 - Ah, [24](#)
- Cin
 - Ah::OpAmp, [102](#)
- ClosedBox
 - Ah::ClosedBox, [42](#)
- create
 - Ah::AnalogDevOpAmpCreator, [33](#)
 - Ah::BurrBrownOpAmpCreator, [38](#)
 - Ah::LinTechOpAmpCreator, [76](#)
 - Ah::MiscOpAmpCreator, [81](#)
 - Ah::OpAmpCreator, [103](#)
 - Ah::SpeakerCreator, [142](#)
- Current
 - Ah::Current, [47](#)
- damped
 - Ah::ClosedBox, [43](#)
- Dbl
 - Ah, [24](#)
- det
 - Ah::VierPol, [148](#)
- Divider
 - Ah::Divider, [49](#)
- double_complex
 - Ah, [24](#)
- En
 - Ah::OpAmp, [102](#)
- eps0
 - Ah::PhysConst, [104](#)
- equal
 - Ah::VierPol, [148](#)
- f
 - Ah::OpAmp, [102](#)
- FET
 - Ah::FET, [51](#)
- Fg
 - Ah::OpAmp, [102](#)
- FgEn
 - Ah::OpAmp, [102](#)
- FgIn
 - Ah::OpAmp, [102](#)
- Filter2PoleInv
 - Ah::Filter2PoleInv, [54](#)
- Fr
 - Ah::Speaker, [141](#)
- FreqImpedance
 - Ah::FreqImpedance, [56](#)
- getA
 - Ah::Amplifier, [31](#)
 - Ah::Filter2PoleInv, [54](#)
 - Ah::InvAmp, [69](#)
 - Ah::NoneInvAmp, [91](#)
 - Ah::SalKey, [137](#)
- getAd
 - Ah::NoneInvAmp, [91](#)
 - Ah::OpAmp, [101](#)
- getAtt

- Ah::Divider, 49
- Ah::PickUp, 113
- getCap
 - Ah::Capacitor, 40
- getFres
 - Ah::Speaker, 140
- getIload
 - Ah::TransistorFollower, 145
- getImp
 - Ah::Speaker, 140
- getInd
 - Ah::Inductor, 61
- getIout
 - Ah::BPT, 37
 - Ah::FET, 51
 - Ah::Transistor, 143
- getName
 - Ah::Box, 35
 - Ah::OpAmp, 101
 - Ah::PickUp, 113
 - Ah::Speaker, 140
- getQe
 - Ah::Speaker, 140
- getQm
 - Ah::Speaker, 140
- getQt
 - Ah::ClosedBox, 43
 - Ah::Speaker, 140
- getRes
 - Ah::Resistor, 134
- getS
 - Ah::BPT, 37
 - Ah::FET, 51
 - Ah::Transistor, 143, 144
- getSpeaker
 - Ah::Box, 35
- getSpl
 - Ah::Monacor_SPP110_8, 87
 - Ah::Speaker, 140
 - Ah::Visaton_SC_10, 161
- getUload
 - Ah::TransistorFollower, 145
- getUp
 - Ah::FET, 51
- getUt
 - Ah::BPT, 37
- getVa
 - Ah::Speaker, 140
- getVal
 - Ah::PhysUnitCplx, 107
 - Ah::PhysUnitDbl, 111
- getY
 - Ah::IntPol, 64
- getZin
 - Ah::Divider, 49
 - Ah::OpAmp, 101
- getZout
 - Ah::OpAmp, 101
- h11
 - Ah::VierPolH, 153
- h12
 - Ah::VierPolH, 153
- h21
 - Ah::VierPolH, 154
- h22
 - Ah::VierPolH, 154
- Ics
 - Ah::BPT, 37
- Ids
 - Ah::FET, 52
- Impedance
 - Ah::Impedance, 58
- In
 - Ah::OpAmp, 102
- inDeg
 - Ah, 24
- Inductor
 - Ah::Inductor, 61
- inPdB
 - Ah, 24
- interpolate
 - Ah::IntPol, 64
 - Ah::IntPolLin1Lin2, 66
 - Ah::IntPolLog1Lin2, 67
- IntPol
 - Ah::IntPol, 63
- IntPolLin1Lin2
 - Ah::IntPolLin1Lin2, 66
- IntPolLog1Lin2
 - Ah::IntPolLog1Lin2, 67
- InvAmp
 - Ah::InvAmp, 68
- inVdB
 - Ah, 24
- isInf
 - Ah::PhysUnitCplx, 107
- isNul
 - Ah::PhysUnitCplx, 107
- isValid
 - Ah::PhysUnitCplx, 107
 - Ah::VierPol, 148
- Kef_B110_A
 - Ah::Kef_B110_A, 70
- Kef_B110_B
 - Ah::Kef_B110_B, 71

- Kef_B139_B
 - Ah::Kef_B139_B, [72](#)
- Kef_B200_A
 - Ah::Kef_B200_A, [73](#)
- Kef_B200_B
 - Ah::Kef_B200_B, [74](#)
- Kef_B300_B
 - Ah::Kef_B300_B, [75](#)
- LM833
 - Ah::LM833, [77](#)
- Ls
 - Ah::Speaker, [141](#)
- LT1028
 - Ah::LT1028, [78](#)
- LT1115
 - Ah::LT1115, [79](#)
- MathConst.h
 - MathConst_PI, [192](#)
- MathConst_PI
 - MathConst.h, [192](#)
- Monacor_SP45_4
 - Ah::Monacor_SP45_4, [82](#)
- Monacor_SP45_8
 - Ah::Monacor_SP45_8, [83](#)
- Monacor_SP60_4
 - Ah::Monacor_SP60_4, [84](#)
- Monacor_SP60_8
 - Ah::Monacor_SP60_8, [85](#)
- Monacor_SPP110_4
 - Ah::Monacor_SPP110_4, [86](#)
- Monacor_SPP110_8
 - Ah::Monacor_SPP110_8, [87](#)
- Name
 - Ah::OpAmp, [102](#)
- NE5532
 - Ah::NE5532, [89](#)
- NoneInvAmp
 - Ah::NoneInvAmp, [91](#)
- OP07
 - Ah::OP07, [92](#)
- OP177
 - Ah::OP177, [93](#)
- OP27
 - Ah::OP27, [94](#)
- OP37
 - Ah::OP37, [95](#)
- OP77
 - Ah::OP77, [96](#)
- OPA134
 - Ah::OPA134, [97](#)
- OPA604
 - Ah::OPA604, [98](#)
- OpAmp
 - Ah::OpAmp, [100](#), [101](#)
- OpAmpCreator
 - Ah::OpAmpCreator, [103](#)
- operator *
 - Ah, [24](#), [25](#)
 - Ah::Capacitor, [40](#)
 - Ah::Current, [47](#)
 - Ah::Impedance, [58](#), [59](#)
 - Ah::Inductor, [61](#)
 - Ah::PhysUnitCplx, [108](#)
 - Ah::Power, [131](#)
 - Ah::Resistor, [134](#)
 - Ah::Voltage, [164](#)
- operator!=
 - Ah::Current, [47](#)
 - Ah::Impedance, [59](#)
 - Ah::Power, [131](#)
 - Ah::VierPolA, [151](#)
 - Ah::VierPolH, [154](#)
 - Ah::VierPolY, [157](#)
 - Ah::VierPolZ, [160](#)
 - Ah::Voltage, [164](#)
- operator()
 - Ah::FreqImpedance, [56](#)
- operator+
 - Ah::Capacitor, [40](#)
 - Ah::Current, [47](#)
 - Ah::Impedance, [59](#)
 - Ah::Inductor, [61](#)
 - Ah::PhysUnitCplx, [108](#)
 - Ah::Power, [131](#)
 - Ah::Resistor, [134](#)
 - Ah::Voltage, [164](#)
- operator-
 - Ah::Current, [47](#)
 - Ah::Impedance, [59](#)
 - Ah::PhysUnitCplx, [108](#)
 - Ah::Power, [131](#)
 - Ah::Voltage, [164](#)
- operator/
 - Ah, [25](#)
 - Ah::Capacitor, [40](#)
 - Ah::Current, [47](#)
 - Ah::Impedance, [59](#)
 - Ah::Inductor, [61](#)
 - Ah::PhysUnitCplx, [108](#), [109](#)
 - Ah::Power, [132](#)
 - Ah::Resistor, [134](#)
 - Ah::Voltage, [164](#)
- operator<
 - Ah::Current, [48](#)

- Ah::Impedance, [59](#)
- Ah::Power, [132](#)
- Ah::Voltage, [165](#)
- operator<<
 - Ah::Capacitor, [41](#)
 - Ah::Inductor, [62](#)
 - Ah::PhysUnitCplx, [109](#)
 - Ah::PhysUnitDbl, [111](#)
 - Ah::Resistor, [134](#)
 - Ah::VierPol, [148](#)
- operator=
 - Ah::OpAmp, [101](#)
 - Ah::PickUp, [113](#)
 - Ah::Speaker, [140](#)
 - Ah::VierPolA, [151](#)
 - Ah::VierPolH, [154](#)
 - Ah::VierPolY, [156](#)
 - Ah::VierPolZ, [159](#)
- operator==
 - Ah::Current, [48](#)
 - Ah::Impedance, [59](#)
 - Ah::Power, [132](#)
 - Ah::VierPolA, [151](#)
 - Ah::VierPolH, [154](#)
 - Ah::VierPolY, [157](#)
 - Ah::VierPolZ, [160](#)
 - Ah::Voltage, [165](#)
- operator>
 - Ah::Current, [48](#)
 - Ah::Impedance, [59](#)
 - Ah::Power, [132](#)
 - Ah::Voltage, [165](#)
- operator | |
 - Ah::Capacitor, [41](#)
 - Ah::Impedance, [59](#)
 - Ah::Inductor, [62](#)
 - Ah::Resistor, [134](#)
- PhysConst.h
 - PhysConst_BOLTZMANN, [199](#)
 - PhysConst_EPS0, [199](#)
 - PhysConst_T25, [199](#)
- PhysConst_BOLTZMANN
 - PhysConst.h, [199](#)
- PhysConst_EPS0
 - PhysConst.h, [199](#)
- PhysConst_T25
 - PhysConst.h, [199](#)
- PhysUnitCplx
 - Ah::PhysUnitCplx, [107](#)
- PhysUnitDbl
 - Ah::PhysUnitDbl, [111](#)
- Pi
 - Ah::MathConst, [80](#)
- PickUp
 - Ah::PickUp, [113](#)
- PickUpClearaudioAlphaWood
 - Ah::PickUpClearaudioAlphaWood, [115](#)
- PickUpClearaudioDiscovery
 - Ah::PickUpClearaudioDiscovery, [116](#)
- PickUpClearaudioGammaSGold
 - Ah::PickUpClearaudioGammaSGold, [117](#)
- PickUpClearaudioVirtuosoWood
 - Ah::PickUpClearaudioVirtuosoWood, [118](#)
- PickUpGradoMaster
 - Ah::PickUpGradoMaster, [119](#)
- PickUpGradoPlatinum
 - Ah::PickUpGradoPlatinum, [120](#)
- PickUpGradoReference
 - Ah::PickUpGradoReference, [121](#)
- PickUpGradoStatement
 - Ah::PickUpGradoStatement, [122](#)
- PickUpOrtofon540mk2
 - Ah::PickUpOrtofon540mk2, [123](#)
- PickUpVdhBlackBeautySPX
 - Ah::PickUpVdhBlackBeautySPX, [124](#)
- PickUpVdhColibriXGP
 - Ah::PickUpVdhColibriXGP, [125](#)
- PickUpVdhDDT2Special
 - Ah::PickUpVdhDDT2Special, [126](#)
- PickUpVdhFrogCopper
 - Ah::PickUpVdhFrogCopper, [127](#)
- PickUpVdhMc1Special
 - Ah::PickUpVdhMc1Special, [128](#)
- PickUpVdhMm1
 - Ah::PickUpVdhMm1, [129](#)
- Power
 - Ah::Power, [131](#)
- Qe
 - Ah::Speaker, [141](#)
- Qm
 - Ah::Speaker, [141](#)
- Qt
 - Ah::ClosedBox, [43](#)
 - Ah::Speaker, [141](#)
- Rdc
 - Ah::Speaker, [141](#)
- Resistor
 - Ah::Resistor, [134](#)
- RiaaDeEmphasis
 - Ah, [25](#)
- RiaaPreEmphasis
 - Ah, [25](#)
- Rin
 - Ah::OpAmp, [102](#)
- Rout

- Ah::OpAmp, [102](#)
- SalKey
 - Ah::SalKey, [136](#)
- sp
 - Ah::Box, [35](#)
- Speaker
 - Ah::Speaker, [139](#), [140](#)
- SpeakerCreator
 - Ah::SpeakerCreator, [142](#)
- Spl
 - Ah::Speaker, [141](#)
- t25
 - Ah::PhysConst, [104](#)
- table
 - Ah::IntPol, [64](#)
- TransistorFollower
 - Ah::TransistorFollower, [145](#)
- Unit
 - Ah::PhysUnitCplx, [109](#)
 - Ah::PhysUnitDbl, [111](#)
- Up25
 - Ah::FET, [52](#)
- Va
 - Ah::Speaker, [141](#)
- Value
 - Ah::PhysUnitCplx, [109](#)
 - Ah::PhysUnitDbl, [111](#)
- Version.h
 - AH_ETECH_VERSION, [213](#)
- VierPol
 - Ah::VierPol, [148](#)
- VierPolA
 - Ah::VierPolA, [150](#)
- VierPolH
 - Ah::VierPolH, [153](#)
- VierPolY
 - Ah::VierPolY, [156](#)
- VierPolZ
 - Ah::VierPolZ, [159](#)
- Visaton_SC_10
 - Ah::Visaton_SC_10, [161](#)
- Voltage
 - Ah::Voltage, [164](#)
- x
 - Ah::IntPolData, [65](#)
- y
 - Ah::IntPolData, [65](#)
- y11
 - Ah::VierPolY, [157](#)
- y12
 - Ah::VierPolY, [157](#)
- y21
 - Ah::VierPolY, [157](#)
- y22
 - Ah::VierPolY, [157](#)
- z11
 - Ah::VierPolZ, [160](#)
- z12
 - Ah::VierPolZ, [160](#)
- z21
 - Ah::VierPolZ, [160](#)
- z22
 - Ah::VierPolZ, [160](#)