

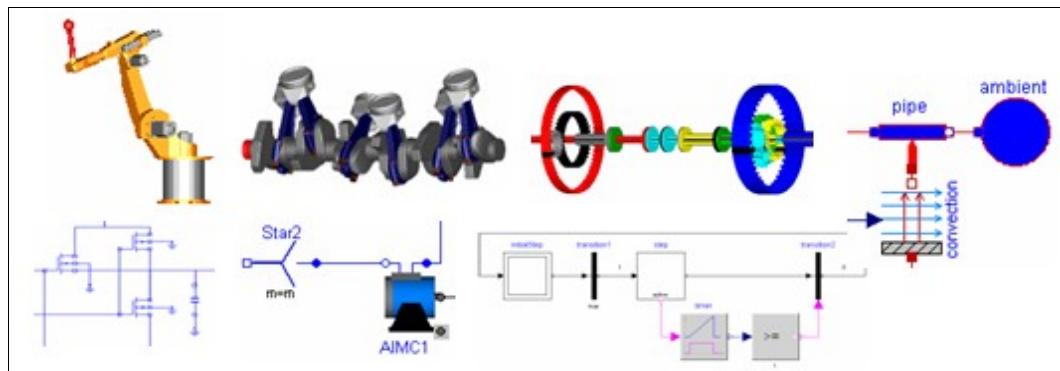


# Modelica Standard Library

Version 2.2.2

August 2007

Tutorial and Reference



Modelica Association  
<http://www.Modelica.org>

Copyright © 1998-2007, Modelica Association (<http://www.Modelica.org>)

Modelica® is a registered trademark of the Modelica Association

This manual can be freely distributed according to the Modelica License (which holds for the source code of the Modelica Standard Library, including its documentation):

Redistribution and use in source and binary forms, with or without modification are permitted, provided that the following conditions are met:

1. The author and copyright notices in the source files, these license conditions and the disclaimer below are (a) retained and (b) reproduced in the documentation provided with the distribution.
2. Modifications of the original source files are allowed, provided that a prominent notice is inserted in each changed file and the accompanying documentation, stating how and when the file was modified, and provided that the conditions under (1) are met.
3. It is not allowed to charge a fee for the original version or a modified version of the software, besides a reasonable fee for distribution and support. Distribution in aggregate with other (possibly commercial) programs as part of a larger (possibly commercial) software distribution is permitted, provided that it is not advertised as a product of your own.

**Disclaimer:**

The software (sources, binaries, etc.) in their original or in a modified form are provided "as is" and the copyright holders assume no responsibility for its contents what so ever. Any express or implied warranties, including, but not limited to, the implied warranties of merchantability and fitness for a particular purpose are disclaimed. In no event shall the copyright holders, or any party who modify and/or redistribute the package, be liable for any direct, indirect, incidental, special, exemplary, or consequential damages, arising in any way out of the use of this software, even if advised of the possibility of such damage.

Modelica is a freely available, object-oriented language for modeling of large, complex, and heterogeneous physical systems. It is suited for multi-domain modeling, for example, mechatronic models in robotics, automotive and aerospace applications involving mechanical, electrical, hydraulic and control subsystems, process oriented applications and generation and distribution of electric power. Models in Modelica are mathematically described by differential, algebraic and discrete equations. No particular variable needs to be solved for manually. A Modelica tool will have enough information to decide that automatically. Modelica is designed such that available, specialized algorithms can be utilized to enable efficient handling of large models having more than one hundred thousand equations. Modelica is suited and used for hardware-in-the-loop simulations and for embedded control systems.

Modelica is developed by the Modelica Association, a non-profit organization with seat in Linköping, Sweden. The Modelica Association also develops the free Modelica Standard Library containing model components in many domains that are based on standardized interface definitions. This Manual was automatically produced from the documentation included in the Modelica Standard Library. The source code of the Modelica Standard Library, as well as this manual, is available at  
<http://www.modelica.org/libraries/Modelica>

## Contents

Modelica.....	71
Modelica.UsersGuide.....	72
Modelica.UsersGuide.Overview.....	72
Modelica.UsersGuide.Connectors.....	74
Modelica.UsersGuide.Conventions.....	75
Modelica.UsersGuide.ReleaseNotes.....	77
Modelica.UsersGuide.Contact.....	78
Modelica.UsersGuide.ModelicaLicense.....	79
Modelica.Blocks.....	80
Modelica.Blocks.Examples.....	80
Modelica.Blocks.Examples.PID_Controller.....	81
Modelica.Blocks.Examples>ShowLogicalSources.....	82
Modelica.Blocks.Examples.LogicalNetwork1.....	82
Modelica.Blocks.Examples.BusUsage.....	82
Modelica.Blocks.Examples.BusUsage_Utility.....	84
Modelica.Blocks.Examples.BusUsage_Utility.Interfaces.....	84
Modelica.Blocks.Examples.BusUsage_Utility.Interfaces.ControlBus.....	84
Modelica.Blocks.Examples.BusUsage_Utility.Interfaces.SubControlBus.....	84
Modelica.Blocks.Examples.BusUsage_Utility.Part.....	85
Modelica.Blocks.Continuous.....	85
Modelica.Blocks.Continuous.Integrator.....	86
Modelica.Blocks.Continuous.LimIntegrator.....	87
Modelica.Blocks.Continuous.Derivative.....	87
Modelica.Blocks.Continuous.FirstOrder.....	88
Modelica.Blocks.Continuous.SecondOrder.....	89
Modelica.Blocks.Continuous.PI.....	90
Modelica.Blocks.Continuous.PID.....	90
Modelica.Blocks.Continuous.LimPID.....	91
Modelica.Blocks.Continuous.TransferFunction.....	93
Modelica.Blocks.Continuous.StateSpace.....	94
Modelica.Blocks.Continuous.Der.....	95
Modelica.Blocks.Continuous.LowpassButterworth.....	96
Modelica.Blocks.Continuous.CriticalDamping.....	96
Modelica.Blocks.Discrete.....	97
Modelica.Blocks.Discrete.Sampler.....	97
Modelica.Blocks.Discrete.ZeroOrderHold.....	98
Modelica.Blocks.Discrete.FirstOrderHold.....	98
Modelica.Blocks.Discrete.UnitDelay.....	99
Modelica.Blocks.Discrete.TransferFunction.....	99
Modelica.Blocks.Discrete.StateSpace.....	100
Modelica.Blocks.Discrete.TriggeredSampler.....	101
Modelica.Blocks.Discrete.TriggeredMax.....	101
Modelica.Blocks.Interfaces.....	102
Modelica.Blocks.Interfaces.RealSignal.....	103
Modelica.Blocks.Interfaces.BooleanSignal.....	104
Modelica.Blocks.Interfaces.IntegerSignal.....	104
Modelica.Blocks.Interfaces.RealInput.....	104
Modelica.Blocks.Interfaces.RealOutput.....	104
Modelica.Blocks.Interfaces.BooleanInput.....	104
Modelica.Blocks.Interfaces.BooleanOutput.....	105
Modelica.Blocks.Interfaces.IntegerInput.....	105
Modelica.Blocks.Interfaces.IntegerOutput.....	105
Modelica.Blocks.Interfaces.BlockIcon.....	105

Modelica.Blocks.Interfaces.SO	105
Modelica.Blocks.Interfaces.MO	106
Modelica.Blocks.Interfaces.SISO	106
Modelica.Blocks.Interfaces.SI2SO	106
Modelica.Blocks.Interfaces.SIMO	106
Modelica.Blocks.Interfaces.MISO	107
Modelica.Blocks.Interfaces.MIMO	107
Modelica.Blocks.Interfaces.MIMOs	108
Modelica.Blocks.Interfaces.MI2MO	108
Modelica.Blocks.Interfaces.SignalSource	108
Modelica.Blocks.Interfaces.SVcontrol	109
Modelica.Blocks.Interfaces.MVcontrol	109
Modelica.Blocks.Interfaces.DiscreteBlockIcon	110
Modelica.Blocks.Interfaces.DiscreteBlock	110
Modelica.Blocks.Interfaces.DiscreteSISO	110
Modelica.Blocks.Interfaces.DiscreteMIMO	110
Modelica.Blocks.Interfaces.DiscreteMIMOs	111
Modelica.Blocks.Interfaces.SVdiscrete	111
Modelica.Blocks.Interfaces.MVdiscrete	112
Modelica.Blocks.Interfaces.BooleanBlockIcon	112
Modelica.Blocks.Interfaces.BooleanSISO	112
Modelica.Blocks.Interfaces.BooleanMIMOs	113
Modelica.Blocks.Interfaces.MI2BooleanMOS	113
Modelica.Blocks.Interfaces.SI2BooleanSO	114
Modelica.Blocks.Interfaces.BooleanSignalSource	114
Modelica.Blocks.Interfaces.IntegerBlockIcon	114
Modelica.Blocks.Interfaces.IntegerSO	114
Modelica.Blocks.Interfaces.IntegerMO	115
Modelica.Blocks.Interfaces.IntegerSignalSource	115
Modelica.Blocks.Interfaces.IntegerSIBooleanSO	115
Modelica.Blocks.Interfaces.IntegerMIBooleanMOS	116
Modelica.Blocks.Interfaces.partialBooleanBlockIcon	116
Modelica.Blocks.Interfaces.partialBooleanSISO	116
Modelica.Blocks.Interfaces.partialBooleanSI2SO	116
Modelica.Blocks.Interfaces.partialBooleanSI3SO	117
Modelica.Blocks.Interfaces.partialBooleanSI	117
Modelica.Blocks.Interfaces.partialBooleanSO	117
Modelica.Blocks.Interfaces.partialBooleanSource	118
Modelica.Blocks.Interfaces.partialBooleanThresholdComparison	118
Modelica.Blocks.Interfaces.partialBooleanComparison	118
Modelica.Blocks.Interfaces.Adaptors	119
Modelica.Blocks.Interfaces.Adaptors.SendReal	119
Modelica.Blocks.Interfaces.Adaptors.SendBoolean	120
Modelica.Blocks.Interfaces.Adaptors.SendInteger	120
Modelica.Blocks.Interfaces.Adaptors.ReceiveReal	120
Modelica.Blocks.Interfaces.Adaptors.ReceiveBoolean	121
Modelica.Blocks.Interfaces.Adaptors.ReceiveInteger	121
Modelica.Blocks.Interfaces.Adaptors.AdaptorReal	121
Modelica.Blocks.Interfaces.Adaptors.AdaptorBoolean	122
Modelica.Blocks.Interfaces.Adaptors.AdaptorInteger	122
Modelica.Blocks.Interfaces.PartialConversionBlock	122
Modelica.Blocks.Logical	122
Modelica.Blocks.Logical.And	124
Modelica.Blocks.Logical.Or	124
Modelica.Blocks.Logical.Xor	124
Modelica.Blocks.Logical.Nor	125
Modelica.Blocks.Logical.Nand	125
Modelica.Blocks.Logical.Not	125

Modelica.Blocks.Logical.Pre.....	126
Modelica.Blocks.Logical.Edge.....	126
Modelica.Blocks.Logical.FallingEdge.....	127
Modelica.Blocks.Logical.Change.....	127
Modelica.Blocks.Logical.GreaterThreshold.....	127
Modelica.Blocks.Logical.GreaterEqualThreshold.....	128
Modelica.Blocks.Logical.LessThreshold.....	128
Modelica.Blocks.Logical.LessEqualThreshold.....	128
Modelica.Blocks.Logical.Greater.....	129
Modelica.Blocks.Logical.GreaterEqual.....	129
Modelica.Blocks.Logical.Less.....	129
Modelica.Blocks.Logical.LessEqual.....	130
Modelica.Blocks.Logical.ZeroCrossing.....	130
Modelica.Blocks.Logical.LogicalSwitch.....	130
Modelica.Blocks.Logical.Switch.....	131
Modelica.Blocks.Logical.Hysteresis.....	131
Modelica.Blocks.Logical.OnOffController.....	132
Modelica.Blocks.Logical.TriggeredTrapezoid.....	132
Modelica.Blocks.Logical.Timer.....	133
Modelica.Blocks.Logical.TerminateSimulation.....	133
Modelica.Blocks.Math.....	133
Modelica.Blocks.Math.UnitConversions.....	135
Modelica.Blocks.Math.UnitConversions.ConvertAllUnits.....	136
Modelica.Blocks.Math.UnitConversions.To_degC.....	136
Modelica.Blocks.Math.UnitConversions.From_degC.....	136
Modelica.Blocks.Math.UnitConversions.To_degF.....	137
Modelica.Blocks.Math.UnitConversions.From_degF.....	137
Modelica.Blocks.Math.UnitConversions.To_degRk.....	137
Modelica.Blocks.Math.UnitConversions.From_degRk.....	137
Modelica.Blocks.Math.UnitConversions.To_deg.....	137
Modelica.Blocks.Math.UnitConversions.From_deg.....	137
Modelica.Blocks.Math.UnitConversions.To_rpm.....	138
Modelica.Blocks.Math.UnitConversions.From_rpm.....	138
Modelica.Blocks.Math.UnitConversions.To_kmh.....	138
Modelica.Blocks.Math.UnitConversions.From_kmh.....	138
Modelica.Blocks.Math.UnitConversions.To_day.....	138
Modelica.Blocks.Math.UnitConversions.From_day.....	138
Modelica.Blocks.Math.UnitConversions.To_hour.....	139
Modelica.Blocks.Math.UnitConversions.From_hour.....	139
Modelica.Blocks.Math.UnitConversions.To_minute.....	139
Modelica.Blocks.Math.UnitConversions.From_minute.....	139
Modelica.Blocks.Math.UnitConversions.To_litre.....	139
Modelica.Blocks.Math.UnitConversions.From_litre.....	139
Modelica.Blocks.Math.UnitConversions.To_kWh.....	140
Modelica.Blocks.Math.UnitConversions.From_kWh.....	140
Modelica.Blocks.Math.UnitConversions.To_bar.....	140
Modelica.Blocks.Math.UnitConversions.From_bar.....	140
Modelica.Blocks.Math.UnitConversions.To_gps.....	140
Modelica.Blocks.Math.UnitConversions.From_gps.....	140
Modelica.Blocks.Math.TwoInputs.....	141
Modelica.Blocks.Math.TwoOutputs.....	141
Modelica.Blocks.Math.Gain.....	141
Modelica.Blocks.Math.MatrixGain.....	142
Modelica.Blocks.Math.Sum.....	142
Modelica.Blocks.Math.Feedback.....	143
Modelica.Blocks.Math.Add.....	143
Modelica.Blocks.Math.Add3.....	144
Modelica.Blocks.Math.Product.....	144

Modelica.Blocks.Math.Division	145
Modelica.Blocks.Math.Abs	145
Modelica.Blocks.Math.Sign	145
Modelica.Blocks.Math.Sqrt	146
Modelica.Blocks.Math.Sin	146
Modelica.Blocks.Math.Cos	147
Modelica.Blocks.Math.Tan	147
Modelica.Blocks.Math.Asin	147
Modelica.Blocks.Math.Acos	148
Modelica.Blocks.Math.Atan	148
Modelica.Blocks.Math.Atan2	148
Modelica.Blocks.Math.Sinh	149
Modelica.Blocks.Math.Cosh	149
Modelica.Blocks.Math.Tanh	150
Modelica.Blocks.Math.Exp	150
Modelica.Blocks.Math.Log	150
Modelica.Blocks.Math.Log10	151
Modelica.Blocks.Math.RealToInteger	152
Modelica.Blocks.Math.IntegerToReal	152
Modelica.Blocks.Math.BooleanToReal	152
Modelica.Blocks.Math.BooleanToInteger	153
Modelica.Blocks.Math.RealToBoolean	153
Modelica.Blocks.Math.IntegerToBoolean	154
Modelica.Blocks.Math.Max	154
Modelica.Blocks.Math.Min	154
Modelica.Blocks.Math.Edge	155
Modelica.Blocks.Math.BooleanChange	155
Modelica.Blocks.Math.IntegerChange	155
Modelica.Blocks.Nonlinear	156
Modelica.Blocks.Nonlinear.Limiter	156
Modelica.Blocks.Nonlinear.VariableLimiter	157
Modelica.Blocks.Nonlinear.DeadZone	157
Modelica.Blocks.Nonlinear.FixedDelay	158
Modelica.Blocks.Nonlinear.PadeDelay	158
Modelica.Blocks.Nonlinear.VariableDelay	159
Modelica.Blocks.Routing	159
Modelica.Blocks.Routing.ExtractSignal	160
Modelica.Blocks.Routing.Extractor	161
Modelica.Blocks.Routing.Multiplex2	161
Modelica.Blocks.Routing.Multiplex3	162
Modelica.Blocks.Routing.Multiplex4	162
Modelica.Blocks.Routing.Multiplex5	163
Modelica.Blocks.Routing.Multiplex6	163
Modelica.Blocks.Routing.DeMultiplex2	164
Modelica.Blocks.Routing.DeMultiplex3	164
Modelica.Blocks.Routing.DeMultiplex4	165
Modelica.Blocks.Routing.DeMultiplex5	165
Modelica.Blocks.Routing.DeMultiplex6	166
Modelica.Blocks.Routing.RealPassThrough	166
Modelica.Blocks.Routing.IntegerPassThrough	167
Modelica.Blocks.Routing.BooleanPassThrough	167
Modelica.Blocks.Sources	167
Modelica.Blocks.Sources.RealExpression	168
Modelica.Blocks.Sources.IntegerExpression	169
Modelica.Blocks.Sources.BooleanExpression	169
Modelica.Blocks.Sources.Clock	170
Modelica.Blocks.Sources.Constant	170
Modelica.Blocks.Sources.Step	171

Modelica.Blocks.Sources.Ramp.....	172
Modelica.Blocks.Sources.Sine.....	172
Modelica.Blocks.Sources.ExpSine.....	173
Modelica.Blocks.Sources.Exponentials.....	174
Modelica.Blocks.Sources.Pulse.....	175
Modelica.Blocks.Sources.SawTooth.....	175
Modelica.Blocks.Sources.Trapezoid.....	176
Modelica.Blocks.Sources.KinematicPTP.....	176
Modelica.Blocks.Sources.KinematicPTP2.....	177
Modelica.Blocks.Sources.TimeTable.....	178
Modelica.Blocks.Sources.CombiTimeTable.....	179
Modelica.Blocks.Sources.BooleanConstant.....	181
Modelica.Blocks.Sources.BooleanStep.....	181
Modelica.Blocks.Sources.BooleanPulse.....	182
Modelica.Blocks.Sources.SampleTrigger.....	182
Modelica.Blocks.Sources.BooleanTable.....	183
Modelica.Blocks.Sources.IntegerConstant.....	183
Modelica.Blocks.Sources.IntegerStep.....	183
Modelica.Blocks.Tables.....	184
Modelica.Blocks.Tables.CombiTable1D.....	184
Modelica.Blocks.Tables.CombiTable1Ds.....	186
Modelica.Blocks.Tables.CombiTable2D.....	188
Modelica.Blocks.Types.....	189
Modelica.Blocks.Types.Extrapolation.....	190
Modelica.Blocks.Types.Extrapolation.Temp.....	190
Modelica.Blocks.Types.Init.....	191
Modelica.Blocks.Types.Init.Temp.....	191
Modelica.Blocks.Types.InitPID.....	192
Modelica.Blocks.Types.InitPID.Temp.....	193
Modelica.Blocks.Types.SimpleController.....	193
Modelica.Blocks.Types.SimpleController.Temp.....	193
Modelica.Blocks.Types.Smoothness.....	194
Modelica.Blocks.Types.Smoothness.Temp.....	194
Modelica.Blocks.Types.StateSelection.....	195
Modelica.Blocks.Types.StateSelection.Temp.....	195
Modelica.Constants.....	195
Modelica.Electrical.....	197
Modelica.Electrical.Analog.....	198
Modelica.Electrical.Analog.Examples.....	199
Modelica.Electrical.Analog.Examples.CauerLowPassAnalog.....	199
Modelica.Electrical.Analog.Examples.CauerLowPassOPV.....	200
Modelica.Electrical.Analog.Examples.CauerLowPassSC.....	200
Modelica.Electrical.Analog.Examples.CharacteristicIdealDiodes.....	201
Modelica.Electrical.Analog.Examples.CharacteristicThyristors.....	201
Modelica.Electrical.Analog.Examples.ChuaCircuit.....	202
Modelica.Electrical.Analog.Examples.DifferenceAmplifier.....	202
Modelica.Electrical.Analog.Examples.HeatingMOSInverter.....	202
Modelica.Electrical.Analog.Examples.HeatingNPN_OrGate.....	203
Modelica.Electrical.Analog.Examples.HeatingRectifier.....	203
Modelica.Electrical.Analog.Examples.NandGate.....	203
Modelica.Electrical.Analog.Examples.Rectifier.....	203
Modelica.Electrical.Analog.Examples.ShowSaturatingInductor.....	204
Modelica.Electrical.Analog.Examples.ShowVariableResistor.....	204
Modelica.Electrical.Analog.Examples.Utilities.....	205
Modelica.Electrical.Analog.Examples.Utilities.Nand.....	205
Modelica.Electrical.Analog.Examples.Utilities.NonlinearResistor.....	205
Modelica.Electrical.Analog.Examples.Utilities.RealSwitch.....	206
Modelica.Electrical.Analog.Examples.Utilities.Transistor.....	206

Modelica.Electrical.Analog.Basic.....	206
Modelica.Electrical.Analog.Basic.Ground.....	207
Modelica.Electrical.Analog.Basic.Resistor.....	207
Modelica.Electrical.Analog.Basic.HeatingResistor.....	208
Modelica.Electrical.Analog.Basic.Conductor.....	208
Modelica.Electrical.Analog.Basic.Capacitor.....	209
Modelica.Electrical.Analog.Basic.Inductor.....	209
Modelica.Electrical.Analog.Basic.SaturatingInductor.....	210
Modelica.Electrical.Analog.Basic.Transformer.....	210
Modelica.Electrical.Analog.Basic.Gyrator.....	211
Modelica.Electrical.Analog.Basic.EMF.....	211
Modelica.Electrical.Analog.Basic.VCV.....	212
Modelica.Electrical.Analog.Basic.VCC.....	212
Modelica.Electrical.Analog.Basic.CCV.....	213
Modelica.Electrical.Analog.Basic.CCC.....	213
Modelica.Electrical.Analog.Basic.OpAmp.....	214
Modelica.Electrical.Analog.Basic.VariableResistor.....	214
Modelica.Electrical.Analog.Basic.VariableConductor.....	215
Modelica.Electrical.Analog.Basic.VariableCapacitor.....	215
Modelica.Electrical.Analog.Basic.VariableInductor.....	216
Modelica.Electrical.Analog.Ideal.....	216
Modelica.Electrical.Analog.Ideal.IdealThyristor.....	217
Modelica.Electrical.Analog.Ideal.IdealGTOThyristor.....	218
Modelica.Electrical.Analog.Ideal.IdealCommutingSwitch.....	218
Modelica.Electrical.Analog.Ideal.IdealIntermediateSwitch.....	219
Modelica.Electrical.Analog.Ideal.ControlledIdealCommutingSwitch.....	220
Modelica.Electrical.Analog.Ideal.ControlledIdealIntermediateSwitch.....	220
Modelica.Electrical.Analog.Ideal.IdealOpAmp.....	221
Modelica.Electrical.Analog.Ideal.IdealOpAmp3Pin.....	221
Modelica.Electrical.Analog.Ideal.IdealOpAmpLimited.....	222
Modelica.Electrical.Analog.Ideal.IdealDiode.....	222
Modelica.Electrical.Analog.Ideal.IdealTransformer.....	223
Modelica.Electrical.Analog.Ideal.IdealGyrator.....	223
Modelica.Electrical.Analog.Ideal.Idle.....	224
Modelica.Electrical.Analog.Ideal.Short.....	224
Modelica.Electrical.Analog.Ideal.IdealOpeningSwitch.....	224
Modelica.Electrical.Analog.Ideal.IdealClosingSwitch.....	225
Modelica.Electrical.Analog.Ideal.ControlledIdealOpeningSwitch.....	225
Modelica.Electrical.Analog.Ideal.ControlledIdealClosingSwitch.....	226
Modelica.Electrical.Analog.Interfaces.....	226
Modelica.Electrical.Analog.Interfaces.Pin.....	227
Modelica.Electrical.Analog.Interfaces.PositivePin.....	227
Modelica.Electrical.Analog.Interfaces.NegativePin.....	227
Modelica.Electrical.Analog.Interfaces.TwoPin.....	228
Modelica.Electrical.Analog.Interfaces.OnePort.....	228
Modelica.Electrical.Analog.Interfaces.TwoPort.....	228
Modelica.Electrical.Analog.Interfaces.AbsoluteSensor.....	229
Modelica.Electrical.Analog.Interfaces.RelativeSensor.....	229
Modelica.Electrical.Analog.Interfaces.VoltageSource.....	229
Modelica.Electrical.Analog.Interfaces.CurrentSource.....	229
Modelica.Electrical.Analog.Lines.....	230
Modelica.Electrical.Analog.Lines.OLine.....	230
Modelica.Electrical.Analog.Lines.ULine.....	231
Modelica.Electrical.Analog.Lines.TLine1.....	231
Modelica.Electrical.Analog.Lines.TLine2.....	232
Modelica.Electrical.Analog.Lines.TLine3.....	233
Modelica.Electrical.Analog.Semiconductors.....	233
Modelica.Electrical.Analog.Semiconductors.Diode.....	234

Modelica.Electrical.Analog.Semiconductors.PMOS.....	235
Modelica.Electrical.Analog.Semiconductors.NMOS.....	236
Modelica.Electrical.Analog.Semiconductors.NPN.....	237
Modelica.Electrical.Analog.Semiconductors.PNP.....	238
Modelica.Electrical.Analog.Semiconductors.HeatingDiode.....	239
Modelica.Electrical.Analog.Semiconductors.HeatingNMOS.....	239
Modelica.Electrical.Analog.Semiconductors.HeatingPMOS.....	241
Modelica.Electrical.Analog.Semiconductors.HeatingNPN.....	242
Modelica.Electrical.Analog.Semiconductors.HeatingPNP.....	243
Modelica.Electrical.Analog.Sensors.....	244
Modelica.Electrical.Analog.Sensors.PotentialSensor.....	245
Modelica.Electrical.Analog.Sensors.VoltageSensor.....	245
Modelica.Electrical.Analog.Sensors.CurrentSensor.....	245
Modelica.Electrical.Analog.Sensors.PowerSensor.....	245
Modelica.Electrical.Analog.Sources.....	246
Modelica.Electrical.Analog.Sources.SignalVoltage.....	246
Modelica.Electrical.Analog.Sources.ConstantVoltage.....	247
Modelica.Electrical.Analog.Sources.StepVoltage.....	247
Modelica.Electrical.Analog.Sources.RampVoltage.....	247
Modelica.Electrical.Analog.Sources.SineVoltage.....	248
Modelica.Electrical.Analog.Sources.ExpSineVoltage.....	248
Modelica.Electrical.Analog.Sources.ExponentialsVoltage.....	249
Modelica.Electrical.Analog.Sources.PulseVoltage.....	249
Modelica.Electrical.Analog.Sources.SawToothVoltage.....	249
Modelica.Electrical.Analog.Sources.TrapezoidVoltage.....	250
Modelica.Electrical.Analog.Sources.TableVoltage.....	250
Modelica.Electrical.Analog.Sources.SignalCurrent.....	251
Modelica.Electrical.Analog.Sources.ConstantCurrent.....	251
Modelica.Electrical.Analog.Sources.StepCurrent.....	252
Modelica.Electrical.Analog.Sources.RampCurrent.....	252
Modelica.Electrical.Analog.Sources.SineCurrent.....	252
Modelica.Electrical.Analog.Sources.ExpSineCurrent.....	253
Modelica.Electrical.Analog.Sources.ExponentialsCurrent.....	253
Modelica.Electrical.Analog.Sources.PulseCurrent.....	254
Modelica.Electrical.Analog.Sources.SawToothCurrent.....	254
Modelica.Electrical.Analog.Sources.TrapezoidCurrent.....	254
Modelica.Electrical.Analog.Sources.TableCurrent.....	255
Modelica.Electrical.Digital.....	256
Modelica.Electrical.Digital.UsersGuide.....	257
Modelica.Electrical.Digital.Examples.....	257
Modelica.Electrical.Digital.Examples.Multiplexer.....	258
Modelica.Electrical.Digital.Examples.FlipFlop.....	258
Modelica.Electrical.Digital.Examples.HalfAdder.....	258
Modelica.Electrical.Digital.Examples.FullAdder.....	259
Modelica.Electrical.Digital.Examples.Adder4.....	259
Modelica.Electrical.Digital.Examples.Counter3.....	260
Modelica.Electrical.Digital.Examples.Counter.....	260
Modelica.Electrical.Digital.Examples.Utilities.....	260
Modelica.Electrical.Digital.Examples.Utilities.MUX4.....	261
Modelica.Electrical.Digital.Examples.Utilities.RS.....	261
Modelica.Electrical.Digital.Examples.Utilities.RSFF.....	262
Modelica.Electrical.Digital.Examples.Utilities.DFF.....	262
Modelica.Electrical.Digital.Examples.Utilities.JKFF.....	263
Modelica.Electrical.Digital.Examples.Utilities.HalfAdder.....	263
Modelica.Electrical.Digital.Examples.Utilities.FullAdder.....	263
Modelica.Electrical.Digital.Examples.Utilities.Adder.....	264
Modelica.Electrical.Digital.Examples.Utilities.Counter3.....	264
Modelica.Electrical.Digital.Examples.Utilities.Counter.....	265

---

Modelica.Electrical.Digital.Interfaces.....	265
Modelica.Electrical.Digital.Interfaces.Logic.....	266
Modelica.Electrical.Digital.Interfaces.LogicValue.....	266
Modelica.Electrical.Digital.Interfaces.DigitalSignal.....	266
Modelica.Electrical.Digital.Interfaces.DigitalInput.....	267
Modelica.Electrical.Digital.Interfaces.DigitalOutput.....	267
Modelica.Electrical.Digital.Interfaces.SISO.....	267
Modelica.Electrical.Digital.Interfaces.MISO.....	267
Modelica.Electrical.Digital.Tables.....	267
Modelica.Electrical.Digital.Delay.....	269
Modelica.Electrical.Digital.Delay.DelayParams.....	270
Modelica.Electrical.Delay.TransportDelay.....	270
Modelica.Electrical.Delay.InertialDelay.....	270
Modelica.Electrical.Delay.InertialDelaySensitive.....	271
Modelica.Electrical.Digital.Basic.....	271
Modelica.Electrical.Digital.Basic.Not.....	272
Modelica.Electrical.Digital.Basic.And.....	272
Modelica.Electrical.Digital.Basic.Nand.....	272
Modelica.Electrical.Digital.Basic.Or.....	273
Modelica.Electrical.Digital.Basic.Nor.....	273
Modelica.Electrical.Digital.Basic.Xor.....	273
Modelica.Electrical.Digital.Basic.Xnor.....	274
Modelica.Electrical.Digital.Gates.....	274
Modelica.Electrical.Digital.Gates.InvGate.....	274
Modelica.Electrical.Digital.Gates.AndGate.....	275
Modelica.Electrical.Digital.Gates.NandGate.....	275
Modelica.Electrical.Digital.Gates.OrGate.....	276
Modelica.Electrical.Digital.Gates.NorGate.....	276
Modelica.Electrical.Digital.Gates.XorGate.....	277
Modelica.Electrical.Digital.Gates.XnorGate.....	277
Modelica.Electrical.Digital.Gates.BufGate.....	277
Modelica.Electrical.Digital.Sources.....	278
Modelica.Electrical.Digital.Sources.Set.....	278
Modelica.Electrical.Digital.Sources.Step.....	279
Modelica.Electrical.Digital.Sources.Table.....	280
Modelica.Electrical.Digital.Sources.Pulse.....	280
Modelica.Electrical.Digital.Sources.Clock.....	281
Modelica.Electrical.Digital.Converters.....	282
Modelica.Electrical.Digital.Converters.LogicToXO1.....	282
Modelica.Electrical.Digital.Converters.LogicToXO1Z.....	283
Modelica.Electrical.Digital.Converters.LogicToUX01.....	283
Modelica.Electrical.Digital.Converters.BooleanToLogic.....	284
Modelica.Electrical.Digital.Converters.LogicToBoolean.....	284
Modelica.Electrical.Digital.Converters.RealToLogic.....	285
Modelica.Electrical.Digital.Converters.LogicToReal.....	286
Modelica.Electrical.Machines.....	286
Modelica.Electrical.Machines.Examples.....	287
Modelica.Electrical.Machines.Examples.AIMC_DOL.....	288
Modelica.Electrical.Machines.Examples.AIMC_YD.....	289
Modelica.Electrical.Machines.Examples.AIMS_start.....	289
Modelica.Electrical.Machines.Examples.AIMC_Inverter.....	290
Modelica.Electrical.Machines.Examples.SMR_Inverter.....	290
Modelica.Electrical.Machines.Examples.SMPM_Inverter.....	291
Modelica.Electrical.Machines.Examples.SMEE_Gen.....	291
Modelica.Electrical.Machines.Examples.DCPM_start.....	292
Modelica.Electrical.Machines.Examples.DCEE_start.....	293
Modelica.Electrical.Machines.Examples.DCSE_start.....	293
Modelica.Electrical.Machines.Examples.TransformerTestbench.....	294

---

Modelica.Electrical.Machines.Examples.Rectifier6pulse.....	294
Modelica.Electrical.Machines.Examples.Rectifier12pulse.....	294
Modelica.Electrical.Machines.Examples.AIMC_Steinmetz.....	295
Modelica.Electrical.Machines.Examples.Utilities.....	295
Modelica.Electrical.Machines.Examples.Utilities.VfController.....	296
Modelica.Electrical.Machines.Examples.Utilities.SwitchYD.....	296
Modelica.Electrical.Machines.Examples.Utilities.TerminalBox.....	296
Modelica.Electrical.Machines.BasicMachines.....	297
Modelica.Electrical.Machines.BasicMachines.AynchronousInductionMachines.....	297
Modelica.Electrical.Machines.BasicMachines.AynchronousInductionMachines.AIM_SquirrelCage.....	298
Modelica.Electrical.Machines.BasicMachines.AynchronousInductionMachines.AIM_SlipRing.....	299
Modelica.Electrical.Machines.BasicMachines.SynchronousInductionMachines.....	300
Modelica.Electrical.Machines.BasicMachines.SynchronousInductionMachines.SM_PermanentMagnetDamperCage.....	301
Modelica.Electrical.Machines.BasicMachines.SynchronousInductionMachines.SM_ElectricalExcitedDumperCage.....	303
Modelica.Electrical.Machines.BasicMachines.SynchronousInductionMachines.SM_ReluctanceRotorDamperCage.....	304
Modelica.Electrical.Machines.BasicMachines.DCMachines.....	306
Modelica.Electrical.Machines.BasicMachines.DC_PermanentMagnet.....	306
Modelica.Electrical.Machines.BasicMachines.DC_ElectricalExcited.....	307
Modelica.Electrical.Machines.BasicMachines.DCMachines.DC_SeriesExcited.....	308
Modelica.Electrical.Machines.BasicMachines.Transformers.....	309
Modelica.Electrical.Machines.BasicMachines.Transformers.Transformer.....	311
Modelica.Electrical.Machines.BasicMachines.Transformers.Yy.....	311
Modelica.Electrical.Machines.BasicMachines.Transformers.Yy.Yy00.....	312
Modelica.Electrical.Machines.BasicMachines.Transformers.Yy.Yy02.....	312
Modelica.Electrical.Machines.BasicMachines.Transformers.Yy.Yy04.....	313
Modelica.Electrical.Machines.BasicMachines.Transformers.Yy.Yy06.....	313
Modelica.Electrical.Machines.BasicMachines.Transformers.Yy.Yy08.....	314
Modelica.Electrical.Machines.BasicMachines.Transformers.Yy.Yy10.....	314
Modelica.Electrical.Machines.BasicMachines.Transformers.Yd.....	315
Modelica.Electrical.Machines.BasicMachines.Transformers.Yd.Yd01.....	315
Modelica.Electrical.Machines.BasicMachines.Transformers.Yd.Yd03.....	316
Modelica.Electrical.Machines.BasicMachines.Transformers.Yd.Yd05.....	316
Modelica.Electrical.Machines.BasicMachines.Transformers.Yd.Yd07.....	317
Modelica.Electrical.Machines.BasicMachines.Transformers.Yd.Yd09.....	317
Modelica.Electrical.Machines.BasicMachines.Transformers.Yd.Yd11.....	318
Modelica.Electrical.Machines.BasicMachines.Transformers.Yz.....	318
Modelica.Electrical.Machines.BasicMachines.Transformers.Yz.Yz01.....	319
Modelica.Electrical.Machines.BasicMachines.Transformers.Yz.Yz03.....	319
Modelica.Electrical.Machines.BasicMachines.Transformers.Yz.Yz05.....	320
Modelica.Electrical.Machines.BasicMachines.Transformers.Yz.Yz07.....	320
Modelica.Electrical.Machines.BasicMachines.Transformers.Yz.Yz09.....	321
Modelica.Electrical.Machines.BasicMachines.Transformers.Yz.Yz11.....	321
Modelica.Electrical.Machines.BasicMachines.Transformers.Dy.....	322
Modelica.Electrical.Machines.BasicMachines.Transformers.Dy.Dy01.....	322
Modelica.Electrical.Machines.BasicMachines.Transformers.Dy.Dy03.....	323
Modelica.Electrical.Machines.BasicMachines.Transformers.Dy.Dy05.....	323
Modelica.Electrical.Machines.BasicMachines.Transformers.Dy.Dy07.....	324
Modelica.Electrical.Machines.BasicMachines.Transformers.Dy.Dy09.....	324
Modelica.Electrical.Machines.BasicMachines.Transformers.Dy.Dy11.....	325
Modelica.Electrical.Machines.BasicMachines.Transformers.Dd.....	325
Modelica.Electrical.Machines.BasicMachines.Transformers.Dd.Dd00.....	326
Modelica.Electrical.Machines.BasicMachines.Transformers.Dd.Dd02.....	326
Modelica.Electrical.Machines.BasicMachines.Transformers.Dd.Dd04.....	327
Modelica.Electrical.Machines.BasicMachines.Transformers.Dd.Dd06.....	327
Modelica.Electrical.Machines.BasicMachines.Transformers.Dd.Dd08.....	328

---

Modelica.Electrical.Machines.BasicMachines.Transformers.Dd.Dd10.....	328
Modelica.Electrical.Machines.BasicMachines.Transformers.Dz.....	329
Modelica.Electrical.Machines.BasicMachines.Transformers.Dz.Dz00.....	329
Modelica.Electrical.Machines.BasicMachines.Transformers.Dz.Dz02.....	330
Modelica.Electrical.Machines.BasicMachines.Transformers.Dz.Dz04.....	330
Modelica.Electrical.Machines.BasicMachines.Transformers.Dz.Dz06.....	331
Modelica.Electrical.Machines.BasicMachines.Transformers.Dz.Dz08.....	331
Modelica.Electrical.Machines.BasicMachines.Transformers.Dz.Dz10.....	332
Modelica.Electrical.Machines.BasicMachines.Components.....	332
Modelica.Electrical.Machines.BasicMachines.Components.BasicAIM.....	333
Modelica.Electrical.Machines.BasicMachines.Components.BasicSM.....	333
Modelica.Electrical.Machines.BasicMachines.Components.PartialAirGap.....	334
Modelica.Electrical.Machines.BasicMachines.Components.AirGapS.....	334
Modelica.Electrical.Machines.BasicMachines.Components.AirGapR.....	335
Modelica.Electrical.Machines.BasicMachines.Components.SquirrelCage.....	335
Modelica.Electrical.Machines.BasicMachines.Components.DamperCage.....	336
Modelica.Electrical.Machines.BasicMachines.Components.ElectricalExcitation.....	336
Modelica.Electrical.Machines.BasicMachines.Components.PermanentMagnet.....	337
Modelica.Electrical.Machines.BasicMachines.Components.BasicDCMachine.....	337
Modelica.Electrical.Machines.BasicMachines.Components.PartialAirGapDC.....	338
Modelica.Electrical.Machines.BasicMachines.Components.AirGapDC.....	338
Modelica.Electrical.Machines.BasicMachines.Components.BasicTransformer.....	339
Modelica.Electrical.Machines.BasicMachines.Components.PartialCore.....	339
Modelica.Electrical.Machines.BasicMachines.Components.IdealCore.....	340
Modelica.Electrical.Machines.Sensors.....	340
Modelica.Electrical.Machines.Sensors.VoltageRMSSensor.....	341
Modelica.Electrical.Machines.Sensors.CurrentRMSSensor.....	341
Modelica.Electrical.Machines.Sensors.ElectricalPowerSensor.....	341
Modelica.Electrical.Machines.Sensors.MechanicalPowerSensor.....	342
Modelica.Electrical.Machines.Sensors.RotorAngle.....	342
Modelica.Electrical.Machines.SpacePhasors.....	343
Modelica.Electrical.Machines.SpacePhasors.Components.....	343
Modelica.Electrical.Machines.SpacePhasors.Components.SpacePhasor.....	343
Modelica.Electrical.Machines.SpacePhasors.Components.Rotator.....	344
Modelica.Electrical.Machines.SpacePhasors.Blocks.....	344
Modelica.Electrical.Machines.SpacePhasors.Blocks.ToSpacePhasor.....	345
Modelica.Electrical.Machines.SpacePhasors.Blocks.FromSpacePhasor.....	345
Modelica.Electrical.Machines.SpacePhasors.Blocks.Rotator.....	345
Modelica.Electrical.Machines.SpacePhasors.Blocks.ToPolar.....	345
Modelica.Electrical.Machines.SpacePhasors.Blocks.FromPolar.....	346
Modelica.Electrical.Machines.SpacePhasors.Functions.....	346
Modelica.Electrical.Machines.SpacePhasors.Functions.ToSpacePhasor.....	347
Modelica.Electrical.Machines.SpacePhasors.Functions.FromSpacePhasor.....	347
Modelica.Electrical.Machines.SpacePhasors.Functions.Rotator.....	347
Modelica.Electrical.Machines.SpacePhasors.Functions.ToPolar.....	348
Modelica.Electrical.Machines.SpacePhasors.Functions.FromPolar.....	348
Modelica.Electrical.Machines.Interfaces.....	349
Modelica.Electrical.Machines.Interfaces.SpacePhasor.....	349
Modelica.Electrical.Machines.Interfaces.Adapter.....	349
Modelica.Electrical.Machines.Interfaces.PartialBasicMachine.....	350
Modelica.Electrical.Machines.Interfaces.PartialBasicInductionMachine.....	350
Modelica.Electrical.Machines.Interfaces.PartialBasicDCMachine.....	351
Modelica.Electrical.MultiPhase.....	351
Modelica.Electrical.MultiPhase.Basic.....	352
Modelica.Electrical.MultiPhase.Basic.Star.....	353
Modelica.Electrical.MultiPhase.Basic.Delta.....	353
Modelica.Electrical.MultiPhase.Basic.PlugToPin_p.....	354
Modelica.Electrical.MultiPhase.Basic.PlugToPin_n.....	354

Modelica.Electrical.MultiPhase.Basic.Resistor.....	354
Modelica.Electrical.MultiPhase.Basic.Conductor.....	355
Modelica.Electrical.MultiPhase.Basic.Capacitor.....	355
Modelica.Electrical.MultiPhase.Basic.Inductor.....	356
Modelica.Electrical.MultiPhase.Basic.SaturatingInductor.....	356
Modelica.Electrical.MultiPhase.Basic.Transformer.....	356
Modelica.Electrical.MultiPhase.Basic.VariableResistor.....	357
Modelica.Electrical.MultiPhase.Basic.VariableConductor.....	357
Modelica.Electrical.MultiPhase.Basic.VariableCapacitor.....	358
Modelica.Electrical.MultiPhase.Basic.VariableInductor.....	358
Modelica.Electrical.MultiPhase.Examples.....	359
Modelica.Electrical.MultiPhase.Examples.TransformerYY.....	359
Modelica.Electrical.MultiPhase.Examples.TransformerYD.....	360
Modelica.Electrical.MultiPhase.Examples.Rectifier.....	360
Modelica.Electrical.MultiPhase.Ideal.....	360
Modelica.Electrical.MultiPhase.Ideal.IdealThyristor.....	361
Modelica.Electrical.MultiPhase.Ideal.IdealGTOThyristor.....	361
Modelica.Electrical.MultiPhase.Ideal.IdealCommutingSwitch.....	362
Modelica.Electrical.MultiPhase.Ideal.IdealIntermediateSwitch.....	362
Modelica.Electrical.MultiPhase.Ideal.IdealDiode.....	363
Modelica.Electrical.MultiPhase.Ideal.IdealTransformer.....	363
Modelica.Electrical.MultiPhase.Ideal.Idle.....	364
Modelica.Electrical.MultiPhase.Ideal.Short.....	364
Modelica.Electrical.MultiPhase.Ideal.IdealOpeningSwitch.....	364
Modelica.Electrical.MultiPhase.Ideal.IdealClosingSwitch.....	365
Modelica.Electrical.MultiPhase.Interfaces.....	365
Modelica.Electrical.MultiPhase.Interfaces.Plug.....	366
Modelica.Electrical.MultiPhase.Interfaces.PositivePlug.....	366
Modelica.Electrical.MultiPhase.Interfaces.NegativePlug.....	367
Modelica.Electrical.MultiPhase.Interfaces.TwoPlug.....	367
Modelica.Electrical.MultiPhase.Interfaces.OnePort.....	367
Modelica.Electrical.MultiPhase.Interfaces.FourPlug.....	368
Modelica.Electrical.MultiPhase.Interfaces.TwoPort.....	368
Modelica.Electrical.MultiPhase.Sensors.....	369
Modelica.Electrical.MultiPhase.Sensors.PotentialSensor.....	369
Modelica.Electrical.MultiPhase.Sensors.VoltageSensor.....	369
Modelica.Electrical.MultiPhase.Sensors.CurrentSensor.....	370
Modelica.Electrical.MultiPhase.Sensors.PowerSensor.....	370
Modelica.Electrical.MultiPhase.Sources.....	371
Modelica.Electrical.MultiPhase.Sources.SignalVoltage.....	371
Modelica.Electrical.MultiPhase.Sources.ConstantVoltage.....	372
Modelica.Electrical.MultiPhase.Sources.SineVoltage.....	372
Modelica.Electrical.MultiPhase.Sources.SignalCurrent.....	373
Modelica.Electrical.MultiPhase.Sources.ConstantCurrent.....	373
Modelica.Electrical.MultiPhase.Sources.SineCurrent.....	373
Modelica(Icons).....	374
Modelica(Icons.Info).....	375
Modelica(Icons.Library).....	375
Modelica(Icons.Library2).....	375
Modelica(Icons.Example).....	376
Modelica(Icons.Function).....	376
Modelica(Icons.Record).....	376
Modelica(Icons.Enumeration).....	376
Modelica(Icons.TypeReal).....	376
Modelica(Icons.TypeInteger).....	376
Modelica(Icons.TypeBoolean).....	377
Modelica(Icons.TypeString).....	377
Modelica(Icons.TranslationalSensor).....	377

Modelica.Icons.RotationalSensor.....	377
Modelica.Icons.GearIcon.....	377
Modelica.Icons.MotorIcon.....	377
Modelica.Icons.SignalBus.....	378
Modelica.Icons.SignalSubBus.....	378
Modelica.Math.....	378
Modelica.Math.Vectors.....	379
Modelica.Math.Vectors.isEqual.....	380
Modelica.Math.Vectors.norm.....	381
Modelica.Math.Vectors.length.....	382
Modelica.Math.Vectors.normalize.....	382
Modelica.Math.Vectors.reverse.....	383
Modelica.Math.Vectors.sort.....	384
Modelica.Math.Matrices.....	385
Modelica.Math.Matrices.isEqual.....	386
Modelica.Math.Matrices.norm.....	387
Modelica.Math.Matrices.sort.....	388
Modelica.Math.Matrices.solve.....	389
Modelica.Math.Matrices.solve2.....	390
Modelica.Math.Matrices.leastSquares.....	391
Modelica.Math.Matrices.equalityLeastSquares.....	391
Modelica.Math.Matrices.LU.....	392
Modelica.Math.Matrices.LU_solve.....	393
Modelica.Math.Matrices.LU_solve2.....	394
Modelica.Math.Matrices.QR.....	396
Modelica.Math.Matrices.eigenValues.....	397
Modelica.Math.Matrices.eigenValueMatrix.....	398
Modelica.Math.Matrices.singularValues.....	398
Modelica.Math.Matrices.det.....	399
Modelica.Math.Matrices.inv.....	400
Modelica.Math.Matrices.rank.....	400
Modelica.Math.Matrices.balance.....	401
Modelica.Math.Matrices.exp.....	402
Modelica.Math.Matrices.integralExp.....	403
Modelica.Math.Matrices.integralExpT.....	404
Modelica.Math.sin.....	405
Modelica.Math.cos.....	405
Modelica.Math.tan.....	406
Modelica.Math.asin.....	406
Modelica.Math.acos.....	406
Modelica.Math.atan.....	407
Modelica.Math.atan2.....	407
Modelica.Math.atan3.....	408
Modelica.Math.sinh.....	408
Modelica.Math.cosh.....	409
Modelica.Math.tanh.....	409
Modelica.Math.asinh.....	409
Modelica.Math.acosh.....	410
Modelica.Math.exp.....	410
Modelica.Math.log.....	411
Modelica.Math.log10.....	411
Modelica.Math.baselcon1.....	412
Modelica.Math.baselcon2.....	412
Modelica.Math.templInterpol1.....	412
Modelica.Math.templInterpol2.....	412
Modelica.Mechanics.....	413
Modelica.Mechanics.MultiBody.....	413
Modelica.Mechanics.MultiBody.UsersGuide.....	414

---

Modelica.Mechanics.MultiBody.World.....	414
Modelica.Mechanics.MultiBody.Examples.....	416
Modelica.Mechanics.MultiBody.Examples.Elementary.....	417
Modelica.Mechanics.MultiBody.Examples.Elementary.DoublePendulum.....	420
Modelica.Mechanics.MultiBody.Examples.Elementary.ForceAndTorque.....	420
Modelica.Mechanics.MultiBody.Examples.Elementary.FreeBody.....	421
Modelica.Mechanics.MultiBody.Examples.Elementary.InitSpringConstant.....	421
Modelica.Mechanics.MultiBody.Examples.Elementary.LineForceWithTwoMasses.....	422
Modelica.Mechanics.MultiBody.Examples.Elementary.Pendulum.....	422
Modelica.Mechanics.MultiBody.Examples.Elementary.PendulumWithSpringDamper.....	423
Modelica.Mechanics.MultiBody.Examples.Elementary.PointGravity.....	423
Modelica.Mechanics.MultiBody.Examples.Elementary.PointGravityWithPointMasses.....	423
Modelica.Mechanics.MultiBody.Examples.Elementary.PointGravityWithPointMasses2.....	424
Modelica.Mechanics.MultiBody.Examples.Elementary.SpringDamperSystem.....	424
Modelica.Mechanics.MultiBody.Examples.Elementary.SpringMassSystem.....	424
Modelica.Mechanics.MultiBody.Examples.Elementary.SpringWithMass.....	425
Modelica.Mechanics.MultiBody.Examples.Elementary.ThreeSprings.....	425
Modelica.Mechanics.MultiBody.Examples.Loops.....	426
Modelica.Mechanics.MultiBody.Examples.Loops.Engine1a.....	427
Modelica.Mechanics.MultiBody.Examples.Loops.Engine1b.....	428
Modelica.Mechanics.MultiBody.Examples.Loops.Engine1b_analytic.....	428
Modelica.Mechanics.MultiBody.Examples.Loops.EngineV6.....	428
Modelica.Mechanics.MultiBody.Examples.Loops.EngineV6_analytic.....	429
Modelica.Mechanics.MultiBody.Examples.Loops.Fourbar1.....	429
Modelica.Mechanics.MultiBody.Examples.Loops.Fourbar2.....	430
Modelica.Mechanics.MultiBody.Examples.Loops.Fourbar_analytic.....	430
Modelica.Mechanics.MultiBody.Examples.Loops.PlanarLoops_analytic.....	430
Modelica.Mechanics.MultiBody.Examples.Loops.Utilities.....	431
Modelica.Mechanics.MultiBody.Examples.Loops.Utilities.Cylinder.....	431
Modelica.Mechanics.MultiBody.Examples.Loops.Utilities.GasForce.....	432
Modelica.Mechanics.MultiBody.Examples.Loops.Utilities.GasForce2.....	432
Modelica.Mechanics.MultiBody.Examples.Loops.Utilities.CylinderBase.....	433
Modelica.Mechanics.MultiBody.Examples.Loops.Utilities.Cylinder_analytic_CAD.....	434
Modelica.Mechanics.MultiBody.Examples.Loops.Utilities.EngineV6_analytic.....	435
Modelica.Mechanics.MultiBody.Examples.Loops.Utilities.Engine1bBase.....	435
Modelica.Mechanics.MultiBody.Examples.Systems.....	435
Modelica.Mechanics.MultiBody.Examples.Systems.RobotR3.....	436
Modelica.Mechanics.MultiBody.Examples.Systems.RobotR3.oneAxis.....	437
Modelica.Mechanics.MultiBody.Examples.Systems.RobotR3.fullRobot.....	437
Modelica.Mechanics.MultiBody.Examples.Systems.RobotR3.Components.....	439
Modelica.Mechanics.MultiBody.Examples.Systems.RobotR3.Components.AxisControlBus.....	439
Modelica.Mechanics.MultiBody.Examples.Systems.RobotR3.Components.ControlBus.....	440
Modelica.Mechanics.MultiBody.Examples.Systems.RobotR3.Components.PathPlanning1.....	440
Modelica.Mechanics.MultiBody.Examples.Systems.RobotR3.Components.PathPlanning6.....	440
Modelica.Mechanics.MultiBody.Examples.Systems.RobotR3.Components.PathToAxisControlBus.....	441
Modelica.Mechanics.MultiBody.Examples.Systems.RobotR3.Components.GearType1.....	442
Modelica.Mechanics.MultiBody.Examples.Systems.RobotR3.Components.GearType2.....	442
Modelica.Mechanics.MultiBody.Examples.Systems.RobotR3.Components.Motor.....	443
Modelica.Mechanics.MultiBody.Examples.Systems.RobotR3.Components.Controller.....	443
Modelica.Mechanics.MultiBody.Examples.Systems.RobotR3.Components.AxisType1.....	444
Modelica.Mechanics.MultiBody.Examples.Systems.RobotR3.Components.AxisType2.....	444
Modelica.Mechanics.MultiBody.Examples.Systems.RobotR3.Components.MechanicalStructure.....	445
Modelica.Mechanics.MultiBody.Forces.....	446
Modelica.Mechanics.MultiBody.Forces.WorldForce.....	448
Modelica.Mechanics.MultiBody.Forces.WorldTorque.....	448
Modelica.Mechanics.MultiBody.Forces.WorldForceAndTorque.....	449
Modelica.Mechanics.MultiBody.Forces.FrameForce.....	450
Modelica.Mechanics.MultiBody.Forces.FrameTorque.....	451

---

Modelica.Mechanics.MultiBody.Forces.FrameForceAndTorque.....	452
Modelica.Mechanics.MultiBody.Forces.Force.....	453
Modelica.Mechanics.MultiBody.Forces.Torque.....	454
Modelica.Mechanics.MultiBody.Forces.ForceAndTorque.....	455
Modelica.Mechanics.MultiBody.Forces.LineForceWithMass.....	456
Modelica.Mechanics.MultiBody.Forces.LineForceWithTwoMasses.....	457
Modelica.Mechanics.MultiBody.Forces.Spring.....	459
Modelica.Mechanics.MultiBody.Forces.Damper.....	461
Modelica.Mechanics.MultiBody.Forces.SpringDamperParallel.....	461
Modelica.Mechanics.MultiBody.Forces.SpringDamperSeries.....	462
Modelica.Mechanics.MultiBody.Frames.....	463
Modelica.Mechanics.MultiBody.Frames.Orientation.....	465
Modelica.Mechanics.MultiBody.Frames.orientationConstraint.....	466
Modelica.Mechanics.MultiBody.Frames.angularVelocity1.....	466
Modelica.Mechanics.MultiBody.Frames.angularVelocity2.....	467
Modelica.Mechanics.MultiBody.Frames.resolve1.....	467
Modelica.Mechanics.MultiBody.Frames.resolve2.....	467
Modelica.Mechanics.MultiBody.Frames.resolveRelative.....	468
Modelica.Mechanics.MultiBody.Frames.resolveDyade1.....	468
Modelica.Mechanics.MultiBody.Frames.resolveDyade2.....	468
Modelica.Mechanics.MultiBody.Frames.nullRotation.....	468
Modelica.Mechanics.MultiBody.Frames.inverseRotation.....	469
Modelica.Mechanics.MultiBody.Frames.relativeRotation.....	469
Modelica.Mechanics.MultiBody.Frames.absoluteRotation.....	469
Modelica.Mechanics.MultiBody.Frames.planarRotation.....	470
Modelica.Mechanics.MultiBody.Frames.planarRotationAngle.....	470
Modelica.Mechanics.MultiBody.Frames.axisRotation.....	471
Modelica.Mechanics.MultiBody.Frames.axesRotations.....	471
Modelica.Mechanics.MultiBody.Frames.axesRotationsAngles.....	471
Modelica.Mechanics.MultiBody.Frames.smallRotation.....	472
Modelica.Mechanics.MultiBody.Frames.from_nxy.....	473
Modelica.Mechanics.MultiBody.Frames.from_nxz.....	473
Modelica.Mechanics.MultiBody.Frames.from_T.....	474
Modelica.Mechanics.MultiBody.Frames.from_T2.....	474
Modelica.Mechanics.MultiBody.Frames.from_T_inv.....	474
Modelica.Mechanics.MultiBody.Frames.from_Q.....	475
Modelica.Mechanics.MultiBody.Frames.to_T.....	475
Modelica.Mechanics.MultiBody.Frames.to_T_inv.....	475
Modelica.Mechanics.MultiBody.Frames.to_Q.....	476
Modelica.Mechanics.MultiBody.Frames.to_vector.....	476
Modelica.Mechanics.MultiBody.Frames.to_exy.....	476
Modelica.Mechanics.MultiBody.Frames.length.....	476
Modelica.Mechanics.MultiBody.Frames.normalize.....	477
Modelica.Mechanics.MultiBody.Frames.axis.....	477
Modelica.Mechanics.MultiBody.Frames.Quaternions.....	477
Modelica.Mechanics.MultiBody.Frames.Quaternions.Orientation.....	479
Modelica.Mechanics.MultiBody.Frames.Quaternions.der_Orientation.....	480
Modelica.Mechanics.MultiBody.Frames.Quaternions.orientationConstraint.....	480
Modelica.Mechanics.MultiBody.Frames.Quaternions.angularVelocity1.....	480
Modelica.Mechanics.MultiBody.Frames.Quaternions.angularVelocity2.....	481
Modelica.Mechanics.MultiBody.Frames.Quaternions.resolve1.....	481
Modelica.Mechanics.MultiBody.Frames.Quaternions.resolve2.....	481
Modelica.Mechanics.MultiBody.Frames.Quaternions.multipleResolve1.....	481
Modelica.Mechanics.MultiBody.Frames.Quaternions.multipleResolve2.....	482
Modelica.Mechanics.MultiBody.Frames.Quaternions.nullRotation.....	482
Modelica.Mechanics.MultiBody.Frames.Quaternions.inverseRotation.....	482
Modelica.Mechanics.MultiBody.Frames.Quaternions.relativeRotation.....	483
Modelica.Mechanics.MultiBody.Frames.Quaternions.absoluteRotation.....	483

Modelica.Mechanics.MultiBody.Frames.Quaternions.planarRotation.....	483
Modelica.Mechanics.MultiBody.Frames.Quaternions.smallRotation.....	483
Modelica.Mechanics.MultiBody.Frames.Quaternions.from_T.....	484
Modelica.Mechanics.MultiBody.Frames.Quaternions.from_T_inv.....	484
Modelica.Mechanics.MultiBody.Frames.Quaternions.to_T.....	484
Modelica.Mechanics.MultiBody.Frames.Quaternions.to_T_inv.....	485
Modelica.Mechanics.MultiBody.Frames.TransformationMatrices.....	485
Modelica.Mechanics.MultiBody.Frames.TransformationMatrices.Orientation.....	488
Modelica.Mechanics.MultiBody.Frames.TransformationMatrices.der_Orientation.....	488
Modelica.Mechanics.MultiBody.Frames.TransformationMatrices.orientationConstraint.....	489
Modelica.Mechanics.MultiBody.Frames.TransformationMatrices.angularVelocity1.....	489
Modelica.Mechanics.MultiBody.Frames.TransformationMatrices.angularVelocity2.....	489
Modelica.Mechanics.MultiBody.Frames.TransformationMatrices.resolve1.....	489
Modelica.Mechanics.MultiBody.Frames.TransformationMatrices.resolve2.....	490
Modelica.Mechanics.MultiBody.Frames.TransformationMatrices.multipleResolve1.....	490
Modelica.Mechanics.MultiBody.Frames.TransformationMatrices.multipleResolve2.....	490
Modelica.Mechanics.MultiBody.Frames.TransformationMatrices.resolveDyade1.....	491
Modelica.Mechanics.MultiBody.Frames.TransformationMatrices.resolveDyade2.....	491
Modelica.Mechanics.MultiBody.Frames.TransformationMatrices.nullRotation.....	491
Modelica.Mechanics.MultiBody.Frames.TransformationMatrices.inverseRotation.....	491
Modelica.Mechanics.MultiBody.Frames.TransformationMatrices.relativeRotation.....	492
Modelica.Mechanics.MultiBody.Frames.TransformationMatrices.absoluteRotation.....	492
Modelica.Mechanics.MultiBody.Frames.TransformationMatrices.planarRotation.....	492
Modelica.Mechanics.MultiBody.Frames.TransformationMatrices.planarRotationAngle.....	493
Modelica.Mechanics.MultiBody.Frames.TransformationMatrices.axisRotation.....	493
Modelica.Mechanics.MultiBody.Frames.TransformationMatrices.axesRotations.....	494
Modelica.Mechanics.MultiBody.Frames.TransformationMatrices.axesRotationsAngles.....	494
Modelica.Mechanics.MultiBody.Frames.TransformationMatrices.smallRotation.....	495
Modelica.Mechanics.MultiBody.Frames.TransformationMatrices.from_nxy.....	495
Modelica.Mechanics.MultiBody.Frames.TransformationMatrices.from_nxz.....	496
Modelica.Mechanics.MultiBody.Frames.TransformationMatrices.from_T.....	496
Modelica.Mechanics.MultiBody.Frames.TransformationMatrices.from_T_inv.....	497
Modelica.Mechanics.MultiBody.Frames.TransformationMatrices.from_Q.....	497
Modelica.Mechanics.MultiBody.Frames.TransformationMatrices.to_T.....	497
Modelica.Mechanics.MultiBody.Frames.TransformationMatrices.to_T_inv.....	497
Modelica.Mechanics.MultiBody.Frames.TransformationMatrices.to_Q.....	498
Modelica.Mechanics.MultiBody.Frames.TransformationMatrices.to_vector.....	498
Modelica.Mechanics.MultiBody.Frames.TransformationMatrices.to_exy.....	498
Modelica.Mechanics.MultiBody.Interfaces.....	499
Modelica.Mechanics.MultiBody.Interfaces.Frame.....	500
Modelica.Mechanics.MultiBody.Interfaces.Frame_a.....	500
Modelica.Mechanics.MultiBody.Interfaces.Frame_b.....	500
Modelica.Mechanics.MultiBody.Interfaces.Frame_resolve.....	501
Modelica.Mechanics.MultiBody.Interfaces.FlangeWithBearing.....	501
Modelica.Mechanics.MultiBody.Interfaces.FlangeWithBearingAdaptor.....	501
Modelica.Mechanics.MultiBody.Interfaces.PartialTwoFrames.....	502
Modelica.Mechanics.MultiBody.Interfaces.PartialTwoFramesDoubleSize.....	502
Modelica.Mechanics.MultiBody.Interfaces.PartialOneFrame_a.....	503
Modelica.Mechanics.MultiBody.Interfaces.PartialOneFrame_b.....	503
Modelica.Mechanics.MultiBody.Interfaces.PartialElementaryJoint.....	503
Modelica.Mechanics.MultiBody.Interfaces.PartialForce.....	504
Modelica.Mechanics.MultiBody.Interfaces.PartialLineForce.....	504
Modelica.Mechanics.MultiBody.Interfaces.PartialAbsoluteSensor.....	505
Modelica.Mechanics.MultiBody.Interfaces.PartialRelativeSensor.....	505
Modelica.Mechanics.MultiBody.Interfaces.PartialCutForceSensor.....	506
Modelica.Mechanics.MultiBody.Interfaces.PartialVisualizer.....	506
Modelica.Mechanics.MultiBody.Joints.....	506
Modelica.Mechanics.MultiBody.Joints.Prismatic.....	509

---

Modelica.Mechanics.MultiBody.Joints.ActuatedPrismatic.....	510
Modelica.Mechanics.MultiBody.Joints.Revolute.....	512
Modelica.Mechanics.MultiBody.Joints.ActuatedRevolute.....	513
Modelica.Mechanics.MultiBody.Joints.Cylindrical.....	514
Modelica.Mechanics.MultiBody.Joints.Universal.....	515
Modelica.Mechanics.MultiBody.Joints.Planar.....	517
Modelica.Mechanics.MultiBody.Joints.Spherical.....	518
Modelica.Mechanics.MultiBody.Joints.FreeMotion.....	519
Modelica.Mechanics.MultiBody.Joints.SphericalSpherical.....	521
Modelica.Mechanics.MultiBody.Joints.UniversalSpherical.....	522
Modelica.Mechanics.MultiBody.Joints.GearConstraint.....	525
Modelica.Mechanics.MultiBody.Joints.Assemblies.....	525
Modelica.Mechanics.MultiBody.Joints.Assemblies.JointUPS.....	527
Modelica.Mechanics.MultiBody.Joints.Assemblies.JointUSR.....	529
Modelica.Mechanics.MultiBody.Joints.Assemblies.JointUSP.....	532
Modelica.Mechanics.MultiBody.Joints.Assemblies.JointSSR.....	535
Modelica.Mechanics.MultiBody.Joints.Assemblies.JointSSP.....	536
Modelica.Mechanics.MultiBody.Joints.Assemblies.JointRRR.....	538
Modelica.Mechanics.MultiBody.Joints.Assemblies.JointRRP.....	539
Modelica.Mechanics.MultiBody.Parts.....	541
Modelica.Mechanics.MultiBody.Parts.Fixed.....	544
Modelica.Mechanics.MultiBody.Parts.FixedTranslation.....	545
Modelica.Mechanics.MultiBody.Parts.FixedRotation.....	545
Modelica.Mechanics.MultiBody.Parts.Body.....	547
Modelica.Mechanics.MultiBody.Parts.BodyShape.....	549
Modelica.Mechanics.MultiBody.Parts.BodyBox.....	551
Modelica.Mechanics.MultiBody.Parts.BodyCylinder.....	552
Modelica.Mechanics.MultiBody.Parts.PointMass.....	554
Modelica.Mechanics.MultiBody.Parts.Mounting1D.....	554
Modelica.Mechanics.MultiBody.Parts.Rotor1D.....	555
Modelica.Mechanics.MultiBody.Parts.BevelGear1D.....	556
Modelica.Mechanics.MultiBody.Sensors.....	557
Modelica.Mechanics.MultiBody.Sensors.AbsoluteSensor.....	559
Modelica.Mechanics.MultiBody.Sensors.RelativeSensor.....	561
Modelica.Mechanics.MultiBody.Sensors.Distance.....	563
Modelica.Mechanics.MultiBody.Sensors.CutForce.....	564
Modelica.Mechanics.MultiBody.Sensors.CutTorque.....	565
Modelica.Mechanics.MultiBody.Sensors.CutForceAndTorque.....	566
Modelica.Mechanics.MultiBody.Sensors.Power.....	567
Modelica.Mechanics.MultiBody.Types.....	567
Modelica.Mechanics.MultiBody.Types.Axis.....	569
Modelica.Mechanics.MultiBody.Types.AxisLabel.....	569
Modelica.Mechanics.MultiBody.Types.RotationSequence.....	569
Modelica.Mechanics.MultiBody.Types.Color.....	569
Modelica.Mechanics.MultiBody.Types.SpecularCoefficient.....	569
Modelica.Mechanics.MultiBody.Types.ShapeType.....	570
Modelica.Mechanics.MultiBody.Types.ShapeExtra.....	570
Modelica.Mechanics.MultiBody.Types.AngularVelocity_degs.....	570
Modelica.Mechanics.MultiBody.Types.AngularAcceleration_degs2.....	571
Modelica.Mechanics.MultiBody.Types.RotationTypes.....	571
Modelica.Mechanics.MultiBody.Types.RotationTypes.Temp.....	571
Modelica.Mechanics.MultiBody.Types.GravityTypes.....	572
Modelica.Mechanics.MultiBody.Types.GravityTypes.Temp.....	572
Modelica.Mechanics.MultiBody.Types.Init.....	572
Modelica.Mechanics.MultiBody.Types.Init.Temp.....	573
Modelica.Mechanics.MultiBody.Types.Defaults.....	574
Modelica.Mechanics.MultiBody.Visualizers.....	575
Modelica.Mechanics.MultiBody.Visualizers.FixedShape.....	576

Modelica.Mechanics.MultiBody.Visualizers.FixedShape2.....	578
Modelica.Mechanics.MultiBody.Visualizers.FixedFrame.....	579
Modelica.Mechanics.MultiBody.Visualizers.FixedArrow.....	580
Modelica.Mechanics.MultiBody.Visualizers.SignalArrow.....	580
Modelica.Mechanics.MultiBody.Visualizers.Advanced.....	581
Modelica.Mechanics.MultiBody.Visualizers.Advanced.Arrow.....	582
Modelica.Mechanics.MultiBody.Visualizers.Advanced.DoubleArrow.....	583
Modelica.Mechanics.MultiBody.Visualizers.Advanced.Shape.....	584
Modelica.Mechanics.Rotational.....	585
Modelica.Mechanics.Rotational.UsersGuide.....	587
Modelica.Mechanics.Rotational.UsersGuide.Overview.....	587
Modelica.Mechanics.Rotational.UsersGuide.FlangeConnectors.....	588
Modelica.Mechanics.Rotational.UsersGuide.SupportTorques.....	588
Modelica.Mechanics.Rotational.UsersGuide.SignConventions.....	588
Modelica.Mechanics.Rotational.UsersGuide.UserDefinedComponents.....	589
Modelica.Mechanics.Rotational.UsersGuide.RequirementsForSimulationTool.....	590
Modelica.Mechanics.Rotational.UsersGuide.ReleaseNotes.....	591
Modelica.Mechanics.Rotational.UsersGuide.Contact.....	591
Modelica.Mechanics.Rotational.Examples.....	591
Modelica.Mechanics.Rotational.Examples.First.....	592
Modelica.Mechanics.Rotational.Examples.Friction.....	592
Modelica.Mechanics.Rotational.Examples.CoupledClutches.....	593
Modelica.Mechanics.Rotational.Examples.LossyGearDemo1.....	593
Modelica.Mechanics.Rotational.Examples.LossyGearDemo2.....	594
Modelica.Mechanics.Rotational.Examples.ElasticBearing.....	594
Modelica.Mechanics.Rotational.Sensors.....	594
Modelica.Mechanics.Rotational.Sensors.AngleSensor.....	595
Modelica.Mechanics.Rotational.Sensors.SpeedSensor.....	595
Modelica.Mechanics.Rotational.Sensors.AccSensor.....	596
Modelica.Mechanics.Rotational.Sensors.RelAngleSensor.....	596
Modelica.Mechanics.Rotational.Sensors.RelSpeedSensor.....	596
Modelica.Mechanics.Rotational.Sensors.RelAccSensor.....	596
Modelica.Mechanics.Rotational.Sensors.TorqueSensor.....	597
Modelica.Mechanics.Rotational.Sensors.PowerSensor.....	597
Modelica.Mechanics.Rotational.Interfaces.....	597
Modelica.Mechanics.Rotational.Interfaces.Flange_a.....	598
Modelica.Mechanics.Rotational.Interfaces.Flange_b.....	599
Modelica.Mechanics.Rotational.Interfaces.Rigid.....	599
Modelica.Mechanics.Rotational.Interfaces.Compliant.....	599
Modelica.Mechanics.Rotational.Interfaces.TwoFlanges.....	600
Modelica.Mechanics.Rotational.Interfaces.Bearing.....	600
Modelica.Mechanics.Rotational.Interfaces.TwoFlangesAndBearing.....	600
Modelica.Mechanics.Rotational.Interfaces.TwoFlangesAndBearingH.....	601
Modelica.Mechanics.Rotational.Interfaces.FrictionBase.....	601
Modelica.Mechanics.Rotational.Interfaces.AbsoluteSensor.....	601
Modelica.Mechanics.Rotational.Interfaces.RelativeSensor.....	602
Modelica.Mechanics.Rotational.Interfaces.PartialSpeedDependentTorque.....	602
Modelica.Mechanics.Rotational.Inertia.....	602
Modelica.Mechanics.Rotational.IdealGear.....	603
Modelica.Mechanics.Rotational.IdealPlanetary.....	603
Modelica.Mechanics.Rotational.IdealGearR2T.....	604
Modelica.Mechanics.Rotational.Spring.....	604
Modelica.Mechanics.Rotational.Damper.....	605
Modelica.Mechanics.Rotational.SpringDamper.....	605
Modelica.Mechanics.Rotational.ElastoBacklash.....	606
Modelica.Mechanics.Rotational.BearingFriction.....	607
Modelica.Mechanics.Rotational.Clutch.....	608
Modelica.Mechanics.Rotational.OneWayClutch.....	610

---

Modelica.Mechanics.Rotational.Brake.....	611
Modelica.Mechanics.Rotational.LossyGear.....	613
Modelica.Mechanics.Rotational.GearEfficiency.....	614
Modelica.Mechanics.Rotational.Gear.....	615
Modelica.Mechanics.Rotational.Gear2.....	615
Modelica.Mechanics.Rotational.Position.....	616
Modelica.Mechanics.Rotational.Speed.....	617
Modelica.Mechanics.Rotational.Accelerate.....	617
Modelica.Mechanics.Rotational.Move.....	618
Modelica.Mechanics.Rotational.Fixed.....	618
Modelica.Mechanics.Rotational.Torque.....	619
Modelica.Mechanics.Rotational.Torque2.....	619
Modelica.Mechanics.Rotational.LinearSpeedDependentTorque.....	620
Modelica.Mechanics.Rotational.QuadraticSpeedDependentTorque.....	620
Modelica.Mechanics.Rotational.ConstantTorque.....	620
Modelica.Mechanics.Rotational.ConstantSpeed.....	621
Modelica.Mechanics.Rotational.TorqueStep.....	621
Modelica.Mechanics.Rotational.RelativeStates.....	622
Modelica.Mechanics.Rotational.InitializeFlange.....	622
Modelica.Mechanics.Rotational.Types.....	623
Modelica.Mechanics.Rotational.Types.Init.....	623
Modelica.Mechanics.Rotational.Types.Init.Temp.....	624
Modelica.Mechanics.Rotational.Types.InitRel.....	625
Modelica.Mechanics.Rotational.Types.InitRel.Temp.....	625
Modelica.Mechanics.Translational.....	626
Modelica.Mechanics.Translational.Examples.....	627
Modelica.Mechanics.Translational.Examples.SignConvention.....	628
Modelica.Mechanics.Translational.Examples.InitialConditions.....	628
Modelica.Mechanics.Translational.Examples.WhyArrows.....	629
Modelica.Mechanics.Translational.Examples.Accelerate.....	629
Modelica.Mechanics.Translational.Examples.Damper.....	629
Modelica.Mechanics.Translational.Examples.Oscillator.....	629
Modelica.Mechanics.Translational.Examples.Sensors.....	629
Modelica.Mechanics.Translational.Examples.Friction.....	630
Modelica.Mechanics.Translational.Examples.PreLoad.....	630
Modelica.Mechanics.Translational.Sensors.....	632
Modelica.Mechanics.Translational.Sensors.ForceSensor.....	632
Modelica.Mechanics.Translational.Sensors.PositionSensor.....	633
Modelica.Mechanics.Translational.Sensors.SpeedSensor.....	633
Modelica.Mechanics.Translational.Sensors.AccSensor.....	633
Modelica.Mechanics.Translational.Interfaces.....	634
Modelica.Mechanics.Translational.Interfaces.Flange_a.....	634
Modelica.Mechanics.Translational.Interfaces.Flange_b.....	634
Modelica.Mechanics.Translational.Interfaces.Rigid.....	635
Modelica.Mechanics.Translational.Interfaces.Compliant.....	635
Modelica.Mechanics.Translational.Interfaces.TwoFlanges.....	636
Modelica.Mechanics.Translational.Interfaces.AbsoluteSensor.....	636
Modelica.Mechanics.Translational.Interfaces.RelativeSensor.....	636
Modelica.Mechanics.Translational.Interfaces.FrictionBase.....	637
Modelica.Mechanics.Translational.SlidingMass.....	637
Modelica.Mechanics.Translational.Stop.....	637
Modelica.Mechanics.Translational.Rod.....	639
Modelica.Mechanics.Translational.Spring.....	640
Modelica.Mechanics.Translational.Damper.....	640
Modelica.Mechanics.Translational.SpringDamper.....	640
Modelica.Mechanics.Translational.ElastoGap.....	641
Modelica.Mechanics.Translational.Position.....	641
Modelica.Mechanics.Translational.Speed.....	642

Modelica.Mechanics.Translational.Accelerate.....	643
Modelica.Mechanics.Translational.Move.....	643
Modelica.Mechanics.Translational.Fixed.....	644
Modelica.Mechanics.Translational.Force.....	644
Modelica.Mechanics.Translational.RelativeStates.....	644
Modelica.Media.....	645
Modelica.Media.UsersGuide.....	646
Modelica.Media.Examples.....	647
Modelica.Media.Examples.SimpleLiquidWater.....	647
Modelica.Media.Examples.IdealGasH2O.....	648
Modelica.Media.Examples.WaterIF97.....	648
Modelica.Media.Examples.MixtureGases.....	648
Modelica.Media.Examples.MoistAir.....	648
Modelica.Media.Examples.TwoPhaseWater.....	649
Modelica.Media.Examples.TwoPhaseWater.ExtendedProperties.....	653
Modelica.Media.Examples.TwoPhaseWater.TestTwoPhaseStates.....	654
Modelica.Media.Examples.TestOnly.....	654
Modelica.Media.Examples.TestOnly.MixIdealGasAir.....	654
Modelica.Media.Examples.TestOnly.FlueGas.....	655
Modelica.Media.Examples.TestOnly.TestMedia.....	655
Modelica.Media.Examples.TestOnly.TestMedia.TemplateMedium.....	655
Modelica.Media.Examples.TestOnly.IdealGasAir.....	655
Modelica.Media.Examples.Tests.....	655
Modelica.Media.Examples.Tests.Components.....	656
Modelica.Media.Examples.Tests.Components.FluidPort.....	656
Modelica.Media.Examples.Tests.Components.FluidPort_a.....	657
Modelica.Media.Examples.Tests.Components.FluidPort_b.....	657
Modelica.Media.Examples.Tests.Components.PortVolume.....	658
Modelica.Media.Examples.Tests.Components.FixedMassFlowRate.....	658
Modelica.Media.Examples.Tests.Components.FixedAmbient.....	659
Modelica.Media.Examples.Tests.Components.ShortPipe.....	660
Modelica.Media.Examples.Tests.Components.PartialTestModel.....	660
Modelica.Media.Examples.Tests.Components.PartialTestModel2.....	660
Modelica.Media.Examples.Tests.MediaTestModels.....	661
Modelica.Media.Examples.Tests.MediaTestModels.Air.....	661
Modelica.Media.Examples.Tests.MediaTestModels.Air.SimpleAir.....	661
Modelica.Media.Examples.Tests.MediaTestModels.Air.DryAirNasa.....	662
Modelica.Media.Examples.Tests.MediaTestModels.Air.MoistAir.....	662
Modelica.Media.Examples.Tests.MediaTestModels.IdealGases.....	663
Modelica.Media.Examples.Tests.MediaTestModels.IdealGases.Air.....	663
Modelica.Media.Examples.Tests.MediaTestModels.IdealGases.SimpleNaturalGas.....	663
Modelica.Media.Examples.Tests.MediaTestModels.IdealGases.SimpleNaturalGasFixedComposition.....	664
Modelica.Media.Examples.Tests.MediaTestModels.Incompressible.....	664
Modelica.Media.Examples.Tests.MediaTestModels.Incompressible.Glycol47.....	664
Modelica.Media.Examples.Tests.MediaTestModels.Incompressible.Essotherm650.....	665
Modelica.Media.Examples.Tests.MediaTestModels.Water.....	665
Modelica.Media.Examples.Tests.MediaTestModels.Water.ConstantPropertyLiquidWater.....	665
Modelica.Media.Examples.Tests.MediaTestModels.Water.IdealSteam.....	666
Modelica.Media.Examples.Tests.MediaTestModels.Water.WaterIF97OnePhase_ph.....	666
Modelica.Media.Examples.Tests.MediaTestModels.Water.WaterIF97_pT.....	667
Modelica.Media.Examples.Tests.MediaTestModels.Water.WaterIF97_ph.....	667
Modelica.Media.Examples.Tests.MediaTestModels.LinearFluid.....	668
Modelica.Media.Examples.Tests.MediaTestModels.LinearFluid.LinearColdWater.....	668
Modelica.Media.Examples.Tests.MediaTestModels.LinearFluid.LinearWater_pT.....	668
Modelica.Media.Examples.SolveOneNonlinearEquation.....	669
Modelica.Media.Examples.SolveOneNonlinearEquation.Inverse_sine.....	669
Modelica.Media.Examples.SolveOneNonlinearEquation.Inverse_sh_T.....	669
Modelica.Media.Examples.SolveOneNonlinearEquation.Inverselncompressible_sh_T.....	670

---

Modelica.Media.Examples.SolveOneNonlinearEquation.Inverse_sh_TX	670
Modelica.Media.Interfaces	671
Modelica.Media.Interfaces.TemplateMedium	671
Modelica.Media.Interfaces.TemplateMedium.BaseProperties	674
Modelica.Media.Interfaces.TemplateMedium.ThermodynamicState	674
Modelica.Media.Interfaces.TemplateMedium.dynamicViscosity	674
Modelica.Media.Interfaces.TemplateMedium.thermalConductivity	675
Modelica.Media.Interfaces.TemplateMedium.specificEntropy	675
Modelica.Media.Interfaces.TemplateMedium.specificHeatCapacityCp	675
Modelica.Media.Interfaces.TemplateMedium.specificHeatCapacityCv	676
Modelica.Media.Interfaces.TemplateMedium.isentropicExponent	676
Modelica.Media.Interfaces.TemplateMedium.velocityOfSound	676
Modelica.Media.Interfaces.PartialMedium	676
Modelica.Media.Interfaces.PartialMedium.FluidConstants	683
Modelica.Media.Interfaces.PartialMedium.ThermodynamicState	683
Modelica.Media.Interfaces.PartialMedium.BasePropertiesRecord	684
Modelica.Media.Interfaces.PartialMedium.BaseProperties	684
Modelica.Media.Interfaces.PartialMedium.setState_pTX	685
Modelica.Media.Interfaces.PartialMedium.setState_phX	685
Modelica.Media.Interfaces.PartialMedium.setState_psX	686
Modelica.Media.Interfaces.PartialMedium.setState_dTX	686
Modelica.Media.Interfaces.PartialMedium.dynamicViscosity	686
Modelica.Media.Interfaces.PartialMedium.thermalConductivity	686
Modelica.Media.Interfaces.PartialMedium.prandtlNumber	687
Modelica.Media.Interfaces.PartialMedium.pressure	687
Modelica.Media.Interfaces.PartialMedium.temperature	687
Modelica.Media.Interfaces.PartialMedium.density	688
Modelica.Media.Interfaces.PartialMedium.specificEnthalpy	688
Modelica.Media.Interfaces.PartialMedium.specificInternalEnergy	688
Modelica.Media.Interfaces.PartialMedium.specificEntropy	688
Modelica.Media.Interfaces.PartialMedium.specificGibbsEnergy	689
Modelica.Media.Interfaces.PartialMedium.specificHelmholtzEnergy	689
Modelica.Media.Interfaces.PartialMedium.specificHeatCapacityCp	689
Modelica.Media.Interfaces.PartialMedium.heatCapacity_cp	689
Modelica.Media.Interfaces.PartialMedium.specificHeatCapacityCv	690
Modelica.Media.Interfaces.PartialMedium.heatCapacity_cv	690
Modelica.Media.Interfaces.PartialMedium.isentropicExponent	690
Modelica.Media.Interfaces.PartialMedium.isentropicEnthalpy	691
Modelica.Media.Interfaces.PartialMedium.velocityOfSound	691
Modelica.Media.Interfaces.PartialMedium.isobaricExpansionCoefficient	691
Modelica.Media.Interfaces.PartialMedium.beta	691
Modelica.Media.Interfaces.PartialMedium.thermalCompressibility	692
Modelica.Media.Interfaces.PartialMedium.kappa	692
Modelica.Media.Interfaces.PartialMedium.density_derP_h	692
Modelica.Media.Interfaces.PartialMedium.density_derh_p	693
Modelica.Media.Interfaces.PartialMedium.density_derP_T	693
Modelica.Media.Interfaces.PartialMedium.density_derT_p	693
Modelica.Media.Interfaces.PartialMedium.density_derX	693
Modelica.Media.Interfaces.PartialMedium.molarMass	694
Modelica.Media.Interfaces.PartialMedium.specificEnthalpy_pTX	694
Modelica.Media.Interfaces.PartialMedium.density_pTX	694
Modelica.Media.Interfaces.PartialMedium.temperature_phX	695
Modelica.Media.Interfaces.PartialMedium.density_phX	695
Modelica.Media.Interfaces.PartialMedium.temperature_psX	695
Modelica.Media.Interfaces.PartialMedium.density_psX	696
Modelica.Media.Interfaces.PartialMedium.specificEnthalpy_psX	696
Modelica.Media.Interfaces.PartialMedium.AbsolutePressure	696
Modelica.Media.Interfaces.PartialMedium.Density	696

Modelica.Media.Interfaces.PartialMedium.DynamicViscosity.....	697
Modelica.Media.Interfaces.PartialMedium.ENTHALPYFLOWRATE.....	697
Modelica.Media.Interfaces.PartialMedium.MASSFLOWRATE.....	697
Modelica.Media.Interfaces.PartialMedium.MASSFRACTION.....	697
Modelica.Media.Interfaces.PartialMedium.MOLEFRACTION.....	698
Modelica.Media.Interfaces.PartialMedium.MOLARMASS.....	698
Modelica.Media.Interfaces.PartialMedium.MOLARVOLUME.....	698
Modelica.Media.Interfaces.PartialMedium.ISENTROPICEXPONENT.....	698
Modelica.Media.Interfaces.PartialMedium.SPECIFICENERGY.....	699
Modelica.Media.Interfaces.PartialMedium.SPECIFICINTERNALENERGY.....	699
Modelica.Media.Interfaces.PartialMedium.SPECIFICENTHALPY.....	699
Modelica.Media.Interfaces.PartialMedium.SPECIFICENTROPY.....	699
Modelica.Media.Interfaces.PartialMedium.SPECIFICHEATCAPACITY.....	700
Modelica.Media.Interfaces.PartialMedium.SURFACTENSION.....	700
Modelica.Media.Interfaces.PartialMedium.TEMPERATURE.....	700
Modelica.Media.Interfaces.PartialMedium.THERMALCONDUCTIVITY.....	700
Modelica.Media.Interfaces.PartialMedium.PRANDTLNUMBER.....	700
Modelica.Media.Interfaces.PartialMedium.VELOCITYOFSOUND.....	701
Modelica.Media.Interfaces.PartialMedium.EXTRAPROPERTY.....	701
Modelica.Media.Interfaces.PartialMedium.CUMULATIVEEXTRAPROPERTY.....	701
Modelica.Media.Interfaces.PartialMedium.EXTRAPROPERTYFLOWRATE.....	701
Modelica.Media.Interfaces.PartialMedium.ISOBARICEXPANSIONCOEFFICIENT.....	701
Modelica.Media.Interfaces.PartialMedium.DIPOLEMOMENT.....	702
Modelica.Media.Interfaces.PartialMedium.DERDENSITYBYPRESSURE.....	702
Modelica.Media.Interfaces.PartialMedium.DERDENSITYBYENTHALPY.....	702
Modelica.Media.Interfaces.PartialMedium.DERENTHALPYBYPRESSURE.....	702
Modelica.Media.Interfaces.PartialMedium.DERDENSITYBYTEMPERATURE.....	702
Modelica.Media.Interfaces.PartialMedium.CHOICES.....	702
Modelica.Media.Interfaces.PartialMedium.CHOICES.INIT.....	703
Modelica.Media.Interfaces.PartialMedium.CHOICES.INIT.TEMP.....	703
Modelica.Media.Interfaces.PartialMedium.CHOICES.REFERENCEENTHALPY.....	703
Modelica.Media.Interfaces.PartialMedium.CHOICES.REFERENCEENTHALPY.TEMP.....	704
Modelica.Media.Interfaces.PartialMedium.CHOICES.REFERENCEENTROPY.....	704
Modelica.Media.Interfaces.PartialMedium.CHOICES.REFERENCEENTROPY.TEMP.....	705
Modelica.Media.Interfaces.PartialMedium.CHOICES.PD.....	705
Modelica.Media.Interfaces.PartialMedium.CHOICES.PD.TEMP.....	705
Modelica.Media.Interfaces.PartialMedium.CHOICES.TH.....	705
Modelica.Media.Interfaces.PartialMedium.CHOICES.TH.TEMP.....	706
Modelica.Media.Interfaces.PartialMedium.CHOICES.EXPLICIT.....	706
Modelica.Media.Interfaces.PartialMedium.CHOICES.EXPLICIT.TEMP.....	706
Modelica.Media.Interfaces.PartialPureSubstance.....	707
Modelica.Media.Interfaces.PartialPureSubstance.setSTATE_pT.....	710
Modelica.Media.Interfaces.PartialPureSubstance.setSTATE_ph.....	710
Modelica.Media.Interfaces.PartialPureSubstance.setSTATE_ps.....	710
Modelica.Media.Interfaces.PartialPureSubstance.setSTATE_dT.....	711
Modelica.Media.Interfaces.PartialPureSubstance.density_ph.....	711
Modelica.Media.Interfaces.PartialPureSubstance.temperature_ph.....	711
Modelica.Media.Interfaces.PartialPureSubstance.pressure_dT.....	712
Modelica.Media.Interfaces.PartialPureSubstance.specificEnthalpy_dT.....	712
Modelica.Media.Interfaces.PartialPureSubstance.specificEnthalpy_ps.....	712
Modelica.Media.Interfaces.PartialPureSubstance.temperature_ps.....	712
Modelica.Media.Interfaces.PartialPureSubstance.density_ps.....	713
Modelica.Media.Interfaces.PartialPureSubstance.specificEnthalpy_pT.....	713
Modelica.Media.Interfaces.PartialPureSubstance.density_pT.....	713
Modelica.Media.Interfaces.PartialLinearFluid.....	714
Modelica.Media.Interfaces.PartialLinearFluid.THERMODYNAMICSTATE.....	719
Modelica.Media.Interfaces.PartialLinearFluid.BASEPROPERTIES.....	719
Modelica.Media.Interfaces.PartialLinearFluid.setSTATE_pTX.....	719

---

Modelica.Media.Interfaces.PartialLinearFluid.setState_phX.....	719
Modelica.Media.Interfaces.PartialLinearFluid.setState_dTX.....	720
Modelica.Media.Interfaces.PartialLinearFluid.setState_psX.....	720
Modelica.Media.Interfaces.PartialLinearFluid.pressure.....	720
Modelica.Media.Interfaces.PartialLinearFluid.temperature.....	721
Modelica.Media.Interfaces.PartialLinearFluid.density.....	721
Modelica.Media.Interfaces.PartialLinearFluid.specificEnthalpy.....	721
Modelica.Media.Interfaces.PartialLinearFluid.specificEntropy.....	721
Modelica.Media.Interfaces.PartialLinearFluid.specificInternalEnergy.....	722
Modelica.Media.Interfaces.PartialLinearFluid.specificGibbsEnergy.....	722
Modelica.Media.Interfaces.PartialLinearFluid.specificHelmholtzEnergy.....	722
Modelica.Media.Interfaces.PartialLinearFluid.velocityOfSound.....	723
Modelica.Media.Interfaces.PartialLinearFluid.isentropicExponent.....	723
Modelica.Media.Interfaces.PartialLinearFluid.isentropicEnthalpy.....	723
Modelica.Media.Interfaces.PartialLinearFluid.specificHeatCapacityCp.....	724
Modelica.Media.Interfaces.PartialLinearFluid.specificHeatCapacityCv.....	724
Modelica.Media.Interfaces.PartialLinearFluid.isothermalCompressibility.....	724
Modelica.Media.Interfaces.PartialLinearFluid.isobaricExpansionCoefficient.....	724
Modelica.Media.Interfaces.PartialLinearFluid.density_derP_h.....	725
Modelica.Media.Interfaces.PartialLinearFluid.density_derh_p.....	725
Modelica.Media.Interfaces.PartialLinearFluid.density_derP_T.....	725
Modelica.Media.Interfaces.PartialLinearFluid.density_derT_p.....	725
Modelica.Media.Interfaces.PartialLinearFluid.molarMass.....	726
Modelica.Media.Interfaces.PartialLinearFluid.T_ph.....	726
Modelica.Media.Interfaces.PartialLinearFluid.T_ps.....	726
Modelica.Media.Interfaces.PartialMixtureMedium.....	727
Modelica.Media.Interfaces.PartialMixtureMedium.ThermodynamicState.....	730
Modelica.Media.Interfaces.PartialMixtureMedium.FluidConstants.....	730
Modelica.Media.Interfaces.PartialMixtureMedium.gasConstant.....	731
Modelica.Media.Interfaces.PartialMixtureMedium.moleToMassFractions.....	731
Modelica.Media.Interfaces.PartialMixtureMedium.massToMoleFractions.....	731
Modelica.Media.Interfaces.PartialCondensingGases.....	732
Modelica.Media.Interfaces.PartialCondensingGases.saturationPressure.....	735
Modelica.Media.Interfaces.PartialCondensingGases.enthalpyOfVaporization.....	735
Modelica.Media.Interfaces.PartialCondensingGases.enthalpyOfLiquid.....	735
Modelica.Media.Interfaces.PartialCondensingGases.enthalpyOfGas.....	736
Modelica.Media.Interfaces.PartialCondensingGases.enthalpyOfCondensingGas.....	736
Modelica.Media.Interfaces.PartialTwoPhaseMedium.....	736
Modelica.Media.Interfaces.PartialTwoPhaseMedium.FluidLimits.....	740
Modelica.Media.Interfaces.PartialTwoPhaseMedium.FluidConstants.....	741
Modelica.Media.Interfaces.PartialTwoPhaseMedium.ThermodynamicState.....	741
Modelica.Media.Interfaces.PartialTwoPhaseMedium.SaturationProperties.....	742
Modelica.Media.Interfaces.PartialTwoPhaseMedium.FixedPhase.....	742
Modelica.Media.Interfaces.PartialTwoPhaseMedium.BaseProperties.....	742
Modelica.Media.Interfaces.PartialTwoPhaseMedium.setDewState.....	742
Modelica.Media.Interfaces.PartialTwoPhaseMedium.setBubbleState.....	743
Modelica.Media.Interfaces.PartialTwoPhaseMedium.setState_dTX.....	743
Modelica.Media.Interfaces.PartialTwoPhaseMedium.setState_phX.....	743
Modelica.Media.Interfaces.PartialTwoPhaseMedium.setState_psX.....	744
Modelica.Media.Interfaces.PartialTwoPhaseMedium.setState_pTX.....	744
Modelica.Media.Interfaces.PartialTwoPhaseMedium.setSat_T.....	744
Modelica.Media.Interfaces.PartialTwoPhaseMedium.setSat_p.....	745
Modelica.Media.Interfaces.PartialTwoPhaseMedium.bubbleEnthalpy.....	745
Modelica.Media.Interfaces.PartialTwoPhaseMedium.dewEnthalpy.....	745
Modelica.Media.Interfaces.PartialTwoPhaseMedium.bubbleEntropy.....	746
Modelica.Media.Interfaces.PartialTwoPhaseMedium.dewEntropy.....	746
Modelica.Media.Interfaces.PartialTwoPhaseMedium.bubbleDensity.....	746
Modelica.Media.Interfaces.PartialTwoPhaseMedium.dewDensity.....	746

Modelica.Media.Interfaces.PartialTwoPhaseMedium.saturationPressure.....	747
Modelica.Media.Interfaces.PartialTwoPhaseMedium.saturationTemperature.....	747
Modelica.Media.Interfaces.PartialTwoPhaseMedium.saturationPressure_sat.....	747
Modelica.Media.Interfaces.PartialTwoPhaseMedium.saturationTemperature_sat.....	747
Modelica.Media.Interfaces.PartialTwoPhaseMedium.saturationTemperature_derP.....	748
Modelica.Media.Interfaces.PartialTwoPhaseMedium.saturationTemperature_derP_sat.....	748
Modelica.Media.Interfaces.PartialTwoPhaseMedium.surfaceTension.....	748
Modelica.Media.Interfaces.PartialTwoPhaseMedium.molarMass.....	749
Modelica.Media.Interfaces.PartialTwoPhaseMedium.dBubbleDensity_dPressure.....	749
Modelica.Media.Interfaces.PartialTwoPhaseMedium.dDewDensity_dPressure.....	749
Modelica.Media.Interfaces.PartialTwoPhaseMedium.dBubbleEnthalpy_dPressure.....	749
Modelica.Media.Interfaces.PartialTwoPhaseMedium.dDewEnthalpy_dPressure.....	750
Modelica.Media.Interfaces.PartialTwoPhaseMedium.specificEnthalpy_pTX.....	750
Modelica.Media.Interfaces.PartialTwoPhaseMedium.temperature_phX.....	750
Modelica.Media.Interfaces.PartialTwoPhaseMedium.density_phX.....	751
Modelica.Media.Interfaces.PartialTwoPhaseMedium.temperature_psX.....	751
Modelica.Media.Interfaces.PartialTwoPhaseMedium.density_psX.....	751
Modelica.Media.Interfaces.PartialTwoPhaseMedium.specificEnthalpy_psX.....	752
Modelica.Media.Interfaces.PartialTwoPhaseMedium.setState_pT.....	752
Modelica.Media.Interfaces.PartialTwoPhaseMedium.setState_ph.....	752
Modelica.Media.Interfaces.PartialTwoPhaseMedium.setState_ps.....	753
Modelica.Media.Interfaces.PartialTwoPhaseMedium.setState_dT.....	753
Modelica.Media.Interfaces.PartialTwoPhaseMedium.setState_px.....	753
Modelica.Media.Interfaces.PartialTwoPhaseMedium.setState_Tx.....	754
Modelica.Media.Interfaces.PartialTwoPhaseMedium.vapourQuality.....	754
Modelica.Media.Interfaces.PartialTwoPhaseMedium.density_ph.....	754
Modelica.Media.Interfaces.PartialTwoPhaseMedium.temperature_ph.....	755
Modelica.Media.Interfaces.PartialTwoPhaseMedium.pressure_dT.....	755
Modelica.Media.Interfaces.PartialTwoPhaseMedium.specificEnthalpy_dT.....	755
Modelica.Media.Interfaces.PartialTwoPhaseMedium.specificEnthalpy_ps.....	755
Modelica.Media.Interfaces.PartialTwoPhaseMedium.temperature_ps.....	756
Modelica.Media.Interfaces.PartialTwoPhaseMedium.density_ps.....	756
Modelica.Media.Interfaces.PartialTwoPhaseMedium.specificEnthalpy_pT.....	756
Modelica.Media.Interfaces.PartialTwoPhaseMedium.density_pT.....	757
Modelica.Media.Interfaces.PartialSimpleMedium.....	757
Modelica.Media.Interfaces.PartialSimpleMedium.ThermodynamicState.....	761
Modelica.Media.Interfaces.PartialSimpleMedium.BaseProperties.....	761
Modelica.Media.Interfaces.PartialSimpleMedium.setState_pTX.....	762
Modelica.Media.Interfaces.PartialSimpleMedium.setState_phX.....	762
Modelica.Media.Interfaces.PartialSimpleMedium.setState_psX.....	762
Modelica.Media.Interfaces.PartialSimpleMedium.setState_dTX.....	763
Modelica.Media.Interfaces.PartialSimpleMedium.dynamicViscosity.....	763
Modelica.Media.Interfaces.PartialSimpleMedium.thermalConductivity.....	763
Modelica.Media.Interfaces.PartialSimpleMedium.specificHeatCapacityCp.....	763
Modelica.Media.Interfaces.PartialSimpleMedium.specificHeatCapacityCv.....	764
Modelica.Media.Interfaces.PartialSimpleMedium.isentropicExponent.....	764
Modelica.Media.Interfaces.PartialSimpleMedium.velocityOfSound.....	764
Modelica.Media.Interfaces.PartialSimpleMedium.specificEnthalpy_pTX.....	765
Modelica.Media.Interfaces.PartialSimpleMedium.temperature_phX.....	765
Modelica.Media.Interfaces.PartialSimpleMedium.density_phX.....	765
Modelica.Media.Interfaces.PartialSimpleIdealGasMedium.....	765
Modelica.Media.Interfaces.PartialSimpleIdealGasMedium.ThermodynamicState.....	769
Modelica.Media.Interfaces.PartialSimpleIdealGasMedium.BaseProperties.....	769
Modelica.Media.Interfaces.PartialSimpleIdealGasMedium.setState_pTX.....	770
Modelica.Media.Interfaces.PartialSimpleIdealGasMedium.setState_phX.....	770
Modelica.Media.Interfaces.PartialSimpleIdealGasMedium.setState_psX.....	770
Modelica.Media.Interfaces.PartialSimpleIdealGasMedium.setState_dTX.....	771
Modelica.Media.Interfaces.PartialSimpleIdealGasMedium.pressure.....	771

---

Modelica.Media.Interfaces.PartialSimpleIdealGasMedium.temperature.....	771
Modelica.Media.Interfaces.PartialSimpleIdealGasMedium.density.....	772
Modelica.Media.Interfaces.PartialSimpleIdealGasMedium.specificEnthalpy.....	772
Modelica.Media.Interfaces.PartialSimpleIdealGasMedium.specificInternalEnergy.....	772
Modelica.Media.Interfaces.PartialSimpleIdealGasMedium.specificEntropy.....	772
Modelica.Media.Interfaces.PartialSimpleIdealGasMedium.specificGibbsEnergy.....	773
Modelica.Media.Interfaces.PartialSimpleIdealGasMedium.specificHelmholtzEnergy.....	773
Modelica.Media.Interfaces.PartialSimpleIdealGasMedium.dynamicViscosity.....	773
Modelica.Media.Interfaces.PartialSimpleIdealGasMedium.thermalConductivity.....	774
Modelica.Media.Interfaces.PartialSimpleIdealGasMedium.specificHeatCapacityCp.....	774
Modelica.Media.Interfaces.PartialSimpleIdealGasMedium.specificHeatCapacityCv.....	774
Modelica.Media.Interfaces.PartialSimpleIdealGasMedium.isentropicExponent.....	774
Modelica.Media.Interfaces.PartialSimpleIdealGasMedium.velocityOfSound.....	775
Modelica.Media.Interfaces.PartialSimpleIdealGasMedium.specificEnthalpy_pTX.....	775
Modelica.Media.Interfaces.PartialSimpleIdealGasMedium.temperature_phX.....	775
Modelica.Media.Interfaces.PartialSimpleIdealGasMedium.density_phX.....	776
Modelica.Media.Common.....	776
Modelica.Media.Common.SaturationProperties.....	782
Modelica.Media.Common.SaturationBoundaryProperties.....	782
Modelica.Media.Common.IF97BaseTwoPhase.....	782
Modelica.Media.Common.IF97PhaseBoundaryProperties.....	783
Modelica.Media.Common.GibbsDerivs.....	783
Modelica.Media.Common.HelmholtzDerivs.....	784
Modelica.Media.Common.TwoPhaseTransportProps.....	784
Modelica.Media.Common.PhaseBoundaryProperties.....	785
Modelica.Media.Common.NewtonDerivatives_ph.....	785
Modelica.Media.Common.NewtonDerivatives_ps.....	786
Modelica.Media.Common.NewtonDerivatives_pT.....	786
Modelica.Media.Common.ExtraDerivatives.....	786
Modelica.Media.Common.BridgmansTables.....	787
Modelica.Media.Common.gibbsToBridgmansTables.....	788
Modelica.Media.Common.helmholtzToBridgmansTables.....	788
Modelica.Media.Common.gibbsToBoundaryProps.....	788
Modelica.Media.Common.helmholtzToBoundaryProps.....	789
Modelica.Media.Common.cv2Phase.....	789
Modelica.Media.Common.cvdpT2Phase.....	789
Modelica.Media.Common.gibbsToExtraDerivs.....	790
Modelica.Media.Common.helmholtzToExtraDerivs.....	790
Modelica.Media.Common.Helmholtz_ph.....	790
Modelica.Media.Common.Helmholtz_pT.....	791
Modelica.Media.Common.Helmholtz_ps.....	791
Modelica.Media.Common.OneNonLinearEquation.....	791
Modelica.Media.Common.OneNonLinearEquation.f_nonlinear_Data.....	792
Modelica.Media.Common.OneNonLinearEquation.f_nonlinear.....	792
Modelica.Media.Common.OneNonLinearEquation.solve.....	793
Modelica.Media.Air.....	793
Modelica.Media.Air.SimpleAir.....	794
Modelica.Media.Air.DryAirNasa.....	798
Modelica.Media.Air.DryAirNasa.dynamicViscosity.....	802
Modelica.Media.Air.DryAirNasa.thermalConductivity.....	803
Modelica.Media.Air.MoistAir.....	803
Modelica.Media.Air.MoistAir.BaseProperties.....	809
Modelica.Media.Air.MoistAir.setState_pTX.....	809
Modelica.Media.Air.MoistAir.setState_phX.....	810
Modelica.Media.Air.MoistAir.setState_dTX.....	810
Modelica.Media.Air.MoistAir.Xsaturation.....	811
Modelica.Media.Air.MoistAir.xsaturation.....	811
Modelica.Media.Air.MoistAir.xsaturation_pT.....	811

Modelica.Media.Air.MoistAir.massFraction_pTphi.....	812
Modelica.Media.Air.MoistAir.relativeHumidity_pTX.....	812
Modelica.Media.Air.MoistAir.relativeHumidity.....	813
Modelica.Media.Air.MoistAir.gasConstant.....	813
Modelica.Media.Air.MoistAir.gasConstant_X.....	813
Modelica.Media.Air.MoistAir.saturationPressureLiquid.....	814
Modelica.Media.Air.MoistAir.saturationPressureLiquid_der.....	814
Modelica.Media.Air.MoistAir.sublimationPressureIce.....	814
Modelica.Media.Air.MoistAir.sublimationPressureIce_der.....	815
Modelica.Media.Air.MoistAir.saturationPressure.....	815
Modelica.Media.Air.MoistAir.saturationPressure_der.....	816
Modelica.Media.Air.MoistAir.saturationTemperature.....	816
Modelica.Media.Air.MoistAir.enthalpyOfVaporization.....	816
Modelica.Media.Air.MoistAir.HeatCapacityOfWater.....	817
Modelica.Media.Air.MoistAir.enthalpyOfLiquid.....	817
Modelica.Media.Air.MoistAir.enthalpyOfGas.....	818
Modelica.Media.Air.MoistAir.enthalpyOfCondensingGas.....	818
Modelica.Media.Air.MoistAir.enthalpyOfWater.....	818
Modelica.Media.Air.MoistAir.enthalpyOfWater_der.....	819
Modelica.Media.Air.MoistAir.pressure.....	819
Modelica.Media.Air.MoistAir.temperature.....	820
Modelica.Media.Air.MoistAir.T_phX.....	820
Modelica.Media.Air.MoistAir.density.....	820
Modelica.Media.Air.MoistAir.specificEnthalpy.....	821
Modelica.Media.Air.MoistAir.h_pTX.....	821
Modelica.Media.Air.MoistAir.h_pTX_der.....	821
Modelica.Media.Air.MoistAir.specificInternalEnergy.....	822
Modelica.Media.Air.MoistAir.specificInternalEnergy_pTX.....	822
Modelica.Media.Air.MoistAir.specificInternalEnergy_pTX_der.....	823
Modelica.Media.Air.MoistAir.specificEntropy.....	823
Modelica.Media.Air.MoistAir.specificGibbsEnergy.....	824
Modelica.Media.Air.MoistAir.specificHelmholtzEnergy.....	824
Modelica.Media.Air.MoistAir.specificHeatCapacityCp.....	824
Modelica.Media.Air.MoistAir.specificHeatCapacityCv.....	825
Modelica.Media.Air.MoistAir.dynamicViscosity.....	825
Modelica.Media.Air.MoistAir.thermalConductivity.....	826
Modelica.Media.Air.MoistAir.Utilities.....	826
Modelica.Media.Air.MoistAir.Utilities.spliceFunction.....	826
Modelica.Media.Air.MoistAir.Utilities.spliceFunction_der.....	826
Modelica.Media.Air.MoistAir.PsychrometricData.....	827
Modelica.Media.CompressibleLiquids.....	828
Modelica.Media.CompressibleLiquids.Common.....	829
Modelica.Media.CompressibleLiquids.Common.LinearWater_pT.....	829
Modelica.Media.CompressibleLiquids.LinearColdWater.....	833
Modelica.Media.CompressibleLiquids.LinearColdWater.dynamicViscosity.....	836
Modelica.Media.CompressibleLiquids.LinearColdWater.thermalConductivity.....	837
Modelica.Media.CompressibleLiquids.LinearWater_pT_Ambient.....	837
Modelica.Media.IdealGases.....	837
Modelica.Media.IdealGases.Common.....	841
Modelica.Media.IdealGases.Common.DataRecord.....	841
Modelica.Media.IdealGases.Common.SingleGasNasa.....	842
Modelica.Media.IdealGases.Common.SingleGasNasa.ThermodynamicState.....	847
Modelica.Media.IdealGases.Common.SingleGasNasa.FluidConstants.....	847
Modelica.Media.IdealGases.Common.SingleGasNasa.BaseProperties.....	848
Modelica.Media.IdealGases.Common.SingleGasNasa.setState_pTX.....	848
Modelica.Media.IdealGases.Common.SingleGasNasa.setState_phX.....	848
Modelica.Media.IdealGases.Common.SingleGasNasa.setState_psX.....	849
Modelica.Media.IdealGases.Common.SingleGasNasa.setState_dTX.....	849

Modelica.Media.IdealGases.Common.SingleGasNasa.pressure.....	849
Modelica.Media.IdealGases.Common.SingleGasNasa.temperature.....	850
Modelica.Media.IdealGases.Common.SingleGasNasa.density.....	850
Modelica.Media.IdealGases.Common.SingleGasNasa.specificEnthalpy.....	850
Modelica.Media.IdealGases.Common.SingleGasNasa.specificInternalEnergy.....	851
Modelica.Media.IdealGases.Common.SingleGasNasa.specificEntropy.....	851
Modelica.Media.IdealGases.Common.SingleGasNasa.specificGibbsEnergy.....	851
Modelica.Media.IdealGases.Common.SingleGasNasa.specificHelmholtzEnergy.....	851
Modelica.Media.IdealGases.Common.SingleGasNasa.specificHeatCapacityCp.....	852
Modelica.Media.IdealGases.Common.SingleGasNasa.specificHeatCapacityCv.....	852
Modelica.Media.IdealGases.Common.SingleGasNasa.isentropicExponent.....	852
Modelica.Media.IdealGases.Common.SingleGasNasa.velocityOfSound.....	852
Modelica.Media.IdealGases.Common.SingleGasNasa.isentropicEnthalpyApproximation.....	853
Modelica.Media.IdealGases.Common.SingleGasNasa.isentropicEnthalpy.....	853
Modelica.Media.IdealGases.Common.SingleGasNasa.isobaricExpansionCoefficient.....	854
Modelica.Media.IdealGases.Common.SingleGasNasa.isothermalCompressibility.....	854
Modelica.Media.IdealGases.Common.SingleGasNasa.density_derP_T.....	854
Modelica.Media.IdealGases.Common.SingleGasNasa.density_derT_p.....	854
Modelica.Media.IdealGases.Common.SingleGasNasa.density_derX.....	855
Modelica.Media.IdealGases.Common.SingleGasNasa.cp_T.....	855
Modelica.Media.IdealGases.Common.SingleGasNasa.cp_Tlow.....	855
Modelica.Media.IdealGases.Common.SingleGasNasa.cp_Tlow_der.....	856
Modelica.Media.IdealGases.Common.SingleGasNasa.h_T.....	856
Modelica.Media.IdealGases.Common.SingleGasNasa.h_T_der.....	856
Modelica.Media.IdealGases.Common.SingleGasNasa.h_Tlow.....	857
Modelica.Media.IdealGases.Common.SingleGasNasa.h_Tlow_der.....	857
Modelica.Media.IdealGases.Common.SingleGasNasa.s0_T.....	858
Modelica.Media.IdealGases.Common.SingleGasNasa.s0_Tlow.....	858
Modelica.Media.IdealGases.Common.SingleGasNasa.dynamicViscosityLowPressure.....	858
Modelica.Media.IdealGases.Common.SingleGasNasa.dynamicViscosity.....	859
Modelica.Media.IdealGases.Common.SingleGasNasa.thermalConductivityEstimate.....	859
Modelica.Media.IdealGases.Common.SingleGasNasa.thermalConductivity.....	860
Modelica.Media.IdealGases.Common.SingleGasNasa.molarMass.....	860
Modelica.Media.IdealGases.Common.SingleGasNasa.T_h.....	860
Modelica.Media.IdealGases.Common.SingleGasNasa.T_ps.....	861
Modelica.Media.IdealGases.Common.MixtureGasNasa.....	861
Modelica.Media.IdealGases.Common.MixtureGasNasa.BaseProperties.....	865
Modelica.Media.IdealGases.Common.MixtureGasNasa.setState_pTX.....	865
Modelica.Media.IdealGases.Common.MixtureGasNasa.setState_phX.....	866
Modelica.Media.IdealGases.Common.MixtureGasNasa.setState_psX.....	866
Modelica.Media.IdealGases.Common.MixtureGasNasa.setState_dTX.....	866
Modelica.Media.IdealGases.Common.MixtureGasNasa.pressure.....	866
Modelica.Media.IdealGases.Common.MixtureGasNasa.temperature.....	867
Modelica.Media.IdealGases.Common.MixtureGasNasa.density.....	867
Modelica.Media.IdealGases.Common.MixtureGasNasa.specificEnthalpy.....	867
Modelica.Media.IdealGases.Common.MixtureGasNasa.specificInternalEnergy.....	868
Modelica.Media.IdealGases.Common.MixtureGasNasa.specificEntropy.....	868
Modelica.Media.IdealGases.Common.MixtureGasNasa.specificGibbsEnergy.....	868
Modelica.Media.IdealGases.Common.MixtureGasNasa.specificHelmholtzEnergy.....	868
Modelica.Media.IdealGases.Common.MixtureGasNasa.h_TX.....	869
Modelica.Media.IdealGases.Common.MixtureGasNasa.h_TX_der.....	869
Modelica.Media.IdealGases.Common.MixtureGasNasa.gasConstant.....	870
Modelica.Media.IdealGases.Common.MixtureGasNasa.specificHeatCapacityCp.....	870
Modelica.Media.IdealGases.Common.MixtureGasNasa.specificHeatCapacityCv.....	870
Modelica.Media.IdealGases.Common.MixtureGasNasa.MixEntropy.....	870
Modelica.Media.IdealGases.Common.MixtureGasNasa.s_TX.....	871
Modelica.Media.IdealGases.Common.MixtureGasNasa.isentropicExponent.....	871
Modelica.Media.IdealGases.Common.MixtureGasNasa.velocityOfSound.....	871

Modelica.Media.IdealGases.Common.MixtureGasNasa.isentropicEnthalpyApproximation.....	872
Modelica.Media.IdealGases.Common.MixtureGasNasa.isentropicEnthalpy.....	872
Modelica.Media.IdealGases.Common.MixtureGasNasa.gasMixtureViscosity.....	872
Modelica.Media.IdealGases.Common.MixtureGasNasa.dynamicViscosity.....	873
Modelica.Media.IdealGases.Common.MixtureGasNasa.mixtureViscosityChung.....	873
Modelica.Media.IdealGases.Common.MixtureGasNasa.lowPressureThermalConductivity.....	874
Modelica.Media.IdealGases.Common.MixtureGasNasa.thermalConductivity.....	875
Modelica.Media.IdealGases.Common.MixtureGasNasa.isobaricExpansionCoefficient.....	875
Modelica.Media.IdealGases.Common.MixtureGasNasa.isoTHERMALCompressibility.....	875
Modelica.Media.IdealGases.Common.MixtureGasNasa.density_derP_T.....	876
Modelica.Media.IdealGases.Common.MixtureGasNasa.density_derT_p.....	876
Modelica.Media.IdealGases.Common.MixtureGasNasa.density_derX.....	876
Modelica.Media.IdealGases.Common.MixtureGasNasa.molarMass.....	876
Modelica.Media.IdealGases.Common.MixtureGasNasa.T_hX.....	877
Modelica.Media.IdealGases.Common.MixtureGasNasa.T_psX.....	877
Modelica.Media.IdealGases.Common.FluidData.....	877
Modelica.Media.IdealGases.Common.SingleGasesData.....	890
Modelica.Media.IdealGases.MixtureGases.....	1233
Modelica.Media.IdealGases.MixtureGases.CombustionAir.....	1234
Modelica.Media.IdealGases.MixtureGases.AirSteam.....	1234
Modelica.Media.IdealGases.MixtureGases.FlueGasLambdaOnePlus.....	1234
Modelica.Media.IdealGases.MixtureGases.FlueGasSixComponents.....	1234
Modelica.Media.IdealGases.MixtureGases.SimpleNaturalGas.....	1235
Modelica.Media.IdealGases.MixtureGases.SimpleNaturalGasFixedComposition.....	1235
Modelica.Media.IdealGases.SingleGases.....	1235
Modelica.Media.IdealGases.SingleGases.Ag.....	1272
Modelica.Media.IdealGases.SingleGases.Agplus.....	1272
Modelica.Media.IdealGases.SingleGases.Agminus.....	1272
Modelica.Media.IdealGases.SingleGases.Air.....	1272
Modelica.Media.IdealGases.SingleGases.AL.....	1273
Modelica.Media.IdealGases.SingleGases.ALplus.....	1273
Modelica.Media.IdealGases.SingleGases.ALminus.....	1273
Modelica.Media.IdealGases.SingleGases.ALBr.....	1273
Modelica.Media.IdealGases.SingleGases.ALBr2.....	1273
Modelica.Media.IdealGases.SingleGases.ALBr3.....	1274
Modelica.Media.IdealGases.SingleGases.ALC.....	1274
Modelica.Media.IdealGases.SingleGases.ALC2.....	1274
Modelica.Media.IdealGases.SingleGases.ALCL.....	1274
Modelica.Media.IdealGases.SingleGases.ALCLplus.....	1274
Modelica.Media.IdealGases.SingleGases.ALCL2.....	1275
Modelica.Media.IdealGases.SingleGases.ALCL3.....	1275
Modelica.Media.IdealGases.SingleGases.ALF.....	1275
Modelica.Media.IdealGases.SingleGases.ALFplus.....	1275
Modelica.Media.IdealGases.SingleGases.ALFCL.....	1275
Modelica.Media.IdealGases.SingleGases.ALFCL2.....	1276
Modelica.Media.IdealGases.SingleGases.ALF2.....	1276
Modelica.Media.IdealGases.SingleGases.ALF2minus.....	1276
Modelica.Media.IdealGases.SingleGases.ALF2CL.....	1276
Modelica.Media.IdealGases.SingleGases.ALF3.....	1276
Modelica.Media.IdealGases.SingleGases.ALF4minus.....	1277
Modelica.Media.IdealGases.SingleGases.ALH.....	1277
Modelica.Media.IdealGases.SingleGases.ALHCL.....	1277
Modelica.Media.IdealGases.SingleGases.ALHCL2.....	1277
Modelica.Media.IdealGases.SingleGases.ALHF.....	1277
Modelica.Media.IdealGases.SingleGases.ALHFCL.....	1278
Modelica.Media.IdealGases.SingleGases.ALHF2.....	1278
Modelica.Media.IdealGases.SingleGases.ALH2.....	1278
Modelica.Media.IdealGases.SingleGases.ALH2CL.....	1278

---

Modelica.Media.IdealGases.SingleGases.ALH2F .....	1278
Modelica.Media.IdealGases.SingleGases.ALH3 .....	1279
Modelica.Media.IdealGases.SingleGases.ALI .....	1279
Modelica.Media.IdealGases.SingleGases.ALI2 .....	1279
Modelica.Media.IdealGases.SingleGases.ALI3 .....	1279
Modelica.Media.IdealGases.SingleGases.ALN .....	1279
Modelica.Media.IdealGases.SingleGases.ALO .....	1280
Modelica.Media.IdealGases.SingleGases.AL_Oplus .....	1280
Modelica.Media.IdealGases.SingleGases.AL_Ominus .....	1280
Modelica.Media.IdealGases.SingleGases.ALOCL .....	1280
Modelica.Media.IdealGases.SingleGases.ALOCL2 .....	1280
Modelica.Media.IdealGases.SingleGases.ALOF .....	1281
Modelica.Media.IdealGases.SingleGases.ALOF2 .....	1281
Modelica.Media.IdealGases.SingleGases.ALOF2minus .....	1281
Modelica.Media.IdealGases.SingleGases.ALOH .....	1281
Modelica.Media.IdealGases.SingleGases.ALOHCL .....	1281
Modelica.Media.IdealGases.SingleGases.ALOHCL2 .....	1282
Modelica.Media.IdealGases.SingleGases.ALOHF .....	1282
Modelica.Media.IdealGases.SingleGases.ALOHF2 .....	1282
Modelica.Media.IdealGases.SingleGases.ALO2 .....	1282
Modelica.Media.IdealGases.SingleGases.ALO2minus .....	1283
Modelica.Media.IdealGases.SingleGases.AL_OH_2 .....	1283
Modelica.Media.IdealGases.SingleGases.AL_OH_2CL .....	1284
Modelica.Media.IdealGases.SingleGases.AL_OH_2F .....	1284
Modelica.Media.IdealGases.SingleGases.AL_OH_3 .....	1284
Modelica.Media.IdealGases.SingleGases.ALS .....	1285
Modelica.Media.IdealGases.SingleGases.ALS2 .....	1285
Modelica.Media.IdealGases.SingleGases.AL2 .....	1285
Modelica.Media.IdealGases.SingleGases.AL2Br6 .....	1285
Modelica.Media.IdealGases.SingleGases.AL2C2 .....	1285
Modelica.Media.IdealGases.SingleGases.AL2CL6 .....	1286
Modelica.Media.IdealGases.SingleGases.AL2F6 .....	1286
Modelica.Media.IdealGases.SingleGases.AL2I6 .....	1286
Modelica.Media.IdealGases.SingleGases.AL2O .....	1286
Modelica.Media.IdealGases.SingleGases.AL2Oplus .....	1286
Modelica.Media.IdealGases.SingleGases.AL2O2 .....	1287
Modelica.Media.IdealGases.SingleGases.AL2O2plus .....	1287
Modelica.Media.IdealGases.SingleGases.AL2O3 .....	1287
Modelica.Media.IdealGases.SingleGases.AL2S .....	1287
Modelica.Media.IdealGases.SingleGases.AL2S2 .....	1287
Modelica.Media.IdealGases.SingleGases.Ar .....	1288
Modelica.Media.IdealGases.SingleGases.Arplus .....	1288
Modelica.Media.IdealGases.SingleGases.B .....	1288
Modelica.Media.IdealGases.SingleGases.Bplus .....	1288
Modelica.Media.IdealGases.SingleGases.Bminus .....	1288
Modelica.Media.IdealGases.SingleGases.BBr .....	1289
Modelica.Media.IdealGases.SingleGases.BBr2 .....	1289
Modelica.Media.IdealGases.SingleGases.BBr3 .....	1290
Modelica.Media.IdealGases.SingleGases.BC .....	1290
Modelica.Media.IdealGases.SingleGases.BC2 .....	1291
Modelica.Media.IdealGases.SingleGases.BCL .....	1291
Modelica.Media.IdealGases.SingleGases.BCLplus .....	1291
Modelica.Media.IdealGases.SingleGases.BCLOH .....	1291
Modelica.Media.IdealGases.SingleGases.BCL_OH_2 .....	1292
Modelica.Media.IdealGases.SingleGases.BCL2 .....	1292
Modelica.Media.IdealGases.SingleGases.BCL2plus .....	1292
Modelica.Media.IdealGases.SingleGases.BCL2OH .....	1292
Modelica.Media.IdealGases.SingleGases.BF .....	1292

Modelica.Media.IdealGases.SingleGases.BFCL.....	1293
Modelica.Media.IdealGases.SingleGases.BFCL2.....	1293
Modelica.Media.IdealGases.SingleGases.BFOH.....	1293
Modelica.Media.IdealGases.SingleGases.BF_OH_2.....	1293
Modelica.Media.IdealGases.SingleGases.BF2.....	1293
Modelica.Media.IdealGases.SingleGases.BF2plus.....	1294
Modelica.Media.IdealGases.SingleGases.BF2minus.....	1294
Modelica.Media.IdealGases.SingleGases.BF2CL.....	1294
Modelica.Media.IdealGases.SingleGases.BF2OH.....	1294
Modelica.Media.IdealGases.SingleGases.BF3.....	1294
Modelica.Media.IdealGases.SingleGases.BF4minus.....	1295
Modelica.Media.IdealGases.SingleGases.BH.....	1295
Modelica.Media.IdealGases.SingleGases.BHCL.....	1295
Modelica.Media.IdealGases.SingleGases.BHCL2.....	1295
Modelica.Media.IdealGases.SingleGases.BHF.....	1295
Modelica.Media.IdealGases.SingleGases.BHFCL.....	1296
Modelica.Media.IdealGases.SingleGases.BHF2.....	1296
Modelica.Media.IdealGases.SingleGases.BH2.....	1296
Modelica.Media.IdealGases.SingleGases.BH2CL.....	1296
Modelica.Media.IdealGases.SingleGases.BH2F.....	1296
Modelica.Media.IdealGases.SingleGases.BH3.....	1297
Modelica.Media.IdealGases.SingleGases.BH3NH3.....	1297
Modelica.Media.IdealGases.SingleGases.BH4.....	1297
Modelica.Media.IdealGases.SingleGases.BI.....	1297
Modelica.Media.IdealGases.SingleGases.BI2.....	1297
Modelica.Media.IdealGases.SingleGases.BI3.....	1298
Modelica.Media.IdealGases.SingleGases.BN.....	1298
Modelica.Media.IdealGases.SingleGases.BO.....	1298
Modelica.Media.IdealGases.SingleGases.BOminus.....	1298
Modelica.Media.IdealGases.SingleGases.BOCL.....	1298
Modelica.Media.IdealGases.SingleGases.BOCL2.....	1299
Modelica.Media.IdealGases.SingleGases.BOF.....	1299
Modelica.Media.IdealGases.SingleGases.BOF2.....	1299
Modelica.Media.IdealGases.SingleGases.BOH.....	1299
Modelica.Media.IdealGases.SingleGases.BO2.....	1299
Modelica.Media.IdealGases.SingleGases.BO2minus.....	1300
Modelica.Media.IdealGases.SingleGases.B_OH_2.....	1300
Modelica.Media.IdealGases.SingleGases.BS.....	1300
Modelica.Media.IdealGases.SingleGases.BS2.....	1300
Modelica.Media.IdealGases.SingleGases.B2.....	1300
Modelica.Media.IdealGases.SingleGases.B2C.....	1301
Modelica.Media.IdealGases.SingleGases.B2CL4.....	1301
Modelica.Media.IdealGases.SingleGases.B2F4.....	1301
Modelica.Media.IdealGases.SingleGases.B2H.....	1302
Modelica.Media.IdealGases.SingleGases.B2H2.....	1302
Modelica.Media.IdealGases.SingleGases.B2H3.....	1303
Modelica.Media.IdealGases.SingleGases.B2H3_db.....	1303
Modelica.Media.IdealGases.SingleGases.B2H4.....	1303
Modelica.Media.IdealGases.SingleGases.B2H4_db.....	1303
Modelica.Media.IdealGases.SingleGases.B2H5.....	1304
Modelica.Media.IdealGases.SingleGases.B2H5_db.....	1304
Modelica.Media.IdealGases.SingleGases.B2H6.....	1304
Modelica.Media.IdealGases.SingleGases.B2O.....	1304
Modelica.Media.IdealGases.SingleGases.B2O2.....	1304
Modelica.Media.IdealGases.SingleGases.B2O3.....	1305
Modelica.Media.IdealGases.SingleGases.B2_OH_4.....	1305
Modelica.Media.IdealGases.SingleGases.B2S.....	1305
Modelica.Media.IdealGases.SingleGases.B2S2.....	1306

---

Modelica.Media.IdealGases.SingleGases.B2S3.....	1306
Modelica.Media.IdealGases.SingleGases.B3H7_C2v.....	1307
Modelica.Media.IdealGases.SingleGases.B3H7_Cs.....	1307
Modelica.Media.IdealGases.SingleGases.B3H9.....	1307
Modelica.Media.IdealGases.SingleGases.B3N3H6.....	1308
Modelica.Media.IdealGases.SingleGases.B3O3CL3.....	1308
Modelica.Media.IdealGases.SingleGases.B3O3FCL2.....	1308
Modelica.Media.IdealGases.SingleGases.B3O3F2CL.....	1308
Modelica.Media.IdealGases.SingleGases.B3O3F3.....	1308
Modelica.Media.IdealGases.SingleGases.B4H4.....	1309
Modelica.Media.IdealGases.SingleGases.B4H10.....	1309
Modelica.Media.IdealGases.SingleGases.B4H12.....	1309
Modelica.Media.IdealGases.SingleGases.B5H9.....	1310
Modelica.Media.IdealGases.SingleGases.Ba.....	1310
Modelica.Media.IdealGases.SingleGases.Baplus.....	1311
Modelica.Media.IdealGases.SingleGases.BaBr.....	1311
Modelica.Media.IdealGases.SingleGases.BaBr2.....	1312
Modelica.Media.IdealGases.SingleGases.BaCL.....	1312
Modelica.Media.IdealGases.SingleGases.BaCLplus.....	1312
Modelica.Media.IdealGases.SingleGases.BaCL2.....	1312
Modelica.Media.IdealGases.SingleGases.BaF.....	1312
Modelica.Media.IdealGases.SingleGases.BaFplus.....	1313
Modelica.Media.IdealGases.SingleGases.BaF2.....	1313
Modelica.Media.IdealGases.SingleGases.BaH.....	1313
Modelica.Media.IdealGases.SingleGases.BaI.....	1313
Modelica.Media.IdealGases.SingleGases.BaI2.....	1313
Modelica.Media.IdealGases.SingleGases.BaO.....	1314
Modelica.Media.IdealGases.SingleGases.BaOplus.....	1314
Modelica.Media.IdealGases.SingleGases.BaOH.....	1315
Modelica.Media.IdealGases.SingleGases.BaOHplus.....	1315
Modelica.Media.IdealGases.SingleGases.Ba_OH_2.....	1316
Modelica.Media.IdealGases.SingleGases.BaS.....	1316
Modelica.Media.IdealGases.SingleGases.Ba2.....	1316
Modelica.Media.IdealGases.SingleGases.Be.....	1316
Modelica.Media.IdealGases.SingleGases.Beplus.....	1316
Modelica.Media.IdealGases.SingleGases.Beplusplus.....	1317
Modelica.Media.IdealGases.SingleGases.BeBr.....	1317
Modelica.Media.IdealGases.SingleGases.BeBr2.....	1317
Modelica.Media.IdealGases.SingleGases.BeCL.....	1317
Modelica.Media.IdealGases.SingleGases.BeCL2.....	1318
Modelica.Media.IdealGases.SingleGases.BeF.....	1318
Modelica.Media.IdealGases.SingleGases.BeF2.....	1319
Modelica.Media.IdealGases.SingleGases.BeH.....	1319
Modelica.Media.IdealGases.SingleGases.BeHplus.....	1320
Modelica.Media.IdealGases.SingleGases.BeH2.....	1320
Modelica.Media.IdealGases.SingleGases.Bel.....	1320
Modelica.Media.IdealGases.SingleGases.Bel2.....	1320
Modelica.Media.IdealGases.SingleGases.BeN.....	1320
Modelica.Media.IdealGases.SingleGases.BeO.....	1321
Modelica.Media.IdealGases.SingleGases.BeOH.....	1321
Modelica.Media.IdealGases.SingleGases.BeOHplus.....	1321
Modelica.Media.IdealGases.SingleGases.Be_OH_2.....	1321
Modelica.Media.IdealGases.SingleGases.BeS.....	1321
Modelica.Media.IdealGases.SingleGases.Be2.....	1322
Modelica.Media.IdealGases.SingleGases.Be2CL4.....	1322
Modelica.Media.IdealGases.SingleGases.Be2F4.....	1322
Modelica.Media.IdealGases.SingleGases.Be2O.....	1322
Modelica.Media.IdealGases.SingleGases.Be2OF2.....	1322

Modelica.Media.IdealGases.SingleGases.Be2O2.....	1323
Modelica.Media.IdealGases.SingleGases.Be3O3.....	1323
Modelica.Media.IdealGases.SingleGases.Be4O4.....	1323
Modelica.Media.IdealGases.SingleGases.Br.....	1323
Modelica.Media.IdealGases.SingleGases.Brplus.....	1323
Modelica.Media.IdealGases.SingleGases.Brminus.....	1324
Modelica.Media.IdealGases.SingleGases.BrCL.....	1324
Modelica.Media.IdealGases.SingleGases.BrF.....	1324
Modelica.Media.IdealGases.SingleGases.BrF3.....	1324
Modelica.Media.IdealGases.SingleGases.BrF5.....	1325
Modelica.Media.IdealGases.SingleGases.BrO.....	1325
Modelica.Media.IdealGases.SingleGases.OMgO.....	1326
Modelica.Media.IdealGases.SingleGases.BrOO.....	1326
Modelica.Media.IdealGases.SingleGases.BrO3.....	1327
Modelica.Media.IdealGases.SingleGases.Br2.....	1327
Modelica.Media.IdealGases.SingleGases.BrBrO.....	1327
Modelica.Media.IdealGases.SingleGases.BrOBr.....	1327
Modelica.Media.IdealGases.SingleGases.C.....	1327
Modelica.Media.IdealGases.SingleGases.Cplus.....	1328
Modelica.Media.IdealGases.SingleGases.Cminus.....	1328
Modelica.Media.IdealGases.SingleGases.CBr.....	1328
Modelica.Media.IdealGases.SingleGases.CBr2.....	1328
Modelica.Media.IdealGases.SingleGases.CBr3.....	1329
Modelica.Media.IdealGases.SingleGases.CBr4.....	1329
Modelica.Media.IdealGases.SingleGases.CCL.....	1330
Modelica.Media.IdealGases.SingleGases.CCL2.....	1330
Modelica.Media.IdealGases.SingleGases.CCL2Br2.....	1331
Modelica.Media.IdealGases.SingleGases.CCL3.....	1331
Modelica.Media.IdealGases.SingleGases.CCL3Br.....	1331
Modelica.Media.IdealGases.SingleGases.CCL4.....	1331
Modelica.Media.IdealGases.SingleGases.CF.....	1331
Modelica.Media.IdealGases.SingleGases.CFplus.....	1332
Modelica.Media.IdealGases.SingleGases.CFBr3.....	1332
Modelica.Media.IdealGases.SingleGases.CFCL.....	1332
Modelica.Media.IdealGases.SingleGases.CFCLBr2.....	1332
Modelica.Media.IdealGases.SingleGases.CFCL2.....	1332
Modelica.Media.IdealGases.SingleGases.CFCL2Br.....	1333
Modelica.Media.IdealGases.SingleGases.CFCL3.....	1333
Modelica.Media.IdealGases.SingleGases.CF2.....	1333
Modelica.Media.IdealGases.SingleGases.CF2plus.....	1333
Modelica.Media.IdealGases.SingleGases.CF2Br2.....	1333
Modelica.Media.IdealGases.SingleGases.CF2CL.....	1334
Modelica.Media.IdealGases.SingleGases.CF2CLBr.....	1334
Modelica.Media.IdealGases.SingleGases.CF2CL2.....	1334
Modelica.Media.IdealGases.SingleGases.CF3.....	1334
Modelica.Media.IdealGases.SingleGases.CF3plus.....	1334
Modelica.Media.IdealGases.SingleGases.CF3Br.....	1335
Modelica.Media.IdealGases.SingleGases.CF3CL.....	1335
Modelica.Media.IdealGases.SingleGases.CF4.....	1335
Modelica.Media.IdealGases.SingleGases.CHplus.....	1336
Modelica.Media.IdealGases.SingleGases.CHBr3.....	1336
Modelica.Media.IdealGases.SingleGases.CHCL.....	1337
Modelica.Media.IdealGases.SingleGases.CHCLBr2.....	1337
Modelica.Media.IdealGases.SingleGases.CHCL2.....	1338
Modelica.Media.IdealGases.SingleGases.CHCL2Br.....	1338
Modelica.Media.IdealGases.SingleGases.CHCL3.....	1338
Modelica.Media.IdealGases.SingleGases.CHF.....	1338
Modelica.Media.IdealGases.SingleGases.CHFBr2.....	1338

---

Modelica.Media.IdealGases.SingleGases.CHFCL	1339
Modelica.Media.IdealGases.SingleGases.CHFCLBr	1339
Modelica.Media.IdealGases.SingleGases.CHFCL2	1339
Modelica.Media.IdealGases.SingleGases.CHF2	1339
Modelica.Media.IdealGases.SingleGases.CHF2Br	1339
Modelica.Media.IdealGases.SingleGases.CHF2CL	1340
Modelica.Media.IdealGases.SingleGases.CHF3	1340
Modelica.Media.IdealGases.SingleGases.CHI3	1341
Modelica.Media.IdealGases.SingleGases.CH2	1341
Modelica.Media.IdealGases.SingleGases.CH2Br2	1342
Modelica.Media.IdealGases.SingleGases.CH2CL	1342
Modelica.Media.IdealGases.SingleGases.CH2CLBr	1342
Modelica.Media.IdealGases.SingleGases.CH2CL2	1342
Modelica.Media.IdealGases.SingleGases.CH2F	1342
Modelica.Media.IdealGases.SingleGases.CH2FBr	1343
Modelica.Media.IdealGases.SingleGases.CH2FCL	1343
Modelica.Media.IdealGases.SingleGases.CH2F2	1343
Modelica.Media.IdealGases.SingleGases.CH2I2	1343
Modelica.Media.IdealGases.SingleGases.CH3	1343
Modelica.Media.IdealGases.SingleGases.CH3Br	1344
Modelica.Media.IdealGases.SingleGases.CH3CL	1344
Modelica.Media.IdealGases.SingleGases.CH3F	1344
Modelica.Media.IdealGases.SingleGases.CH3I	1344
Modelica.Media.IdealGases.SingleGases.CH2OH	1344
Modelica.Media.IdealGases.SingleGases.CH2OHplus	1345
Modelica.Media.IdealGases.SingleGases.CH3O	1345
Modelica.Media.IdealGases.SingleGases.CH4	1345
Modelica.Media.IdealGases.SingleGases.CH3OH	1345
Modelica.Media.IdealGases.SingleGases.CH3OOH	1345
Modelica.Media.IdealGases.SingleGases.Cl	1346
Modelica.Media.IdealGases.SingleGases.Cl2	1346
Modelica.Media.IdealGases.SingleGases.Cl3	1346
Modelica.Media.IdealGases.SingleGases.Cl4	1347
Modelica.Media.IdealGases.SingleGases.CN	1347
Modelica.Media.IdealGases.SingleGases.CNplus	1348
Modelica.Media.IdealGases.SingleGases.CNminus	1348
Modelica.Media.IdealGases.SingleGases.CNN	1348
Modelica.Media.IdealGases.SingleGases.CO	1349
Modelica.Media.IdealGases.SingleGases.COplus	1349
Modelica.Media.IdealGases.SingleGases.COCL	1349
Modelica.Media.IdealGases.SingleGases.COCL2	1349
Modelica.Media.IdealGases.SingleGases.COFCL	1349
Modelica.Media.IdealGases.SingleGases.COF2	1350
Modelica.Media.IdealGases.SingleGases.COHCL	1350
Modelica.Media.IdealGases.SingleGases.COHF	1350
Modelica.Media.IdealGases.SingleGases.COS	1350
Modelica.Media.IdealGases.SingleGases.CO2	1350
Modelica.Media.IdealGases.SingleGases.CO2plus	1351
Modelica.Media.IdealGases.SingleGases.COOH	1351
Modelica.Media.IdealGases.SingleGases.CP	1351
Modelica.Media.IdealGases.SingleGases.CS	1351
Modelica.Media.IdealGases.SingleGases.CS2	1351
Modelica.Media.IdealGases.SingleGases.C2	1352
Modelica.Media.IdealGases.SingleGases.C2plus	1352
Modelica.Media.IdealGases.SingleGases.C2minus	1352
Modelica.Media.IdealGases.SingleGases.C2CL	1352
Modelica.Media.IdealGases.SingleGases.C2CL2	1352
Modelica.Media.IdealGases.SingleGases.C2CL3	1353

Modelica.Media.IdealGases.SingleGases.C2CL4.....	1353
Modelica.Media.IdealGases.SingleGases.C2CL6.....	1353
Modelica.Media.IdealGases.SingleGases.C2F.....	1353
Modelica.Media.IdealGases.SingleGases.C2FCL.....	1353
Modelica.Media.IdealGases.SingleGases.C2FCL3.....	1354
Modelica.Media.IdealGases.SingleGases.C2F2.....	1354
Modelica.Media.IdealGases.SingleGases.C2F2CL2.....	1354
Modelica.Media.IdealGases.SingleGases.C2F3.....	1354
Modelica.Media.IdealGases.SingleGases.C2F3CL.....	1354
Modelica.Media.IdealGases.SingleGases.C2F4.....	1355
Modelica.Media.IdealGases.SingleGases.C2F6.....	1355
Modelica.Media.IdealGases.SingleGases.C2H.....	1355
Modelica.Media.IdealGases.SingleGases.C2HCL.....	1355
Modelica.Media.IdealGases.SingleGases.C2HCL3.....	1355
Modelica.Media.IdealGases.SingleGases.C2HF.....	1356
Modelica.Media.IdealGases.SingleGases.C2HFCL2.....	1356
Modelica.Media.IdealGases.SingleGases.C2HF2CL.....	1356
Modelica.Media.IdealGases.SingleGases.C2HF3.....	1356
Modelica.Media.IdealGases.SingleGases.C2H2_vinylidene.....	1356
Modelica.Media.IdealGases.SingleGases.C2H2CL2.....	1357
Modelica.Media.IdealGases.SingleGases.C2H2FCL.....	1357
Modelica.Media.IdealGases.SingleGases.C2H2F2.....	1357
Modelica.Media.IdealGases.SingleGases.CH2CO_ketene.....	1357
Modelica.Media.IdealGases.SingleGases.O_CH_2O.....	1357
Modelica.Media.IdealGases.SingleGases.HO_CO_2OH.....	1358
Modelica.Media.IdealGases.SingleGases.C2H3_vinyl.....	1358
Modelica.Media.IdealGases.SingleGases.CH2BrminusCOOH.....	1358
Modelica.Media.IdealGases.SingleGases.C2H3CL.....	1358
Modelica.Media.IdealGases.SingleGases.CH2CLminusCOOH.....	1358
Modelica.Media.IdealGases.SingleGases.C2H3F.....	1359
Modelica.Media.IdealGases.SingleGases.CH3CN.....	1359
Modelica.Media.IdealGases.SingleGases.CH3CO_acetyl.....	1359
Modelica.Media.IdealGases.SingleGases.C2H4.....	1359
Modelica.Media.IdealGases.SingleGases.C2H4O_ethyleng_o.....	1359
Modelica.Media.IdealGases.SingleGases.CH3CHO_ethanal.....	1360
Modelica.Media.IdealGases.SingleGases.CH3COOH.....	1360
Modelica.Media.IdealGases.SingleGases.OHCH2COOH.....	1360
Modelica.Media.IdealGases.SingleGases.C2H5.....	1360
Modelica.Media.IdealGases.SingleGases.C2H5Br.....	1360
Modelica.Media.IdealGases.SingleGases.C2H6.....	1361
Modelica.Media.IdealGases.SingleGases.CH3N2CH3.....	1361
Modelica.Media.IdealGases.SingleGases.C2H5OH.....	1361
Modelica.Media.IdealGases.SingleGases.CH3OCH3.....	1361
Modelica.Media.IdealGases.SingleGases.CH3O2CH3.....	1361
Modelica.Media.IdealGases.SingleGases.CCN.....	1362
Modelica.Media.IdealGases.SingleGases.CNC.....	1362
Modelica.Media.IdealGases.SingleGases.OCCN.....	1362
Modelica.Media.IdealGases.SingleGases.C2N2.....	1362
Modelica.Media.IdealGases.SingleGases.C2O.....	1362
Modelica.Media.IdealGases.SingleGases.C3.....	1363
Modelica.Media.IdealGases.SingleGases.C3H3_1_propynl.....	1363
Modelica.Media.IdealGases.SingleGases.C3H3_2_propynl.....	1363
Modelica.Media.IdealGases.SingleGases.C3H4_allene.....	1363
Modelica.Media.IdealGases.SingleGases.C3H4_propyne.....	1363
Modelica.Media.IdealGases.SingleGases.C3H4_cyclo.....	1364
Modelica.Media.IdealGases.SingleGases.C3H5_allyl.....	1364
Modelica.Media.IdealGases.SingleGases.C3H6_propylene.....	1364
Modelica.Media.IdealGases.SingleGases.C3H6_cyclo.....	1364

Modelica.Media.IdealGases.SingleGases.C3H6O_propylox.....	1364
Modelica.Media.IdealGases.SingleGases.C3H6O_acetone.....	1365
Modelica.Media.IdealGases.SingleGases.C3H6O_propanal.....	1365
Modelica.Media.IdealGases.SingleGases.C3H7_n_propyl.....	1365
Modelica.Media.IdealGases.SingleGases.C3H7_i_propyl.....	1365
Modelica.Media.IdealGases.SingleGases.C3H8.....	1365
Modelica.Media.IdealGases.SingleGases.C3H8O_1propanol.....	1366
Modelica.Media.IdealGases.SingleGases.C3H8O_2propanol.....	1366
Modelica.Media.IdealGases.SingleGases.CNCOCN.....	1366
Modelica.Media.IdealGases.SingleGases.C3O2.....	1366
Modelica.Media.IdealGases.SingleGases.C4.....	1366
Modelica.Media.IdealGases.SingleGases.C4H2_butadiyne.....	1367
Modelica.Media.IdealGases.SingleGases.C4H4_1_3minuscyclo.....	1367
Modelica.Media.IdealGases.SingleGases.C4H6_butadiene.....	1367
Modelica.Media.IdealGases.SingleGases.C4H6_1butyne.....	1367
Modelica.Media.IdealGases.SingleGases.C4H6_2butyne.....	1367
Modelica.Media.IdealGases.SingleGases.C4H6_cyclo.....	1368
Modelica.Media.IdealGases.SingleGases.C4H8_1_butene.....	1368
Modelica.Media.IdealGases.SingleGases.C4H8_cis2_buten.....	1368
Modelica.Media.IdealGases.SingleGases.C4H8_isobutene.....	1368
Modelica.Media.IdealGases.SingleGases.C4H8_cyclo.....	1368
Modelica.Media.IdealGases.SingleGases.C4H9_n_butyl.....	1369
Modelica.Media.IdealGases.SingleGases.C4H9_i_butyl.....	1369
Modelica.Media.IdealGases.SingleGases.C4H9_s_butyl.....	1369
Modelica.Media.IdealGases.SingleGases.C4H9_t_butyl.....	1369
Modelica.Media.IdealGases.SingleGases.C4H10_n_butane.....	1369
Modelica.Media.IdealGases.SingleGases.C4H10_isobutane.....	1370
Modelica.Media.IdealGases.SingleGases.C4N2.....	1370
Modelica.Media.IdealGases.SingleGases.C5.....	1370
Modelica.Media.IdealGases.SingleGases.C5H6_1_3cyclo.....	1370
Modelica.Media.IdealGases.SingleGases.C5H8_cyclo.....	1370
Modelica.Media.IdealGases.SingleGases.C5H10_1_pentene.....	1371
Modelica.Media.IdealGases.SingleGases.C5H10_cyclo.....	1371
Modelica.Media.IdealGases.SingleGases.C5H11_pentyl.....	1371
Modelica.Media.IdealGases.SingleGases.C5H11_t_pentyl.....	1371
Modelica.Media.IdealGases.SingleGases.C5H12_n_pentane.....	1371
Modelica.Media.IdealGases.SingleGases.C5H12_i_pentane.....	1372
Modelica.Media.IdealGases.SingleGases.CH3C_CH3_2CH3.....	1372
Modelica.Media.IdealGases.SingleGases.C6D5_phenyl.....	1372
Modelica.Media.IdealGases.SingleGases.C6D6.....	1372
Modelica.Media.IdealGases.SingleGases.C6H2.....	1372
Modelica.Media.IdealGases.SingleGases.C6H5_phenyl.....	1373
Modelica.Media.IdealGases.SingleGases.C6H5O_phenoxy.....	1373
Modelica.Media.IdealGases.SingleGases.C6H6.....	1373
Modelica.Media.IdealGases.SingleGases.C6H5OH_phenol.....	1373
Modelica.Media.IdealGases.SingleGases.C6H10_cyclo.....	1373
Modelica.Media.IdealGases.SingleGases.C6H12_1_hexene.....	1374
Modelica.Media.IdealGases.SingleGases.C6H12_cyclo.....	1374
Modelica.Media.IdealGases.SingleGases.C6H13_n_hexyl.....	1374
Modelica.Media.IdealGases.SingleGases.C6H14_n_hexane.....	1374
Modelica.Media.IdealGases.SingleGases.C7H7_benzyl.....	1375
Modelica.Media.IdealGases.SingleGases.C7H8.....	1375
Modelica.Media.IdealGases.SingleGases.C7H8O_cresol_mx.....	1376
Modelica.Media.IdealGases.SingleGases.C7H14_1_heptene.....	1376
Modelica.Media.IdealGases.SingleGases.C7H15_n_heptyl.....	1377
Modelica.Media.IdealGases.SingleGases.C7H16_n_heptane.....	1377
Modelica.Media.IdealGases.SingleGases.C7H16_2_methylh.....	1377
Modelica.Media.IdealGases.SingleGases.C8H8_styrene.....	1377

Modelica.Media.IdealGases.SingleGases.C8H10_ethylbenz.....	1377
Modelica.Media.IdealGases.SingleGases.C8H16_1_octene.....	1378
Modelica.Media.IdealGases.SingleGases.C8H17_n_octyl.....	1378
Modelica.Media.IdealGases.SingleGases.C8H18_n_octane.....	1378
Modelica.Media.IdealGases.SingleGases.C8H18_isooctane.....	1378
Modelica.Media.IdealGases.SingleGases.C9H19_n_nonyl.....	1378
Modelica.Media.IdealGases.SingleGases.C10H8_naphthale.....	1379
Modelica.Media.IdealGases.SingleGases.C10H21_n_decyl.....	1379
Modelica.Media.IdealGases.SingleGases.C12H9_o_bipheny.....	1379
Modelica.Media.IdealGases.SingleGases.C12H10_biphenyl.....	1379
Modelica.Media.IdealGases.SingleGases.Ca.....	1379
Modelica.Media.IdealGases.SingleGases.Cplus.....	1380
Modelica.Media.IdealGases.SingleGases.CaBr.....	1380
Modelica.Media.IdealGases.SingleGases.CaBr2.....	1380
Modelica.Media.IdealGases.SingleGases.CaCL.....	1380
Modelica.Media.IdealGases.SingleGases.CaCLplus.....	1380
Modelica.Media.IdealGases.SingleGases.CaCL2.....	1381
Modelica.Media.IdealGases.SingleGases.CaF.....	1381
Modelica.Media.IdealGases.SingleGases.CaFplus.....	1381
Modelica.Media.IdealGases.SingleGases.CaF2.....	1381
Modelica.Media.IdealGases.SingleGases.CaH.....	1381
Modelica.Media.IdealGases.SingleGases.Cal.....	1382
Modelica.Media.IdealGases.SingleGases.Cal2.....	1382
Modelica.Media.IdealGases.SingleGases.CaO.....	1382
Modelica.Media.IdealGases.SingleGases.CaOplus.....	1382
Modelica.Media.IdealGases.SingleGases.CaOH.....	1382
Modelica.Media.IdealGases.SingleGases.CaOHplus.....	1383
Modelica.Media.IdealGases.SingleGases.Ca_OH_2.....	1383
Modelica.Media.IdealGases.SingleGases.CaS.....	1383
Modelica.Media.IdealGases.SingleGases.Ca2.....	1383
Modelica.Media.IdealGases.SingleGases.Cd.....	1383
Modelica.Media.IdealGases.SingleGases.Cdplus.....	1384
Modelica.Media.IdealGases.SingleGases.CL.....	1384
Modelica.Media.IdealGases.SingleGases.CLplus.....	1384
Modelica.Media.IdealGases.SingleGases.CLminus.....	1384
Modelica.Media.IdealGases.SingleGases.CLCN.....	1384
Modelica.Media.IdealGases.SingleGases.CLF.....	1385
Modelica.Media.IdealGases.SingleGases.CLF3.....	1385
Modelica.Media.IdealGases.SingleGases.CLF5.....	1385
Modelica.Media.IdealGases.SingleGases.CLO.....	1385
Modelica.Media.IdealGases.SingleGases.CLO2.....	1385
Modelica.Media.IdealGases.SingleGases.CL2.....	1386
Modelica.Media.IdealGases.SingleGases.CL2O.....	1386
Modelica.Media.IdealGases.SingleGases.Co.....	1386
Modelica.Media.IdealGases.SingleGases.Coplus.....	1386
Modelica.Media.IdealGases.SingleGases.Cominus.....	1386
Modelica.Media.IdealGases.SingleGases.Cr.....	1387
Modelica.Media.IdealGases.SingleGases.Crplus.....	1387
Modelica.Media.IdealGases.SingleGases.Crminus.....	1387
Modelica.Media.IdealGases.SingleGases.CrN.....	1387
Modelica.Media.IdealGases.SingleGases.CrO.....	1387
Modelica.Media.IdealGases.SingleGases.CrO2.....	1388
Modelica.Media.IdealGases.SingleGases.CrO3.....	1388
Modelica.Media.IdealGases.SingleGases.CrO3minus.....	1388
Modelica.Media.IdealGases.SingleGases.Cs.....	1388
Modelica.Media.IdealGases.SingleGases.Csplus.....	1388
Modelica.Media.IdealGases.SingleGases.Csminus.....	1389
Modelica.Media.IdealGases.SingleGases.CsBO2.....	1389

---

Modelica.Media.IdealGases.SingleGases.CsBr.....	1389
Modelica.Media.IdealGases.SingleGases.CsCL.....	1389
Modelica.Media.IdealGases.SingleGases.CsF.....	1389
Modelica.Media.IdealGases.SingleGases.CsH.....	1390
Modelica.Media.IdealGases.SingleGases.CsI.....	1390
Modelica.Media.IdealGases.SingleGases.CsLi.....	1390
Modelica.Media.IdealGases.SingleGases.CsNO2.....	1390
Modelica.Media.IdealGases.SingleGases.CsNO3.....	1390
Modelica.Media.IdealGases.SingleGases.CsNa.....	1391
Modelica.Media.IdealGases.SingleGases.CsO.....	1391
Modelica.Media.IdealGases.SingleGases.CsOH.....	1391
Modelica.Media.IdealGases.SingleGases.CsRb.....	1391
Modelica.Media.IdealGases.SingleGases.Cs2.....	1391
Modelica.Media.IdealGases.SingleGases.Cs2Br2.....	1392
Modelica.Media.IdealGases.SingleGases.Cs2CO3.....	1392
Modelica.Media.IdealGases.SingleGases.Cs2CL2.....	1392
Modelica.Media.IdealGases.SingleGases.Cs2F2.....	1392
Modelica.Media.IdealGases.SingleGases.Cs2I2.....	1392
Modelica.Media.IdealGases.SingleGases.Cs2O.....	1393
Modelica.Media.IdealGases.SingleGases.Cs2Oplus.....	1393
Modelica.Media.IdealGases.SingleGases.Cs2O2.....	1393
Modelica.Media.IdealGases.SingleGases.Cs2O2H2.....	1393
Modelica.Media.IdealGases.SingleGases.Cs2SO4.....	1393
Modelica.Media.IdealGases.SingleGases.Cu.....	1394
Modelica.Media.IdealGases.SingleGases.Cuplus.....	1394
Modelica.Media.IdealGases.SingleGases.Cuminus.....	1394
Modelica.Media.IdealGases.SingleGases.CuCL.....	1394
Modelica.Media.IdealGases.SingleGases.CuF.....	1394
Modelica.Media.IdealGases.SingleGases.CuF2.....	1395
Modelica.Media.IdealGases.SingleGases.CuO.....	1395
Modelica.Media.IdealGases.SingleGases.Cu2.....	1395
Modelica.Media.IdealGases.SingleGases.Cu3CL3.....	1395
Modelica.Media.IdealGases.SingleGases.D.....	1395
Modelica.Media.IdealGases.SingleGases.Dplus.....	1396
Modelica.Media.IdealGases.SingleGases.Dminus.....	1396
Modelica.Media.IdealGases.SingleGases.DBr.....	1396
Modelica.Media.IdealGases.SingleGases.DCL.....	1396
Modelica.Media.IdealGases.SingleGases.DF.....	1396
Modelica.Media.IdealGases.SingleGases.DOCL.....	1397
Modelica.Media.IdealGases.SingleGases.DO2.....	1397
Modelica.Media.IdealGases.SingleGases.DO2minus.....	1397
Modelica.Media.IdealGases.SingleGases.D2.....	1397
Modelica.Media.IdealGases.SingleGases.D2plus.....	1397
Modelica.Media.IdealGases.SingleGases.D2minus.....	1398
Modelica.Media.IdealGases.SingleGases.D2O.....	1398
Modelica.Media.IdealGases.SingleGases.D2O2.....	1399
Modelica.Media.IdealGases.SingleGases.D2S.....	1399
Modelica.Media.IdealGases.SingleGases.eminus.....	1400
Modelica.Media.IdealGases.SingleGases.F.....	1400
Modelica.Media.IdealGases.SingleGases.Fplus.....	1400
Modelica.Media.IdealGases.SingleGases.Fminus.....	1401
Modelica.Media.IdealGases.SingleGases.FCN.....	1401
Modelica.Media.IdealGases.SingleGases.FCO.....	1401
Modelica.Media.IdealGases.SingleGases.FO.....	1401
Modelica.Media.IdealGases.SingleGases.FO2_FOO.....	1401
Modelica.Media.IdealGases.SingleGases.FO2_OFO.....	1402
Modelica.Media.IdealGases.SingleGases.F2.....	1402
Modelica.Media.IdealGases.SingleGases.F2O.....	1402

Modelica.Media.IdealGases.SingleGases.F2O2.....	1402
Modelica.Media.IdealGases.SingleGases.FS2F.....	1402
Modelica.Media.IdealGases.SingleGases.Fe.....	1403
Modelica.Media.IdealGases.SingleGases.Feplus.....	1403
Modelica.Media.IdealGases.SingleGases.Fe_CO_5.....	1403
Modelica.Media.IdealGases.SingleGases.FeCL.....	1403
Modelica.Media.IdealGases.SingleGases.FeCL2.....	1403
Modelica.Media.IdealGases.SingleGases.FeCL3.....	1404
Modelica.Media.IdealGases.SingleGases.FeO.....	1404
Modelica.Media.IdealGases.SingleGases.Fe_OH_2.....	1404
Modelica.Media.IdealGases.SingleGases.Fe2CL4.....	1404
Modelica.Media.IdealGases.SingleGases.Fe2CL6.....	1404
Modelica.Media.IdealGases.SingleGases.Ga.....	1405
Modelica.Media.IdealGases.SingleGases.Gaplus.....	1405
Modelica.Media.IdealGases.SingleGases.GaBr.....	1405
Modelica.Media.IdealGases.SingleGases.GaBr2.....	1405
Modelica.Media.IdealGases.SingleGases.GaBr3.....	1405
Modelica.Media.IdealGases.SingleGases.GaCL.....	1406
Modelica.Media.IdealGases.SingleGases.GaCL2.....	1406
Modelica.Media.IdealGases.SingleGases.GaCL3.....	1406
Modelica.Media.IdealGases.SingleGases.GaF.....	1406
Modelica.Media.IdealGases.SingleGases.GaF2.....	1406
Modelica.Media.IdealGases.SingleGases.GaF3.....	1407
Modelica.Media.IdealGases.SingleGases.GaH.....	1407
Modelica.Media.IdealGases.SingleGases.GaI.....	1407
Modelica.Media.IdealGases.SingleGases.GaI2.....	1407
Modelica.Media.IdealGases.SingleGases.GaI3.....	1407
Modelica.Media.IdealGases.SingleGases.GaO.....	1408
Modelica.Media.IdealGases.SingleGases.GaOH.....	1408
Modelica.Media.IdealGases.SingleGases.Ga2Br2.....	1408
Modelica.Media.IdealGases.SingleGases.Ga2Br4.....	1408
Modelica.Media.IdealGases.SingleGases.Ga2Br6.....	1408
Modelica.Media.IdealGases.SingleGases.Ga2CL2.....	1409
Modelica.Media.IdealGases.SingleGases.Ga2CL4.....	1409
Modelica.Media.IdealGases.SingleGases.Ga2CL6.....	1409
Modelica.Media.IdealGases.SingleGases.Ga2F2.....	1409
Modelica.Media.IdealGases.SingleGases.Ga2F4.....	1409
Modelica.Media.IdealGases.SingleGases.Ga2F6.....	1410
Modelica.Media.IdealGases.SingleGases.Ga2I2.....	1410
Modelica.Media.IdealGases.SingleGases.Ga2I4.....	1410
Modelica.Media.IdealGases.SingleGases.Ga2I6.....	1410
Modelica.Media.IdealGases.SingleGases.Ga2O.....	1410
Modelica.Media.IdealGases.SingleGases.Ge.....	1411
Modelica.Media.IdealGases.SingleGases.Geplus.....	1411
Modelica.Media.IdealGases.SingleGases.Geminus.....	1411
Modelica.Media.IdealGases.SingleGases.GeBr.....	1411
Modelica.Media.IdealGases.SingleGases.GeBr2.....	1411
Modelica.Media.IdealGases.SingleGases.GeBr3.....	1412
Modelica.Media.IdealGases.SingleGases.GeBr4.....	1412
Modelica.Media.IdealGases.SingleGases.GeCL.....	1412
Modelica.Media.IdealGases.SingleGases.GeCL2.....	1412
Modelica.Media.IdealGases.SingleGases.GeCL3.....	1412
Modelica.Media.IdealGases.SingleGases.GeCL4.....	1413
Modelica.Media.IdealGases.SingleGases.GeF.....	1413
Modelica.Media.IdealGases.SingleGases.GeF2.....	1413
Modelica.Media.IdealGases.SingleGases.GeF3.....	1413
Modelica.Media.IdealGases.SingleGases.GeF4.....	1414
Modelica.Media.IdealGases.SingleGases.GeH4.....	1414

---

Modelica.Media.IdealGases.SingleGases.Gel.....	1415
Modelica.Media.IdealGases.SingleGases.GeO.....	1415
Modelica.Media.IdealGases.SingleGases.GeO2.....	1415
Modelica.Media.IdealGases.SingleGases.GeS.....	1416
Modelica.Media.IdealGases.SingleGases.GeS2.....	1416
Modelica.Media.IdealGases.SingleGases.Ge2.....	1416
Modelica.Media.IdealGases.SingleGases.H.....	1416
Modelica.Media.IdealGases.SingleGases.Hplus.....	1416
Modelica.Media.IdealGases.SingleGases.Hminus.....	1417
Modelica.Media.IdealGases.SingleGases.HALO.....	1417
Modelica.Media.IdealGases.SingleGases.HALO2.....	1417
Modelica.Media.IdealGases.SingleGases.HBO.....	1417
Modelica.Media.IdealGases.SingleGases.HB0plus.....	1417
Modelica.Media.IdealGases.SingleGases.HBO2.....	1418
Modelica.Media.IdealGases.SingleGases.HBS.....	1418
Modelica.Media.IdealGases.SingleGases.HBSplus.....	1418
Modelica.Media.IdealGases.SingleGases.HCN.....	1418
Modelica.Media.IdealGases.SingleGases.HCO.....	1418
Modelica.Media.IdealGases.SingleGases.HCOplus.....	1419
Modelica.Media.IdealGases.SingleGases.HCCN.....	1419
Modelica.Media.IdealGases.SingleGases.HCCO.....	1419
Modelica.Media.IdealGases.SingleGases.HCL.....	1419
Modelica.Media.IdealGases.SingleGases.HD.....	1419
Modelica.Media.IdealGases.SingleGases.HDplus.....	1420
Modelica.Media.IdealGases.SingleGases.HDO.....	1420
Modelica.Media.IdealGases.SingleGases.HDO2.....	1420
Modelica.Media.IdealGases.SingleGases.HF.....	1420
Modelica.Media.IdealGases.SingleGases.HI.....	1420
Modelica.Media.IdealGases.SingleGases.HNC.....	1421
Modelica.Media.IdealGases.SingleGases.HNCO.....	1421
Modelica.Media.IdealGases.SingleGases.HNO.....	1421
Modelica.Media.IdealGases.SingleGases.HNO2.....	1421
Modelica.Media.IdealGases.SingleGases.HNO3.....	1421
Modelica.Media.IdealGases.SingleGases.HOCL.....	1422
Modelica.Media.IdealGases.SingleGases.HOF.....	1422
Modelica.Media.IdealGases.SingleGases.HO2.....	1422
Modelica.Media.IdealGases.SingleGases.HO2minus.....	1422
Modelica.Media.IdealGases.SingleGases.HPO.....	1422
Modelica.Media.IdealGases.SingleGases.HSO3F.....	1423
Modelica.Media.IdealGases.SingleGases.H2.....	1423
Modelica.Media.IdealGases.SingleGases.H2plus.....	1423
Modelica.Media.IdealGases.SingleGases.H2minus.....	1423
Modelica.Media.IdealGases.SingleGases.HBOH.....	1423
Modelica.Media.IdealGases.SingleGases.HCOOH.....	1424
Modelica.Media.IdealGases.SingleGases.H2F2.....	1424
Modelica.Media.IdealGases.SingleGases.H2O.....	1424
Modelica.Media.IdealGases.SingleGases.H2Oplus.....	1424
Modelica.Media.IdealGases.SingleGases.H2O2.....	1424
Modelica.Media.IdealGases.SingleGases.H2S.....	1425
Modelica.Media.IdealGases.SingleGases.H2SO4.....	1425
Modelica.Media.IdealGases.SingleGases.H2BOH.....	1425
Modelica.Media.IdealGases.SingleGases.HB_OH_2.....	1425
Modelica.Media.IdealGases.SingleGases.H3BO3.....	1425
Modelica.Media.IdealGases.SingleGases.H3B3O3.....	1426
Modelica.Media.IdealGases.SingleGases.H3B3O6.....	1426
Modelica.Media.IdealGases.SingleGases.H3F3.....	1426
Modelica.Media.IdealGases.SingleGases.H3Oplus.....	1426
Modelica.Media.IdealGases.SingleGases.H4F4.....	1426

---

Modelica.Media.IdealGases.SingleGases.H5F5.....	1427
Modelica.Media.IdealGases.SingleGases.H6F6.....	1427
Modelica.Media.IdealGases.SingleGases.H7F7.....	1427
Modelica.Media.IdealGases.SingleGases.He.....	1427
Modelica.Media.IdealGases.SingleGases.Heplus.....	1428
Modelica.Media.IdealGases.SingleGases.Hg.....	1428
Modelica.Media.IdealGases.SingleGases.Hgplus.....	1429
Modelica.Media.IdealGases.SingleGases.HgBr2.....	1429
Modelica.Media.IdealGases.SingleGases.I.....	1430
Modelica.Media.IdealGases.SingleGases.Iplus.....	1430
Modelica.Media.IdealGases.SingleGases.Iminus.....	1430
Modelica.Media.IdealGases.SingleGases.IF5.....	1431
Modelica.Media.IdealGases.SingleGases.IF7.....	1431
Modelica.Media.IdealGases.SingleGases.I2.....	1431
Modelica.Media.IdealGases.SingleGases.In.....	1431
Modelica.Media.IdealGases.SingleGases.Inplus.....	1431
Modelica.Media.IdealGases.SingleGases.InBr.....	1432
Modelica.Media.IdealGases.SingleGases.InBr2.....	1432
Modelica.Media.IdealGases.SingleGases.InBr3.....	1432
Modelica.Media.IdealGases.SingleGases.InCL.....	1432
Modelica.Media.IdealGases.SingleGases.InCL2.....	1432
Modelica.Media.IdealGases.SingleGases.InCL3.....	1433
Modelica.Media.IdealGases.SingleGases.InF.....	1433
Modelica.Media.IdealGases.SingleGases.InF2.....	1433
Modelica.Media.IdealGases.SingleGases.InF3.....	1433
Modelica.Media.IdealGases.SingleGases.InH.....	1433
Modelica.Media.IdealGases.SingleGases.InI.....	1434
Modelica.Media.IdealGases.SingleGases.InI2.....	1434
Modelica.Media.IdealGases.SingleGases.InI3.....	1434
Modelica.Media.IdealGases.SingleGases.InO.....	1434
Modelica.Media.IdealGases.SingleGases.InOH.....	1434
Modelica.Media.IdealGases.SingleGases.In2Br2.....	1435
Modelica.Media.IdealGases.SingleGases.In2Br4.....	1435
Modelica.Media.IdealGases.SingleGases.In2Br6.....	1435
Modelica.Media.IdealGases.SingleGases.In2CL2.....	1435
Modelica.Media.IdealGases.SingleGases.In2CL4.....	1435
Modelica.Media.IdealGases.SingleGases.In2CL6.....	1436
Modelica.Media.IdealGases.SingleGases.In2F2.....	1436
Modelica.Media.IdealGases.SingleGases.In2F4.....	1436
Modelica.Media.IdealGases.SingleGases.In2F6.....	1436
Modelica.Media.IdealGases.SingleGases.In2I2.....	1436
Modelica.Media.IdealGases.SingleGases.In2I4.....	1437
Modelica.Media.IdealGases.SingleGases.In2I6.....	1437
Modelica.Media.IdealGases.SingleGases.In2O.....	1437
Modelica.Media.IdealGases.SingleGases.K.....	1437
Modelica.Media.IdealGases.SingleGases.Kplus.....	1437
Modelica.Media.IdealGases.SingleGases.Kminus.....	1438
Modelica.Media.IdealGases.SingleGases.KALF4.....	1438
Modelica.Media.IdealGases.SingleGases.KBO2.....	1438
Modelica.Media.IdealGases.SingleGases.KBr.....	1438
Modelica.Media.IdealGases.SingleGases.KCN.....	1438
Modelica.Media.IdealGases.SingleGases.KCL.....	1439
Modelica.Media.IdealGases.SingleGases.KF.....	1439
Modelica.Media.IdealGases.SingleGases.KH.....	1439
Modelica.Media.IdealGases.SingleGases.KI.....	1439
Modelica.Media.IdealGases.SingleGases.KLi.....	1439
Modelica.Media.IdealGases.SingleGases.KNO2.....	1440
Modelica.Media.IdealGases.SingleGases.KNO3.....	1440

Modelica.Media.IdealGases.SingleGases.KNa.....	1440
Modelica.Media.IdealGases.SingleGases.KO.....	1440
Modelica.Media.IdealGases.SingleGases.KOH.....	1440
Modelica.Media.IdealGases.SingleGases.K2.....	1441
Modelica.Media.IdealGases.SingleGases.K2plus.....	1441
Modelica.Media.IdealGases.SingleGases.K2Br2.....	1441
Modelica.Media.IdealGases.SingleGases.K2CO3.....	1441
Modelica.Media.IdealGases.SingleGases.K2C2N2.....	1441
Modelica.Media.IdealGases.SingleGases.K2CL2.....	1442
Modelica.Media.IdealGases.SingleGases.K2F2.....	1442
Modelica.Media.IdealGases.SingleGases.K2I2.....	1442
Modelica.Media.IdealGases.SingleGases.K2O.....	1442
Modelica.Media.IdealGases.SingleGases.K2Oplus.....	1442
Modelica.Media.IdealGases.SingleGases.K2O2.....	1443
Modelica.Media.IdealGases.SingleGases.K2O2H2.....	1443
Modelica.Media.IdealGases.SingleGases.K2SO4.....	1443
Modelica.Media.IdealGases.SingleGases.Kr.....	1443
Modelica.Media.IdealGases.SingleGases.Krplus.....	1443
Modelica.Media.IdealGases.SingleGases.Li.....	1444
Modelica.Media.IdealGases.SingleGases.Liplus.....	1444
Modelica.Media.IdealGases.SingleGases.Liminus.....	1444
Modelica.Media.IdealGases.SingleGases.LiALF4.....	1444
Modelica.Media.IdealGases.SingleGases.LiBO2.....	1444
Modelica.Media.IdealGases.SingleGases.LiBr.....	1445
Modelica.Media.IdealGases.SingleGases.LiCL.....	1445
Modelica.Media.IdealGases.SingleGases.LiF.....	1445
Modelica.Media.IdealGases.SingleGases.LiH.....	1445
Modelica.Media.IdealGases.SingleGases.LiI.....	1445
Modelica.Media.IdealGases.SingleGases.LiN.....	1446
Modelica.Media.IdealGases.SingleGases.LiNO2.....	1446
Modelica.Media.IdealGases.SingleGases.LiNO3.....	1446
Modelica.Media.IdealGases.SingleGases.LiO.....	1446
Modelica.Media.IdealGases.SingleGases.LiOF.....	1446
Modelica.Media.IdealGases.SingleGases.LiOH.....	1447
Modelica.Media.IdealGases.SingleGases.LiON.....	1447
Modelica.Media.IdealGases.SingleGases.Li2.....	1447
Modelica.Media.IdealGases.SingleGases.Li2plus.....	1447
Modelica.Media.IdealGases.SingleGases.Li2Br2.....	1447
Modelica.Media.IdealGases.SingleGases.Li2F2.....	1448
Modelica.Media.IdealGases.SingleGases.Li2I2.....	1448
Modelica.Media.IdealGases.SingleGases.Li2O.....	1448
Modelica.Media.IdealGases.SingleGases.Li2Oplus.....	1448
Modelica.Media.IdealGases.SingleGases.Li2O2.....	1448
Modelica.Media.IdealGases.SingleGases.Li2O2H2.....	1449
Modelica.Media.IdealGases.SingleGases.Li2SO4.....	1449
Modelica.Media.IdealGases.SingleGases.Li3plus.....	1449
Modelica.Media.IdealGases.SingleGases.Li3Br3.....	1449
Modelica.Media.IdealGases.SingleGases.Li3CL3.....	1449
Modelica.Media.IdealGases.SingleGases.Li3F3.....	1450
Modelica.Media.IdealGases.SingleGases.Li3I3.....	1450
Modelica.Media.IdealGases.SingleGases.Mg.....	1450
Modelica.Media.IdealGases.SingleGases.Mgplus.....	1450
Modelica.Media.IdealGases.SingleGases.MgBr.....	1450
Modelica.Media.IdealGases.SingleGases.MgBr2.....	1451
Modelica.Media.IdealGases.SingleGases.MgCL.....	1451
Modelica.Media.IdealGases.SingleGases.MgCLplus.....	1451
Modelica.Media.IdealGases.SingleGases.MgCL2.....	1451
Modelica.Media.IdealGases.SingleGases.MgF.....	1451

Modelica.Media.IdealGases.SingleGases.MgFplus.....	1452
Modelica.Media.IdealGases.SingleGases.MgF2.....	1452
Modelica.Media.IdealGases.SingleGases.MgF2plus.....	1452
Modelica.Media.IdealGases.SingleGases.MgH.....	1452
Modelica.Media.IdealGases.SingleGases.Mgl.....	1452
Modelica.Media.IdealGases.SingleGases.Mgl2.....	1453
Modelica.Media.IdealGases.SingleGases.MgN.....	1453
Modelica.Media.IdealGases.SingleGases.MgO.....	1453
Modelica.Media.IdealGases.SingleGases.MgOH.....	1453
Modelica.Media.IdealGases.SingleGases.MgOHplus.....	1453
Modelica.Media.IdealGases.SingleGases.Mg_OH_2.....	1454
Modelica.Media.IdealGases.SingleGases.MgS.....	1454
Modelica.Media.IdealGases.SingleGases.Mg2.....	1454
Modelica.Media.IdealGases.SingleGases.Mg2F4.....	1454
Modelica.Media.IdealGases.SingleGases.Mn.....	1454
Modelica.Media.IdealGases.SingleGases.Mnplus.....	1455
Modelica.Media.IdealGases.SingleGases.Mo.....	1455
Modelica.Media.IdealGases.SingleGases.Moplus.....	1455
Modelica.Media.IdealGases.SingleGases.Mominus.....	1455
Modelica.Media.IdealGases.SingleGases.MoO.....	1455
Modelica.Media.IdealGases.SingleGases.MoO2.....	1456
Modelica.Media.IdealGases.SingleGases.MoO3.....	1456
Modelica.Media.IdealGases.SingleGases.MoO3minus.....	1456
Modelica.Media.IdealGases.SingleGases.Mo2O6.....	1456
Modelica.Media.IdealGases.SingleGases.Mo3O9.....	1456
Modelica.Media.IdealGases.SingleGases.Mo4O12.....	1457
Modelica.Media.IdealGases.SingleGases.Mo5O15.....	1457
Modelica.Media.IdealGases.SingleGases.N.....	1457
Modelica.Media.IdealGases.SingleGases.Nplus.....	1457
Modelica.Media.IdealGases.SingleGases.Nminus.....	1457
Modelica.Media.IdealGases.SingleGases.NCO.....	1458
Modelica.Media.IdealGases.SingleGases.ND.....	1458
Modelica.Media.IdealGases.SingleGases.ND2.....	1458
Modelica.Media.IdealGases.SingleGases.ND3.....	1458
Modelica.Media.IdealGases.SingleGases.NF.....	1458
Modelica.Media.IdealGases.SingleGases.NF2.....	1459
Modelica.Media.IdealGases.SingleGases.NF3.....	1459
Modelica.Media.IdealGases.SingleGases.NH.....	1459
Modelica.Media.IdealGases.SingleGases.NHplus.....	1459
Modelica.Media.IdealGases.SingleGases.NHF.....	1459
Modelica.Media.IdealGases.SingleGases.NHF2.....	1460
Modelica.Media.IdealGases.SingleGases.NH2.....	1460
Modelica.Media.IdealGases.SingleGases.NH2F.....	1460
Modelica.Media.IdealGases.SingleGases.NH3.....	1460
Modelica.Media.IdealGases.SingleGases.NH2OH.....	1460
Modelica.Media.IdealGases.SingleGases.NH4plus.....	1461
Modelica.Media.IdealGases.SingleGases.NO.....	1461
Modelica.Media.IdealGases.SingleGases.NOCL.....	1461
Modelica.Media.IdealGases.SingleGases.NOF.....	1461
Modelica.Media.IdealGases.SingleGases.NOF3.....	1461
Modelica.Media.IdealGases.SingleGases.NO2.....	1462
Modelica.Media.IdealGases.SingleGases.NO2minus.....	1462
Modelica.Media.IdealGases.SingleGases.NO2CL.....	1462
Modelica.Media.IdealGases.SingleGases.NO2F.....	1462
Modelica.Media.IdealGases.SingleGases.NO3.....	1462
Modelica.Media.IdealGases.SingleGases.NO3minus.....	1463
Modelica.Media.IdealGases.SingleGases.NO3F.....	1463
Modelica.Media.IdealGases.SingleGases.N2.....	1463

---

Modelica.Media.IdealGases.SingleGases.N2plus.....	1463
Modelica.Media.IdealGases.SingleGases.N2minus.....	1463
Modelica.Media.IdealGases.SingleGases.NCN.....	1464
Modelica.Media.IdealGases.SingleGases.N2D2_cis.....	1464
Modelica.Media.IdealGases.SingleGases.N2F2.....	1464
Modelica.Media.IdealGases.SingleGases.N2F4.....	1464
Modelica.Media.IdealGases.SingleGases.N2H2.....	1464
Modelica.Media.IdealGases.SingleGases.NH2NO2.....	1465
Modelica.Media.IdealGases.SingleGases.N2H4.....	1465
Modelica.Media.IdealGases.SingleGases.N2O.....	1465
Modelica.Media.IdealGases.SingleGases.N2Oplus.....	1466
Modelica.Media.IdealGases.SingleGases.N2O3.....	1466
Modelica.Media.IdealGases.SingleGases.N2O4.....	1467
Modelica.Media.IdealGases.SingleGases.N2O5.....	1467
Modelica.Media.IdealGases.SingleGases.N3.....	1467
Modelica.Media.IdealGases.SingleGases.N3H.....	1468
Modelica.Media.IdealGases.SingleGases.Na.....	1468
Modelica.Media.IdealGases.SingleGases.Naplus.....	1468
Modelica.Media.IdealGases.SingleGases.Naminus.....	1468
Modelica.Media.IdealGases.SingleGases.NaALF4.....	1468
Modelica.Media.IdealGases.SingleGases.NaBO2.....	1469
Modelica.Media.IdealGases.SingleGases.NaBr.....	1469
Modelica.Media.IdealGases.SingleGases.NaCN.....	1469
Modelica.Media.IdealGases.SingleGases.NaCL.....	1469
Modelica.Media.IdealGases.SingleGases.NaF.....	1469
Modelica.Media.IdealGases.SingleGases.NaH.....	1470
Modelica.Media.IdealGases.SingleGases.NaI.....	1470
Modelica.Media.IdealGases.SingleGases.NaLi.....	1470
Modelica.Media.IdealGases.SingleGases.NaNO2.....	1470
Modelica.Media.IdealGases.SingleGases.NaNO3.....	1470
Modelica.Media.IdealGases.SingleGases.NaO.....	1471
Modelica.Media.IdealGases.SingleGases.NaOH.....	1471
Modelica.Media.IdealGases.SingleGases.NaOHplus.....	1471
Modelica.Media.IdealGases.SingleGases.Na2.....	1471
Modelica.Media.IdealGases.SingleGases.Na2Br2.....	1471
Modelica.Media.IdealGases.SingleGases.Na2CL2.....	1472
Modelica.Media.IdealGases.SingleGases.Na2F2.....	1472
Modelica.Media.IdealGases.SingleGases.Na2I2.....	1472
Modelica.Media.IdealGases.SingleGases.Na2O.....	1472
Modelica.Media.IdealGases.SingleGases.Na2Oplus.....	1472
Modelica.Media.IdealGases.SingleGases.Na2O2.....	1473
Modelica.Media.IdealGases.SingleGases.Na2O2H2.....	1473
Modelica.Media.IdealGases.SingleGases.Na2SO4.....	1473
Modelica.Media.IdealGases.SingleGases.Na3CL3.....	1473
Modelica.Media.IdealGases.SingleGases.Na3F3.....	1473
Modelica.Media.IdealGases.SingleGases.Nb.....	1474
Modelica.Media.IdealGases.SingleGases.Nbplus.....	1474
Modelica.Media.IdealGases.SingleGases.Nbminus.....	1474
Modelica.Media.IdealGases.SingleGases.NbCL5.....	1474
Modelica.Media.IdealGases.SingleGases.NbO.....	1474
Modelica.Media.IdealGases.SingleGases.NbOCL3.....	1475
Modelica.Media.IdealGases.SingleGases.NbO2.....	1475
Modelica.Media.IdealGases.SingleGases.Ne.....	1475
Modelica.Media.IdealGases.SingleGases.Neplus.....	1476
Modelica.Media.IdealGases.SingleGases.Ni.....	1476
Modelica.Media.IdealGases.SingleGases.Niplus.....	1477
Modelica.Media.IdealGases.SingleGases.Niminus.....	1477
Modelica.Media.IdealGases.SingleGases.NiCL.....	1478

Modelica.Media.IdealGases.SingleGases.NiCL2.....	1478
Modelica.Media.IdealGases.SingleGases.NiO.....	1478
Modelica.Media.IdealGases.SingleGases.NiS.....	1478
Modelica.Media.IdealGases.SingleGases.O.....	1478
Modelica.Media.IdealGases.SingleGases.Oplus.....	1479
Modelica.Media.IdealGases.SingleGases.Ominus.....	1479
Modelica.Media.IdealGases.SingleGases.OD.....	1479
Modelica.Media.IdealGases.SingleGases.ODminus.....	1479
Modelica.Media.IdealGases.SingleGases.OH.....	1479
Modelica.Media.IdealGases.SingleGases.OHplus.....	1480
Modelica.Media.IdealGases.SingleGases.OHminus.....	1480
Modelica.Media.IdealGases.SingleGases.O2.....	1480
Modelica.Media.IdealGases.SingleGases.O2plus.....	1480
Modelica.Media.IdealGases.SingleGases.O2minus.....	1480
Modelica.Media.IdealGases.SingleGases.O3.....	1481
Modelica.Media.IdealGases.SingleGases.P.....	1481
Modelica.Media.IdealGases.SingleGases.Pplus.....	1481
Modelica.Media.IdealGases.SingleGases.Pminus.....	1481
Modelica.Media.IdealGases.SingleGases.PCL.....	1481
Modelica.Media.IdealGases.SingleGases.PCL2.....	1482
Modelica.Media.IdealGases.SingleGases.PCL2minus.....	1482
Modelica.Media.IdealGases.SingleGases.PCL3.....	1482
Modelica.Media.IdealGases.SingleGases.PCL5.....	1482
Modelica.Media.IdealGases.SingleGases.PF.....	1482
Modelica.Media.IdealGases.SingleGases.PFplus.....	1483
Modelica.Media.IdealGases.SingleGases.PFminus.....	1483
Modelica.Media.IdealGases.SingleGases.PFCL.....	1483
Modelica.Media.IdealGases.SingleGases.PFCLminus.....	1483
Modelica.Media.IdealGases.SingleGases.PFCL2.....	1483
Modelica.Media.IdealGases.SingleGases.PFCL4.....	1484
Modelica.Media.IdealGases.SingleGases.PF2.....	1484
Modelica.Media.IdealGases.SingleGases.PF2minus.....	1484
Modelica.Media.IdealGases.SingleGases.PF2CL.....	1484
Modelica.Media.IdealGases.SingleGases.PF2CL3.....	1484
Modelica.Media.IdealGases.SingleGases.PF3.....	1485
Modelica.Media.IdealGases.SingleGases.PF3CL2.....	1485
Modelica.Media.IdealGases.SingleGases.PF4CL.....	1485
Modelica.Media.IdealGases.SingleGases.PF5.....	1485
Modelica.Media.IdealGases.SingleGases.PH.....	1485
Modelica.Media.IdealGases.SingleGases.PH2.....	1486
Modelica.Media.IdealGases.SingleGases.PH2minus.....	1486
Modelica.Media.IdealGases.SingleGases.PH3.....	1486
Modelica.Media.IdealGases.SingleGases.PN.....	1486
Modelica.Media.IdealGases.SingleGases.PO.....	1486
Modelica.Media.IdealGases.SingleGases.POminus.....	1487
Modelica.Media.IdealGases.SingleGases.POCL3.....	1487
Modelica.Media.IdealGases.SingleGases.POFCL2.....	1487
Modelica.Media.IdealGases.SingleGases.POF2CL.....	1487
Modelica.Media.IdealGases.SingleGases.POF3.....	1487
Modelica.Media.IdealGases.SingleGases.PO2.....	1488
Modelica.Media.IdealGases.SingleGases.PO2minus.....	1488
Modelica.Media.IdealGases.SingleGases.PS.....	1488
Modelica.Media.IdealGases.SingleGases.P2.....	1488
Modelica.Media.IdealGases.SingleGases.P2O3.....	1488
Modelica.Media.IdealGases.SingleGases.P2O4.....	1489
Modelica.Media.IdealGases.SingleGases.P2O5.....	1489
Modelica.Media.IdealGases.SingleGases.P3.....	1489
Modelica.Media.IdealGases.SingleGases.P3O6.....	1489

---

Modelica.Media.IdealGases.SingleGases.P4.....	1489
Modelica.Media.IdealGases.SingleGases.P4O6.....	1490
Modelica.Media.IdealGases.SingleGases.P4O7.....	1490
Modelica.Media.IdealGases.SingleGases.P4O8.....	1490
Modelica.Media.IdealGases.SingleGases.P4O9.....	1490
Modelica.Media.IdealGases.SingleGases.P4O10.....	1490
Modelica.Media.IdealGases.SingleGases.Pb.....	1491
Modelica.Media.IdealGases.SingleGases.Pbplus.....	1491
Modelica.Media.IdealGases.SingleGases.Pbminus.....	1491
Modelica.Media.IdealGases.SingleGases.PbBr.....	1491
Modelica.Media.IdealGases.SingleGases.PbBr2.....	1491
Modelica.Media.IdealGases.SingleGases.PbBr3.....	1492
Modelica.Media.IdealGases.SingleGases.PbBr4.....	1492
Modelica.Media.IdealGases.SingleGases.PbCL.....	1492
Modelica.Media.IdealGases.SingleGases.PbCL2.....	1492
Modelica.Media.IdealGases.SingleGases.PbCL3.....	1492
Modelica.Media.IdealGases.SingleGases.PbCL4.....	1493
Modelica.Media.IdealGases.SingleGases.PbF.....	1493
Modelica.Media.IdealGases.SingleGases.PbF2.....	1493
Modelica.Media.IdealGases.SingleGases.PbF3.....	1493
Modelica.Media.IdealGases.SingleGases.PbF4.....	1493
Modelica.Media.IdealGases.SingleGases.Pbl.....	1494
Modelica.Media.IdealGases.SingleGases.Pbl2.....	1494
Modelica.Media.IdealGases.SingleGases.Pbl3.....	1494
Modelica.Media.IdealGases.SingleGases.Pbl4.....	1494
Modelica.Media.IdealGases.SingleGases.PbO.....	1494
Modelica.Media.IdealGases.SingleGases.PbO2.....	1495
Modelica.Media.IdealGases.SingleGases.PbS.....	1495
Modelica.Media.IdealGases.SingleGases.PbS2.....	1495
Modelica.Media.IdealGases.SingleGases.Rb.....	1495
Modelica.Media.IdealGases.SingleGases.Rbplus.....	1495
Modelica.Media.IdealGases.SingleGases.Rbminus.....	1496
Modelica.Media.IdealGases.SingleGases.RbBO2.....	1496
Modelica.Media.IdealGases.SingleGases.RbBr.....	1496
Modelica.Media.IdealGases.SingleGases.RbCL.....	1496
Modelica.Media.IdealGases.SingleGases.RbF.....	1496
Modelica.Media.IdealGases.SingleGases.RbH.....	1497
Modelica.Media.IdealGases.SingleGases.Rbl.....	1497
Modelica.Media.IdealGases.SingleGases.RbK.....	1497
Modelica.Media.IdealGases.SingleGases.RbLi.....	1497
Modelica.Media.IdealGases.SingleGases.RbNO2.....	1497
Modelica.Media.IdealGases.SingleGases.RbNO3.....	1498
Modelica.Media.IdealGases.SingleGases.RbNa.....	1498
Modelica.Media.IdealGases.SingleGases.RbO.....	1498
Modelica.Media.IdealGases.SingleGases.RbOH.....	1498
Modelica.Media.IdealGases.SingleGases.Rb2Br2.....	1498
Modelica.Media.IdealGases.SingleGases.Rb2CL2.....	1499
Modelica.Media.IdealGases.SingleGases.Rb2F2.....	1499
Modelica.Media.IdealGases.SingleGases.Rb2I2.....	1499
Modelica.Media.IdealGases.SingleGases.Rb2O.....	1499
Modelica.Media.IdealGases.SingleGases.Rb2O2.....	1499
Modelica.Media.IdealGases.SingleGases.Rb2O2H2.....	1500
Modelica.Media.IdealGases.SingleGases.Rb2SO4.....	1500
Modelica.Media.IdealGases.SingleGases.Rn.....	1500
Modelica.Media.IdealGases.SingleGases.Rnplus.....	1500
Modelica.Media.IdealGases.SingleGases.S.....	1500
Modelica.Media.IdealGases.SingleGases.Splus.....	1501
Modelica.Media.IdealGases.SingleGases.Sminus.....	1501

---

Modelica.Media.IdealGases.SingleGases.SCL.....	1501
Modelica.Media.IdealGases.SingleGases.SCL2.....	1501
Modelica.Media.IdealGases.SingleGases.SCL2plus.....	1501
Modelica.Media.IdealGases.SingleGases.SD.....	1502
Modelica.Media.IdealGases.SingleGases.SF.....	1502
Modelica.Media.IdealGases.SingleGases.SFplus.....	1502
Modelica.Media.IdealGases.SingleGases.SFminus.....	1502
Modelica.Media.IdealGases.SingleGases.SF2.....	1502
Modelica.Media.IdealGases.SingleGases.SF2plus.....	1503
Modelica.Media.IdealGases.SingleGases.SF2minus.....	1503
Modelica.Media.IdealGases.SingleGases.SF3.....	1503
Modelica.Media.IdealGases.SingleGases.SF3plus.....	1503
Modelica.Media.IdealGases.SingleGases.SF3minus.....	1503
Modelica.Media.IdealGases.SingleGases.SF4.....	1504
Modelica.Media.IdealGases.SingleGases.SF4plus.....	1504
Modelica.Media.IdealGases.SingleGases.SF4minus.....	1504
Modelica.Media.IdealGases.SingleGases.SF5.....	1504
Modelica.Media.IdealGases.SingleGases.SF5plus.....	1504
Modelica.Media.IdealGases.SingleGases.SF5minus.....	1505
Modelica.Media.IdealGases.SingleGases.SF6.....	1505
Modelica.Media.IdealGases.SingleGases.SF6minus.....	1505
Modelica.Media.IdealGases.SingleGases.SH.....	1505
Modelica.Media.IdealGases.SingleGases.SHminus.....	1505
Modelica.Media.IdealGases.SingleGases.SN.....	1506
Modelica.Media.IdealGases.SingleGases.SO.....	1506
Modelica.Media.IdealGases.SingleGases.SOminus.....	1506
Modelica.Media.IdealGases.SingleGases.SOF2.....	1506
Modelica.Media.IdealGases.SingleGases.SO2.....	1506
Modelica.Media.IdealGases.SingleGases.SO2minus.....	1507
Modelica.Media.IdealGases.SingleGases.SO2CL2.....	1507
Modelica.Media.IdealGases.SingleGases.SO2FCL.....	1507
Modelica.Media.IdealGases.SingleGases.SO2F2.....	1507
Modelica.Media.IdealGases.SingleGases.SO3.....	1507
Modelica.Media.IdealGases.SingleGases.S2.....	1508
Modelica.Media.IdealGases.SingleGases.S2minus.....	1508
Modelica.Media.IdealGases.SingleGases.S2CL2.....	1508
Modelica.Media.IdealGases.SingleGases.S2F2.....	1508
Modelica.Media.IdealGases.SingleGases.S2O.....	1508
Modelica.Media.IdealGases.SingleGases.S3.....	1509
Modelica.Media.IdealGases.SingleGases.S4.....	1509
Modelica.Media.IdealGases.SingleGases.S5.....	1510
Modelica.Media.IdealGases.SingleGases.S6.....	1510
Modelica.Media.IdealGases.SingleGases.S7.....	1511
Modelica.Media.IdealGases.SingleGases.S8.....	1511
Modelica.Media.IdealGases.SingleGases.Sc.....	1511
Modelica.Media.IdealGases.SingleGases.Scplus.....	1512
Modelica.Media.IdealGases.SingleGases.Scminus.....	1512
Modelica.Media.IdealGases.SingleGases.ScO.....	1512
Modelica.Media.IdealGases.SingleGases.ScOplus.....	1512
Modelica.Media.IdealGases.SingleGases.ScO2.....	1512
Modelica.Media.IdealGases.SingleGases.Sc2O.....	1513
Modelica.Media.IdealGases.SingleGases.Sc2O2.....	1513
Modelica.Media.IdealGases.SingleGases.Si.....	1513
Modelica.Media.IdealGases.SingleGases.Siplus.....	1513
Modelica.Media.IdealGases.SingleGases.Siminus.....	1513
Modelica.Media.IdealGases.SingleGases.SiBr.....	1514
Modelica.Media.IdealGases.SingleGases.SiBr2.....	1514
Modelica.Media.IdealGases.SingleGases.SiBr3.....	1514

---

Modelica.Media.IdealGases.SingleGases.SiBr4.....	1514
Modelica.Media.IdealGases.SingleGases.SiC.....	1515
Modelica.Media.IdealGases.SingleGases.SiC2.....	1515
Modelica.Media.IdealGases.SingleGases.SiCL.....	1516
Modelica.Media.IdealGases.SingleGases.SiCL2.....	1516
Modelica.Media.IdealGases.SingleGases.SiCL3.....	1517
Modelica.Media.IdealGases.SingleGases.SiCL4.....	1517
Modelica.Media.IdealGases.SingleGases.SiF.....	1517
Modelica.Media.IdealGases.SingleGases.SiFCL.....	1517
Modelica.Media.IdealGases.SingleGases.SiF2.....	1517
Modelica.Media.IdealGases.SingleGases.SiF3.....	1518
Modelica.Media.IdealGases.SingleGases.SiF4.....	1518
Modelica.Media.IdealGases.SingleGases.SiH.....	1518
Modelica.Media.IdealGases.SingleGases.SiHplus.....	1518
Modelica.Media.IdealGases.SingleGases.SiHBr3.....	1518
Modelica.Media.IdealGases.SingleGases.SiHCL.....	1519
Modelica.Media.IdealGases.SingleGases.SiHCL3.....	1519
Modelica.Media.IdealGases.SingleGases.SiHF.....	1519
Modelica.Media.IdealGases.SingleGases.SiHF3.....	1519
Modelica.Media.IdealGases.SingleGases.SiHI3.....	1519
Modelica.Media.IdealGases.SingleGases.SiH2.....	1520
Modelica.Media.IdealGases.SingleGases.SiH2Br2.....	1520
Modelica.Media.IdealGases.SingleGases.SiH2CL2.....	1520
Modelica.Media.IdealGases.SingleGases.SiH2F2.....	1520
Modelica.Media.IdealGases.SingleGases.SiH2I2.....	1520
Modelica.Media.IdealGases.SingleGases.SiH3.....	1521
Modelica.Media.IdealGases.SingleGases.SiH3Br.....	1521
Modelica.Media.IdealGases.SingleGases.SiH3CL.....	1521
Modelica.Media.IdealGases.SingleGases.SiH3F.....	1521
Modelica.Media.IdealGases.SingleGases.SiH3I.....	1521
Modelica.Media.IdealGases.SingleGases.SiH4.....	1522
Modelica.Media.IdealGases.SingleGases.SiI.....	1522
Modelica.Media.IdealGases.SingleGases.SiI2.....	1522
Modelica.Media.IdealGases.SingleGases.SiN.....	1522
Modelica.Media.IdealGases.SingleGases.SiO.....	1522
Modelica.Media.IdealGases.SingleGases.SiO2.....	1523
Modelica.Media.IdealGases.SingleGases.SiS.....	1523
Modelica.Media.IdealGases.SingleGases.SiS2.....	1523
Modelica.Media.IdealGases.SingleGases.Si2.....	1523
Modelica.Media.IdealGases.SingleGases.Si2C.....	1523
Modelica.Media.IdealGases.SingleGases.Si2F6.....	1524
Modelica.Media.IdealGases.SingleGases.Si2N.....	1524
Modelica.Media.IdealGases.SingleGases.Si3.....	1524
Modelica.Media.IdealGases.SingleGases.Sn.....	1524
Modelica.Media.IdealGases.SingleGases.Snplus.....	1524
Modelica.Media.IdealGases.SingleGases.Snminus.....	1525
Modelica.Media.IdealGases.SingleGases.SnBr.....	1525
Modelica.Media.IdealGases.SingleGases.SnBr2.....	1525
Modelica.Media.IdealGases.SingleGases.SnBr3.....	1525
Modelica.Media.IdealGases.SingleGases.SnBr4.....	1525
Modelica.Media.IdealGases.SingleGases.SnCL.....	1526
Modelica.Media.IdealGases.SingleGases.SnCL2.....	1526
Modelica.Media.IdealGases.SingleGases.SnCL3.....	1526
Modelica.Media.IdealGases.SingleGases.SnCL4.....	1526
Modelica.Media.IdealGases.SingleGases.SnF.....	1526
Modelica.Media.IdealGases.SingleGases.SnF2.....	1527
Modelica.Media.IdealGases.SingleGases.SnF3.....	1527
Modelica.Media.IdealGases.SingleGases.SnF4.....	1527

Modelica.Media.IdealGases.SingleGases.SnI.....	1527
Modelica.Media.IdealGases.SingleGases.SnI2.....	1527
Modelica.Media.IdealGases.SingleGases.SnI3.....	1528
Modelica.Media.IdealGases.SingleGases.SnI4.....	1528
Modelica.Media.IdealGases.SingleGases.SnO.....	1528
Modelica.Media.IdealGases.SingleGases.SnO2.....	1528
Modelica.Media.IdealGases.SingleGases.SnS.....	1528
Modelica.Media.IdealGases.SingleGases.SnS2.....	1529
Modelica.Media.IdealGases.SingleGases.Sn2.....	1529
Modelica.Media.IdealGases.SingleGases.Sr.....	1529
Modelica.Media.IdealGases.SingleGases.Srplus.....	1529
Modelica.Media.IdealGases.SingleGases.SrBr.....	1529
Modelica.Media.IdealGases.SingleGases.SrBr2.....	1530
Modelica.Media.IdealGases.SingleGases.SrCL.....	1530
Modelica.Media.IdealGases.SingleGases.SrCLplus.....	1530
Modelica.Media.IdealGases.SingleGases.SrCL2.....	1530
Modelica.Media.IdealGases.SingleGases.SrF.....	1530
Modelica.Media.IdealGases.SingleGases.SrFplus.....	1531
Modelica.Media.IdealGases.SingleGases.SrF2.....	1531
Modelica.Media.IdealGases.SingleGases.SrH.....	1531
Modelica.Media.IdealGases.SingleGases.SrI.....	1531
Modelica.Media.IdealGases.SingleGases.SrI2.....	1531
Modelica.Media.IdealGases.SingleGases.SrO.....	1532
Modelica.Media.IdealGases.SingleGases.SrOH.....	1532
Modelica.Media.IdealGases.SingleGases.SrOHplus.....	1532
Modelica.Media.IdealGases.SingleGases.Sr_OH_2.....	1532
Modelica.Media.IdealGases.SingleGases.SrS.....	1532
Modelica.Media.IdealGases.SingleGases.Sr2.....	1533
Modelica.Media.IdealGases.SingleGases.Ta.....	1533
Modelica.Media.IdealGases.SingleGases.Taplus.....	1533
Modelica.Media.IdealGases.SingleGases.Taminus.....	1533
Modelica.Media.IdealGases.SingleGases.TaCL5.....	1533
Modelica.Media.IdealGases.SingleGases.TaO.....	1534
Modelica.Media.IdealGases.SingleGases.TaO2.....	1534
Modelica.Media.IdealGases.SingleGases.Ti.....	1534
Modelica.Media.IdealGases.SingleGases.Tiplus.....	1534
Modelica.Media.IdealGases.SingleGases.Timinus.....	1534
Modelica.Media.IdealGases.SingleGases.TiCL.....	1535
Modelica.Media.IdealGases.SingleGases.TiCL2.....	1535
Modelica.Media.IdealGases.SingleGases.TiCL3.....	1535
Modelica.Media.IdealGases.SingleGases.TiCL4.....	1535
Modelica.Media.IdealGases.SingleGases.TiO.....	1535
Modelica.Media.IdealGases.SingleGases.TiOplus.....	1536
Modelica.Media.IdealGases.SingleGases.TiOCL.....	1536
Modelica.Media.IdealGases.SingleGases.TiOCL2.....	1536
Modelica.Media.IdealGases.SingleGases.TiO2.....	1536
Modelica.Media.IdealGases.SingleGases.U.....	1536
Modelica.Media.IdealGases.SingleGases.UF.....	1537
Modelica.Media.IdealGases.SingleGases.UFplus.....	1537
Modelica.Media.IdealGases.SingleGases.UFminus.....	1537
Modelica.Media.IdealGases.SingleGases.UF2.....	1537
Modelica.Media.IdealGases.SingleGases.UF2plus.....	1537
Modelica.Media.IdealGases.SingleGases.UF2minus.....	1538
Modelica.Media.IdealGases.SingleGases.UF3.....	1538
Modelica.Media.IdealGases.SingleGases.UF3plus.....	1538
Modelica.Media.IdealGases.SingleGases.UF3minus.....	1538
Modelica.Media.IdealGases.SingleGases.UF4.....	1538
Modelica.Media.IdealGases.SingleGases.UF4plus.....	1539

---

Modelica.Media.IdealGases.SingleGases.UF4minus.....	1539
Modelica.Media.IdealGases.SingleGases.UF5.....	1539
Modelica.Media.IdealGases.SingleGases.UF5plus.....	1540
Modelica.Media.IdealGases.SingleGases.UF5minus.....	1540
Modelica.Media.IdealGases.SingleGases.UF6.....	1541
Modelica.Media.IdealGases.SingleGases.UF6minus.....	1541
Modelica.Media.IdealGases.SingleGases.UO.....	1541
Modelica.Media.IdealGases.SingleGases.UOplus.....	1542
Modelica.Media.IdealGases.SingleGases.UOF.....	1542
Modelica.Media.IdealGases.SingleGases.UOF2.....	1542
Modelica.Media.IdealGases.SingleGases.UOF3.....	1542
Modelica.Media.IdealGases.SingleGases.UOF4.....	1542
Modelica.Media.IdealGases.SingleGases.UO2.....	1543
Modelica.Media.IdealGases.SingleGases.UO2plus.....	1543
Modelica.Media.IdealGases.SingleGases.UO2minus.....	1543
Modelica.Media.IdealGases.SingleGases.UO2F.....	1543
Modelica.Media.IdealGases.SingleGases.UO2F2.....	1543
Modelica.Media.IdealGases.SingleGases.UO3.....	1544
Modelica.Media.IdealGases.SingleGases.UO3minus.....	1544
Modelica.Media.IdealGases.SingleGases.V.....	1544
Modelica.Media.IdealGases.SingleGases.Vplus.....	1544
Modelica.Media.IdealGases.SingleGases.Vminus.....	1544
Modelica.Media.IdealGases.SingleGases.VCL4.....	1545
Modelica.Media.IdealGases.SingleGases.VN.....	1545
Modelica.Media.IdealGases.SingleGases.VO.....	1545
Modelica.Media.IdealGases.SingleGases.VO2.....	1545
Modelica.Media.IdealGases.SingleGases.V4O10.....	1545
Modelica.Media.IdealGases.SingleGases.W.....	1546
Modelica.Media.IdealGases.SingleGases.Wplus.....	1546
Modelica.Media.IdealGases.SingleGases.Wminus.....	1546
Modelica.Media.IdealGases.SingleGases.WCL6.....	1546
Modelica.Media.IdealGases.SingleGases.WO.....	1546
Modelica.Media.IdealGases.SingleGases.WOCL4.....	1547
Modelica.Media.IdealGases.SingleGases.WO2.....	1547
Modelica.Media.IdealGases.SingleGases.WO2CL2.....	1547
Modelica.Media.IdealGases.SingleGases.WO3.....	1547
Modelica.Media.IdealGases.SingleGases.WO3minus.....	1547
Modelica.Media.IdealGases.SingleGases.Xe.....	1548
Modelica.Media.IdealGases.SingleGases.Xeplus.....	1548
Modelica.Media.IdealGases.SingleGases.Zn.....	1548
Modelica.Media.IdealGases.SingleGases.Znplus.....	1548
Modelica.Media.IdealGases.SingleGases.Zr.....	1548
Modelica.Media.IdealGases.SingleGases.Zrplus.....	1549
Modelica.Media.IdealGases.SingleGases.Zrminus.....	1549
Modelica.Media.IdealGases.SingleGases.ZrN.....	1549
Modelica.Media.IdealGases.SingleGases.ZrO.....	1549
Modelica.Media.IdealGases.SingleGases.ZrOplus.....	1549
Modelica.Media.IdealGases.SingleGases.ZrO2.....	1550
Modelica.Media.Incompressible.....	1550
Modelica.Media.Incompressible.Common.....	1551
Modelica.Media.Incompressible.Common.BaseProps_Tpoly.....	1551
Modelica.Media.Incompressible.TableBased.....	1551
Modelica.Media.Incompressible.TableBased.invertTemp.....	1557
Modelica.Media.Incompressible.TableBased.BaseProperties.....	1557
Modelica.Media.Incompressible.TableBased.setState_pTX.....	1558
Modelica.Media.Incompressible.TableBased.setState_dTX.....	1558
Modelica.Media.Incompressible.TableBased.setState_pT.....	1559
Modelica.Media.Incompressible.TableBased.setState_phX.....	1559

Modelica.Media.Incompressible.TableBased.setState_ph.....	1559
Modelica.Media.Incompressible.TableBased.setState_psX.....	1559
Modelica.Media.Incompressible.TableBased.setState_ps.....	1560
Modelica.Media.Incompressible.TableBased.specificHeatCapacityCv.....	1560
Modelica.Media.Incompressible.TableBased.specificHeatCapacityCp.....	1560
Modelica.Media.Incompressible.TableBased.dynamicViscosity.....	1561
Modelica.Media.Incompressible.TableBased.thermalConductivity.....	1561
Modelica.Media.Incompressible.TableBased.s_T.....	1561
Modelica.Media.Incompressible.TableBased.specificEntropy.....	1561
Modelica.Media.Incompressible.TableBased.h_T.....	1562
Modelica.Media.Incompressible.TableBased.h_T_der.....	1562
Modelica.Media.Incompressible.TableBased.h_pT.....	1562
Modelica.Media.Incompressible.TableBased.density_T.....	1563
Modelica.Media.Incompressible.TableBased.temperature.....	1563
Modelica.Media.Incompressible.TableBased.pressure.....	1563
Modelica.Media.Incompressible.TableBased.density.....	1563
Modelica.Media.Incompressible.TableBased.specificEnthalpy.....	1564
Modelica.Media.Incompressible.TableBased.specificInternalEnergy.....	1564
Modelica.Media.Incompressible.TableBased.T_ph.....	1564
Modelica.Media.Incompressible.TableBased.T_ps.....	1564
Modelica.Media.Incompressible.TableBased.Polynomials_Temp.....	1565
Modelica.Media.Incompressible.TableBased.Polynomials_Temp.evaluate.....	1565
Modelica.Media.Incompressible.TableBased.Polynomials_Temp.derivative.....	1566
Modelica.Media.Incompressible.TableBased.Polynomials_Temp.derivativeValue.....	1566
Modelica.Media.Incompressible.TableBased.Polynomials_Temp.secondDerivativeValue.....	1566
Modelica.Media.Incompressible.TableBased.Polynomials_Temp.integral.....	1566
Modelica.Media.Incompressible.TableBased.Polynomials_Temp.integralValue.....	1567
Modelica.Media.Incompressible.TableBased.Polynomials_Temp.fitting.....	1567
Modelica.Media.Incompressible.TableBased.Polynomials_Temp.evaluate_der.....	1568
Modelica.Media.Incompressible.TableBased.Polynomials_Temp.integralValue_der.....	1568
Modelica.Media.Incompressible.TableBased.Polynomials_Temp.derivativeValue_der.....	1568
Modelica.Media.Incompressible.Examples.....	1569
Modelica.Media.Incompressible.Examples.Glycol47.....	1569
Modelica.Media.Incompressible.Examples.Essotherm650.....	1569
Modelica.Media.Incompressible.Examples.TestGlycol.....	1569
Modelica.Media.Water.....	1570
Modelica.Media.Water.ConstantPropertyLiquidWater.....	1573
Modelica.Media.Water.IdealSteam.....	1573
Modelica.Media.Water.WaterIF97OnePhase_ph.....	1573
Modelica.Media.Water.WaterIF97_ph.....	1573
Modelica.Media.Water.WaterIF97_base.....	1574
Modelica.Media.Water.WaterIF97_base.ThermodynamicState.....	1578
Modelica.Media.Water.WaterIF97_base.BaseProperties.....	1579
Modelica.Media.Water.WaterIF97_base.density_ph.....	1579
Modelica.Media.Water.WaterIF97_base.temperature_ph.....	1579
Modelica.Media.Water.WaterIF97_base.temperature_ps.....	1579
Modelica.Media.Water.WaterIF97_base.density_ps.....	1580
Modelica.Media.Water.WaterIF97_base.pressure_dT.....	1580
Modelica.Media.Water.WaterIF97_base.specificEnthalpy_dT.....	1580
Modelica.Media.Water.WaterIF97_base.specificEnthalpy_pT.....	1581
Modelica.Media.Water.WaterIF97_base.specificEnthalpy_ps.....	1581
Modelica.Media.Water.WaterIF97_base.density_pT.....	1581
Modelica.Media.Water.WaterIF97_base.setDewState.....	1582
Modelica.Media.Water.WaterIF97_base.setBubbleState.....	1582
Modelica.Media.Water.WaterIF97_base.dynamicViscosity.....	1582
Modelica.Media.Water.WaterIF97_base.thermalConductivity.....	1583
Modelica.Media.Water.WaterIF97_base.surfaceTension.....	1583
Modelica.Media.Water.WaterIF97_base.pressure.....	1583

Modelica.Media.Water.WaterIF97_base.temperature.....	1583
Modelica.Media.Water.WaterIF97_base.density.....	1584
Modelica.Media.Water.WaterIF97_base.specificEnthalpy.....	1584
Modelica.Media.Water.WaterIF97_base.specificInternalEnergy.....	1584
Modelica.Media.Water.WaterIF97_base.specificGibbsEnergy.....	1585
Modelica.Media.Water.WaterIF97_base.specificHelmholtzEnergy.....	1585
Modelica.Media.Water.WaterIF97_base.specificEntropy.....	1585
Modelica.Media.Water.WaterIF97_base.specificHeatCapacityCp.....	1585
Modelica.Media.Water.WaterIF97_base.specificHeatCapacityCv.....	1586
Modelica.Media.Water.WaterIF97_base.isentropicExponent.....	1586
Modelica.Media.Water.WaterIF97_base.isoenthalCompressibility.....	1586
Modelica.Media.Water.WaterIF97_base.isobaricExpansionCoefficient.....	1587
Modelica.Media.Water.WaterIF97_base.velocityOfSound.....	1587
Modelica.Media.Water.WaterIF97_base.isentropicEnthalpy.....	1587
Modelica.Media.Water.WaterIF97_base.density_derh_p.....	1587
Modelica.Media.Water.WaterIF97_base.density_derp_h.....	1588
Modelica.Media.Water.WaterIF97_base.bubbleEnthalpy.....	1588
Modelica.Media.Water.WaterIF97_base.dewEnthalpy.....	1588
Modelica.Media.Water.WaterIF97_base.bubbleEntropy.....	1589
Modelica.Media.Water.WaterIF97_base.dewEntropy.....	1589
Modelica.Media.Water.WaterIF97_base.bubbleDensity.....	1589
Modelica.Media.Water.WaterIF97_base.dewDensity.....	1589
Modelica.Media.Water.WaterIF97_base.saturationTemperature.....	1590
Modelica.Media.Water.WaterIF97_base.saturationTemperature_derp.....	1590
Modelica.Media.Water.WaterIF97_base.saturationPressure.....	1590
Modelica.Media.Water.WaterIF97_base.dBubbleDensity_dPressure.....	1590
Modelica.Media.Water.WaterIF97_base.dDewDensity_dPressure.....	1591
Modelica.Media.Water.WaterIF97_base.dBubbleEnthalpy_dPressure.....	1591
Modelica.Media.Water.WaterIF97_base.dDewEnthalpy_dPressure.....	1591
Modelica.Media.Water.WaterIF97_base.setState_dTX.....	1592
Modelica.Media.Water.WaterIF97_base.setState_phX.....	1592
Modelica.Media.Water.WaterIF97_base.setState_psX.....	1592
Modelica.Media.Water.WaterIF97_base.setState_pTX.....	1593
Modelica.Media.Water.WaterIF97_fixedregion.....	1593
Modelica.Media.Water.WaterIF97_fixedregion.ThermodynamicState.....	1598
Modelica.Media.Water.WaterIF97_fixedregion.BaseProperties.....	1598
Modelica.Media.Water.WaterIF97_fixedregion.density_ph.....	1598
Modelica.Media.Water.WaterIF97_fixedregion.temperature_ph.....	1598
Modelica.Media.Water.WaterIF97_fixedregion.temperature_ps.....	1599
Modelica.Media.Water.WaterIF97_fixedregion.density_ps.....	1599
Modelica.Media.Water.WaterIF97_fixedregion.pressure_dT.....	1599
Modelica.Media.Water.WaterIF97_fixedregion.specificEnthalpy_dT.....	1600
Modelica.Media.Water.WaterIF97_fixedregion.specificEnthalpy_pT.....	1600
Modelica.Media.Water.WaterIF97_fixedregion.specificEnthalpy_ps.....	1600
Modelica.Media.Water.WaterIF97_fixedregion.density_pT.....	1601
Modelica.Media.Water.WaterIF97_fixedregion.setDewState.....	1601
Modelica.Media.Water.WaterIF97_fixedregion.setBubbleState.....	1601
Modelica.Media.Water.WaterIF97_fixedregion.dynamicViscosity.....	1602
Modelica.Media.Water.WaterIF97_fixedregion.thermalConductivity.....	1602
Modelica.Media.Water.WaterIF97_fixedregion.surfaceTension.....	1602
Modelica.Media.Water.WaterIF97_fixedregion.pressure.....	1603
Modelica.Media.Water.WaterIF97_fixedregion.temperature.....	1603
Modelica.Media.Water.WaterIF97_fixedregion.density.....	1603
Modelica.Media.Water.WaterIF97_fixedregion.specificEnthalpy.....	1603
Modelica.Media.Water.WaterIF97_fixedregion.specificInternalEnergy.....	1604
Modelica.Media.Water.WaterIF97_fixedregion.specificGibbsEnergy.....	1604
Modelica.Media.Water.WaterIF97_fixedregion.specificHelmholtzEnergy.....	1604
Modelica.Media.Water.WaterIF97_fixedregion.specificEntropy.....	1605

Modelica.Media.Water.WaterIF97_fixedregion.specificHeatCapacityCp	1605
Modelica.Media.Water.WaterIF97_fixedregion.specificHeatCapacityCv	1605
Modelica.Media.Water.WaterIF97_fixedregion.isentropicExponent	1605
Modelica.Media.Water.WaterIF97_fixedregion.isothermalCompressibility	1606
Modelica.Media.Water.WaterIF97_fixedregion.isobaricExpansionCoefficient	1606
Modelica.Media.Water.WaterIF97_fixedregion.velocityOfSound	1606
Modelica.Media.Water.WaterIF97_fixedregion.isentropicEnthalpy	1606
Modelica.Media.Water.WaterIF97_fixedregion.density_derh_p	1607
Modelica.Media.Water.WaterIF97_fixedregion.density_derh_h	1607
Modelica.Media.Water.WaterIF97_fixedregion.bubbleEnthalpy	1607
Modelica.Media.Water.WaterIF97_fixedregion.dewEnthalpy	1608
Modelica.Media.Water.WaterIF97_fixedregion.bubbleEntropy	1608
Modelica.Media.Water.WaterIF97_fixedregion.dewEntropy	1608
Modelica.Media.Water.WaterIF97_fixedregion.bubbleDensity	1608
Modelica.Media.Water.WaterIF97_fixedregion.dewDensity	1609
Modelica.Media.Water.WaterIF97_fixedregion.saturationTemperature	1609
Modelica.Media.Water.WaterIF97_fixedregion.saturationTemperature_derh	1609
Modelica.Media.Water.WaterIF97_fixedregion.saturationPressure	1610
Modelica.Media.Water.WaterIF97_fixedregion.dBubbleDensity_dPressure	1610
Modelica.Media.Water.WaterIF97_fixedregion.dDewDensity_dPressure	1610
Modelica.Media.Water.WaterIF97_fixedregion.dBubbleEnthalpy_dPressure	1610
Modelica.Media.Water.WaterIF97_fixedregion.dDewEnthalpy_dPressure	1611
Modelica.Media.Water.WaterIF97_fixedregion.setState_dTX	1611
Modelica.Media.Water.WaterIF97_fixedregion.setState_phX	1611
Modelica.Media.Water.WaterIF97_fixedregion.setState_psX	1612
Modelica.Media.Water.WaterIF97_fixedregion.setState_pTX	1612
Modelica.Media.Water.WaterIF97_R1ph	1612
Modelica.Media.Water.WaterIF97_R2ph	1612
Modelica.Media.Water.WaterIF97_R3ph	1613
Modelica.Media.Water.WaterIF97_R4ph	1613
Modelica.Media.Water.WaterIF97_R5ph	1613
Modelica.Media.Water.WaterIF97_R1pT	1613
Modelica.Media.Water.WaterIF97_R2pT	1613
Modelica.Media.Water.IF97_Utilsities	1613
Modelica.Media.Water.IF97_Utilsities.BaselF97	1618
Modelica.Media.Water.IF97_Utilsities.BaselF97.IterationData	1621
Modelica.Media.Water.IF97_Utilsities.BaselF97.data	1621
Modelica.Media.Water.IF97_Utilsities.BaselF97.getTstar	1622
Modelica.Media.Water.IF97_Utilsities.BaselF97.getpstar	1622
Modelica.Media.Water.IF97_Utilsities.BaselF97.critical	1622
Modelica.Media.Water.IF97_Utilsities.BaselF97.triple	1623
Modelica.Media.Water.IF97_Utilsities.BaselF97.Regions	1623
Modelica.Media.Water.IF97_Utilsities.BaselF97.Regions.boundary23ofT	1626
Modelica.Media.Water.IF97_Utilsities.BaselF97.Regions.boundary23ofp	1627
Modelica.Media.Water.IF97_Utilsities.BaselF97.Regions.hlowerofp5	1627
Modelica.Media.Water.IF97_Utilsities.BaselF97.Regions.hupperofp5	1627
Modelica.Media.Water.IF97_Utilsities.BaselF97.Regions.slowerofp5	1627
Modelica.Media.Water.IF97_Utilsities.BaselF97.Regions.supperofp5	1628
Modelica.Media.Water.IF97_Utilsities.BaselF97.Regions.hlowerofp1	1628
Modelica.Media.Water.IF97_Utilsities.BaselF97.Regions.hupperofp1	1628
Modelica.Media.Water.IF97_Utilsities.BaselF97.Regions.slowerofp1	1628
Modelica.Media.Water.IF97_Utilsities.BaselF97.Regions.supperofp1	1629
Modelica.Media.Water.IF97_Utilsities.BaselF97.Regions.hlowerofp2	1629
Modelica.Media.Water.IF97_Utilsities.BaselF97.Regions.hupperofp2	1629
Modelica.Media.Water.IF97_Utilsities.BaselF97.Regions.slowerofp2	1630
Modelica.Media.Water.IF97_Utilsities.BaselF97.Regions.supperofp2	1630
Modelica.Media.Water.IF97_Utilsities.BaselF97.Regions.d1n	1630
Modelica.Media.Water.IF97_Utilsities.BaselF97.Regions.d2n	1630

Modelica.Media.Water.IF97.Utilities.BaselF97.Regions.dhot1ofp.....	1631
Modelica.Media.Water.IF97.Utilities.BaselF97.Regions.dupper1ofT.....	1631
Modelica.Media.Water.IF97.Utilities.BaselF97.Regions.hl_p_R4b.....	1631
Modelica.Media.Water.IF97.Utilities.BaselF97.Regions.hv_p_R4b.....	1632
Modelica.Media.Water.IF97.Utilities.BaselF97.Regions.sl_p_R4b.....	1632
Modelica.Media.Water.IF97.Utilities.BaselF97.Regions.sv_p_R4b.....	1632
Modelica.Media.Water.IF97.Utilities.BaselF97.Regions.rhol_p_R4b.....	1632
Modelica.Media.Water.IF97.Utilities.BaselF97.Regions.rhov_p_R4b.....	1633
Modelica.Media.Water.IF97.Utilities.BaselF97.Regions.boilingcurve_p.....	1633
Modelica.Media.Water.IF97.Utilities.BaselF97.Regions.dewcurve_p.....	1633
Modelica.Media.Water.IF97.Utilities.BaselF97.Regions.hvl_p.....	1634
Modelica.Media.Water.IF97.Utilities.BaselF97.Regions.hl_p.....	1634
Modelica.Media.Water.IF97.Utilities.BaselF97.Regions.hv_p.....	1634
Modelica.Media.Water.IF97.Utilities.BaselF97.Regions.hvl_p_der.....	1634
Modelica.Media.Water.IF97.Utilities.BaselF97.Regions.rhovl_p.....	1635
Modelica.Media.Water.IF97.Utilities.BaselF97.Regions.rhol_p.....	1635
Modelica.Media.Water.IF97.Utilities.BaselF97.Regions.rhov_p.....	1635
Modelica.Media.Water.IF97.Utilities.BaselF97.Regions.rhovl_p_der.....	1636
Modelica.Media.Water.IF97.Utilities.BaselF97.Regions.sl_p.....	1636
Modelica.Media.Water.IF97.Utilities.BaselF97.Regions.sv_p.....	1636
Modelica.Media.Water.IF97.Utilities.BaselF97.Regions.rhol_T.....	1636
Modelica.Media.Water.IF97.Utilities.BaselF97.Regions.rhov_T.....	1637
Modelica.Media.Water.IF97.Utilities.BaselF97.Regions.region_ph.....	1637
Modelica.Media.Water.IF97.Utilities.BaselF97.Regions.region_ps.....	1637
Modelica.Media.Water.IF97.Utilities.BaselF97.Regions.region_pT.....	1638
Modelica.Media.Water.IF97.Utilities.BaselF97.Regions.region_dT.....	1638
Modelica.Media.Water.IF97.Utilities.BaselF97.Regions.hvl_dp.....	1638
Modelica.Media.Water.IF97.Utilities.BaselF97.Regions.dhl_dp.....	1639
Modelica.Media.Water.IF97.Utilities.BaselF97.Regions.dhv_dp.....	1639
Modelica.Media.Water.IF97.Utilities.BaselF97.Regions.drhovl_dp.....	1639
Modelica.Media.Water.IF97.Utilities.BaselF97.Regions.drhol_dp.....	1640
Modelica.Media.Water.IF97.Utilities.BaselF97.Regions.drhov_dp.....	1640
Modelica.Media.Water.IF97.Utilities.BaselF97.Basic.....	1640
Modelica.Media.Water.IF97.Utilities.BaselF97.Basic.g1.....	1642
Modelica.Media.Water.IF97.Utilities.BaselF97.Basic.g2.....	1643
Modelica.Media.Water.IF97.Utilities.BaselF97.Basic.g2metastable.....	1643
Modelica.Media.Water.IF97.Utilities.BaselF97.Basic.f3.....	1643
Modelica.Media.Water.IF97.Utilities.BaselF97.Basic.g5.....	1643
Modelica.Media.Water.IF97.Utilities.BaselF97.Basic.gibbs.....	1644
Modelica.Media.Water.IF97.Utilities.BaselF97.Basic.g1pitau.....	1644
Modelica.Media.Water.IF97.Utilities.BaselF97.Basic.g2pitau.....	1644
Modelica.Media.Water.IF97.Utilities.BaselF97.Basic.g5pitau.....	1645
Modelica.Media.Water.IF97.Utilities.BaselF97.Basic.f3deltatau.....	1645
Modelica.Media.Water.IF97.Utilities.BaselF97.Basic.tph1.....	1646
Modelica.Media.Water.IF97.Utilities.BaselF97.Basic.tps1.....	1646
Modelica.Media.Water.IF97.Utilities.BaselF97.Basic.tph2.....	1646
Modelica.Media.Water.IF97.Utilities.BaselF97.Basic.tps2a.....	1646
Modelica.Media.Water.IF97.Utilities.BaselF97.Basic.tps2b.....	1647
Modelica.Media.Water.IF97.Utilities.BaselF97.Basic.tps2c.....	1647
Modelica.Media.Water.IF97.Utilities.BaselF97.Basic.tps2.....	1647
Modelica.Media.Water.IF97.Utilities.BaselF97.Basic.tsat.....	1648
Modelica.Media.Water.IF97.Utilities.BaselF97.Basic.dtsatofp.....	1648
Modelica.Media.Water.IF97.Utilities.BaselF97.Basic.tsat_der.....	1648
Modelica.Media.Water.IF97.Utilities.BaselF97.Basic.psat.....	1649
Modelica.Media.Water.IF97.Utilities.BaselF97.Basic.dptofT.....	1649
Modelica.Media.Water.IF97.Utilities.BaselF97.Basic.psat_der.....	1649
Modelica.Media.Water.IF97.Utilities.BaselF97.Basic.p1_hs.....	1649
Modelica.Media.Water.IF97.Utilities.BaselF97.Basic.h2ab_s.....	1650

Modelica.Media.Water.IF97.Utilities.BaselF97.Basic.p2a_hs.....	1650
Modelica.Media.Water.IF97.Utilities.BaselF97.Basic.p2b_hs.....	1651
Modelica.Media.Water.IF97.Utilities.BaselF97.Basic.p2c_hs.....	1651
Modelica.Media.Water.IF97.Utilities.BaselF97.Basic.h3ab_p.....	1652
Modelica.Media.Water.IF97.Utilities.BaselF97.Basic.T3a_ph.....	1652
Modelica.Media.Water.IF97.Utilities.BaselF97.Basic.T3b_ph.....	1653
Modelica.Media.Water.IF97.Utilities.BaselF97.Basic.v3a_ph.....	1653
Modelica.Media.Water.IF97.Utilities.BaselF97.Basic.v3b_ph.....	1654
Modelica.Media.Water.IF97.Utilities.BaselF97.Basic.T3a_ps.....	1654
Modelica.Media.Water.IF97.Utilities.BaselF97.Basic.T3b_ps.....	1655
Modelica.Media.Water.IF97.Utilities.BaselF97.Basic.v3a_ps.....	1655
Modelica.Media.Water.IF97.Utilities.BaselF97.Basic.v3b_ps.....	1656
Modelica.Media.Water.IF97.Utilities.BaselF97.IceBoundaries.....	1656
Modelica.Media.Water.IF97.Utilities.BaselF97.IceBoundaries.pmlcel_T.....	1656
Modelica.Media.Water.IF97.Utilities.BaselF97.IceBoundaries.pmlcellII_T.....	1657
Modelica.Media.Water.IF97.Utilities.BaselF97.IceBoundaries.pmlceV_T.....	1657
Modelica.Media.Water.IF97.Utilities.BaselF97.IceBoundaries.sublimationPressure_T.....	1658
Modelica.Media.Water.IF97.Utilities.BaselF97.Transport.....	1658
Modelica.Media.Water.IF97.Utilities.BaselF97.Transport.visc_dTp.....	1659
Modelica.Media.Water.IF97.Utilities.BaselF97.Transport.cond_dTp.....	1659
Modelica.Media.Water.IF97.Utilities.BaselF97.Transport.surfaceTension.....	1660
Modelica.Media.Water.IF97.Utilities.BaselF97.Isentropic.....	1660
Modelica.Media.Water.IF97.Utilities.BaselF97.Isentropic.hofpT1.....	1661
Modelica.Media.Water.IF97.Utilities.BaselF97.Isentropic.handsofpT1.....	1662
Modelica.Media.Water.IF97.Utilities.BaselF97.Isentropic.hofps1.....	1662
Modelica.Media.Water.IF97.Utilities.BaselF97.Isentropic.hofpT2.....	1662
Modelica.Media.Water.IF97.Utilities.BaselF97.Isentropic.handsofpT2.....	1662
Modelica.Media.Water.IF97.Utilities.BaselF97.Isentropic.hofps2.....	1663
Modelica.Media.Water.IF97.Utilities.BaselF97.Isentropic.hofdT3.....	1663
Modelica.Media.Water.IF97.Utilities.BaselF97.Isentropic.hofps3.....	1663
Modelica.Media.Water.IF97.Utilities.BaselF97.Isentropic.hofpsdt3.....	1664
Modelica.Media.Water.IF97.Utilities.BaselF97.Isentropic.hofps4.....	1664
Modelica.Media.Water.IF97.Utilities.BaselF97.Isentropic.hofpT5.....	1664
Modelica.Media.Water.IF97.Utilities.BaselF97.Isentropic.water_hisentropic.....	1665
Modelica.Media.Water.IF97.Utilities.BaselF97.Isentropic.water_hisentropic_dyn.....	1665
Modelica.Media.Water.IF97.Utilities.BaselF97.Inverses.....	1665
Modelica.Media.Water.IF97.Utilities.BaselF97.Inverses.fixdT.....	1667
Modelica.Media.Water.IF97.Utilities.BaselF97.Inverses.dofp13.....	1667
Modelica.Media.Water.IF97.Utilities.BaselF97.Inverses.dofp23.....	1667
Modelica.Media.Water.IF97.Utilities.BaselF97.Inverses.dofpt3.....	1667
Modelica.Media.Water.IF97.Utilities.BaselF97.Inverses.dtoph3.....	1668
Modelica.Media.Water.IF97.Utilities.BaselF97.Inverses.dtfps3.....	1668
Modelica.Media.Water.IF97.Utilities.BaselF97.Inverses.dtfpsdt3.....	1669
Modelica.Media.Water.IF97.Utilities.BaselF97.Inverses.pofdt125.....	1669
Modelica.Media.Water.IF97.Utilities.BaselF97.Inverses.tofph5.....	1669
Modelica.Media.Water.IF97.Utilities.BaselF97.Inverses.tofps5.....	1670
Modelica.Media.Water.IF97.Utilities.BaselF97.Inverses.tofpst5.....	1670
Modelica.Media.Water.IF97.Utilities.BaselF97.ByRegion.....	1670
Modelica.Media.Water.IF97.Utilities.BaselF97.ByRegion.waterR1_pT.....	1671
Modelica.Media.Water.IF97.Utilities.BaselF97.ByRegion.waterR2_pT.....	1671
Modelica.Media.Water.IF97.Utilities.BaselF97.ByRegion.waterR3_dT.....	1672
Modelica.Media.Water.IF97.Utilities.BaselF97.ByRegion.waterR5_pT.....	1672
Modelica.Media.Water.IF97.Utilities.BaselF97.TwoPhase.....	1672
Modelica.Media.Water.IF97.Utilities.BaselF97.TwoPhase.waterLiq_p.....	1673
Modelica.Media.Water.IF97.Utilities.BaselF97.TwoPhase.waterVap_p.....	1673
Modelica.Media.Water.IF97.Utilities.BaselF97.TwoPhase.waterSat_ph.....	1674
Modelica.Media.Water.IF97.Utilities.BaselF97.TwoPhase.waterR4_ph.....	1674
Modelica.Media.Water.IF97.Utilities.BaselF97.TwoPhase.waterR4_dT.....	1674

---

Modelica.Media.Water.IF97_Utilsities.BaselF97.extraDerivs_ph.....	1675
Modelica.Media.Water.IF97_Utilsities.BaselF97.extraDerivs_pT.....	1675
Modelica.Media.Water.IF97_Utilsities.iter.....	1675
Modelica.Media.Water.IF97_Utilsities.waterBaseProp_ph.....	1675
Modelica.Media.Water.IF97_Utilsities.waterBaseProp_ps.....	1676
Modelica.Media.Water.IF97_Utilsities.rho_props_ps.....	1676
Modelica.Media.Water.IF97_Utilsities.rho_ps.....	1676
Modelica.Media.Water.IF97_Utilsities.T_props_ps.....	1677
Modelica.Media.Water.IF97_Utilsities.T_ps.....	1677
Modelica.Media.Water.IF97_Utilsities.h_props_ps.....	1677
Modelica.Media.Water.IF97_Utilsities.h_ps.....	1678
Modelica.Media.Water.IF97_Utilsities.phase_ps.....	1678
Modelica.Media.Water.IF97_Utilsities.phase_ph.....	1678
Modelica.Media.Water.IF97_Utilsities.phase_dT.....	1679
Modelica.Media.Water.IF97_Utilsities.rho_props_ph.....	1679
Modelica.Media.Water.IF97_Utilsities.rho_ph.....	1679
Modelica.Media.Water.IF97_Utilsities.rho_ph_der.....	1680
Modelica.Media.Water.IF97_Utilsities.T_props_ph.....	1680
Modelica.Media.Water.IF97_Utilsities.T_ph.....	1680
Modelica.Media.Water.IF97_Utilsities.T_ph_der.....	1681
Modelica.Media.Water.IF97_Utilsities.s_props_ph.....	1681
Modelica.Media.Water.IF97_Utilsities.s_ph.....	1681
Modelica.Media.Water.IF97_Utilsities.s_ph_der.....	1682
Modelica.Media.Water.IF97_Utilsities.cv_props_ph.....	1682
Modelica.Media.Water.IF97_Utilsities.cv_ph.....	1682
Modelica.Media.Water.IF97_Utilsities.regionAssertReal.....	1683
Modelica.Media.Water.IF97_Utilsities.cp_props_ph.....	1683
Modelica.Media.Water.IF97_Utilsities.cp_ph.....	1683
Modelica.Media.Water.IF97_Utilsities.beta_props_ph.....	1684
Modelica.Media.Water.IF97_Utilsities.beta_ph.....	1684
Modelica.Media.Water.IF97_Utilsities.kappa_props_ph.....	1684
Modelica.Media.Water.IF97_Utilsities.kappa_ph.....	1685
Modelica.Media.Water.IF97_Utilsities.velocityOfSound_props_ph.....	1685
Modelica.Media.Water.IF97_Utilsities.velocityOfSound_ph.....	1685
Modelica.Media.Water.IF97_Utilsities.isentropicExponent_props_ph.....	1686
Modelica.Media.Water.IF97_Utilsities.isentropicExponent_ph.....	1686
Modelica.Media.Water.IF97_Utilsities.ddph_props.....	1686
Modelica.Media.Water.IF97_Utilsities.ddph.....	1687
Modelica.Media.Water.IF97_Utilsities.ddhp_props.....	1687
Modelica.Media.Water.IF97_Utilsities.ddhp.....	1687
Modelica.Media.Water.IF97_Utilsities.waterBaseProp_pT.....	1688
Modelica.Media.Water.IF97_Utilsities.rho_props_pT.....	1688
Modelica.Media.Water.IF97_Utilsities.rho_pT.....	1688
Modelica.Media.Water.IF97_Utilsities.h_props_pT.....	1689
Modelica.Media.Water.IF97_Utilsities.h_pT.....	1689
Modelica.Media.Water.IF97_Utilsities.h_pT_der.....	1689
Modelica.Media.Water.IF97_Utilsities.rho_pT_der.....	1690
Modelica.Media.Water.IF97_Utilsities.s_props_pT.....	1690
Modelica.Media.Water.IF97_Utilsities.s_pT.....	1690
Modelica.Media.Water.IF97_Utilsities.cv_props_pT.....	1691
Modelica.Media.Water.IF97_Utilsities.cv_pT.....	1691
Modelica.Media.Water.IF97_Utilsities.cp_props_pT.....	1691
Modelica.Media.Water.IF97_Utilsities.cp_pT.....	1692
Modelica.Media.Water.IF97_Utilsities.beta_props_pT.....	1692
Modelica.Media.Water.IF97_Utilsities.beta_pT.....	1692
Modelica.Media.Water.IF97_Utilsities.kappa_props_pT.....	1693
Modelica.Media.Water.IF97_Utilsities.kappa_pT.....	1693
Modelica.Media.Water.IF97_Utilsities.velocityOfSound_props_pT.....	1693

Modelica.Media.Water.IF97_Utilsities.velocityOfSound_pT.....	1694
Modelica.Media.Water.IF97_Utilsities.isentropicExponent_props_pT.....	1694
Modelica.Media.Water.IF97_Utilsities.isentropicExponent_pT.....	1694
Modelica.Media.Water.IF97_Utilsities.waterBaseProp_dT.....	1695
Modelica.Media.Water.IF97_Utilsities.h_props_dT.....	1695
Modelica.Media.Water.IF97_Utilsities.h_dT.....	1695
Modelica.Media.Water.IF97_Utilsities.h_dT_der.....	1696
Modelica.Media.Water.IF97_Utilsities.p_props_dT.....	1696
Modelica.Media.Water.IF97_Utilsities.p_dT.....	1696
Modelica.Media.Water.IF97_Utilsities.p_dT_der.....	1697
Modelica.Media.Water.IF97_Utilsities.s_props_dT.....	1697
Modelica.Media.Water.IF97_Utilsities.s_dT.....	1697
Modelica.Media.Water.IF97_Utilsities.cv_props_dT.....	1698
Modelica.Media.Water.IF97_Utilsities.cv_dT.....	1698
Modelica.Media.Water.IF97_Utilsities.cp_props_dT.....	1698
Modelica.Media.Water.IF97_Utilsities.cp_dT.....	1699
Modelica.Media.Water.IF97_Utilsities.beta_props_dT.....	1699
Modelica.Media.Water.IF97_Utilsities.beta_dT.....	1699
Modelica.Media.Water.IF97_Utilsities.kappa_props_dT.....	1700
Modelica.Media.Water.IF97_Utilsities.kappa_dT.....	1700
Modelica.Media.Water.IF97_Utilsities.velocityOfSound_props_dT.....	1700
Modelica.Media.Water.IF97_Utilsities.velocityOfSound_dT.....	1701
Modelica.Media.Water.IF97_Utilsities.isentropicExponent_props_dT.....	1701
Modelica.Media.Water.IF97_Utilsities.isentropicExponent_dT.....	1701
Modelica.Media.Water.IF97_Utilsities.hl_p.....	1702
Modelica.Media.Water.IF97_Utilsities.hv_p.....	1702
Modelica.Media.Water.IF97_Utilsities.sl_p.....	1702
Modelica.Media.Water.IF97_Utilsities.sv_p.....	1702
Modelica.Media.Water.IF97_Utilsities.rhol_T.....	1703
Modelica.Media.Water.IF97_Utilsities.rhov_T.....	1703
Modelica.Media.Water.IF97_Utilsities.rhol_p.....	1703
Modelica.Media.Water.IF97_Utilsities.rhov_p.....	1704
Modelica.Media.Water.IF97_Utilsities.dynamicViscosity.....	1704
Modelica.Media.Water.IF97_Utilsities.thermalConductivity.....	1704
Modelica.Media.Water.IF97_Utilsities.surfaceTension.....	1705
Modelica.Media.Water.IF97_Utilsities.isentropicEnthalpy.....	1705
Modelica.Media.Water.IF97_Utilsities.isentropicEnthalpy_props.....	1705
Modelica.Media.Water.IF97_Utilsities.isentropicEnthalpy_der.....	1705
Modelica.Media.Water.IF97_Utilsities.dynamicIsentropicEnthalpy.....	1706
Modelica.Slunits.....	1706
Modelica.Slunits.UsersGuide.....	1741
Modelica.Slunits.UsersGuide.HowToUseSlunits.....	1741
Modelica.Slunits.UsersGuide.Conventions.....	1743
Modelica.Slunits.UsersGuide.Literature.....	1743
Modelica.Slunits.UsersGuide.Contact.....	1743
Modelica.Slunits.Conversions.....	1744
Modelica.Slunits.Conversions.NonSlunits.....	1745
Modelica.Slunits.Conversions.NonSlunits.Temperature_degC.....	1746
Modelica.Slunits.Conversions.NonSlunits.Temperature_degF.....	1746
Modelica.Slunits.Conversions.NonSlunits.Temperature_degRk.....	1746
Modelica.Slunits.Conversions.NonSlunits.Angle_deg.....	1747
Modelica.Slunits.Conversions.NonSlunits.AngularVelocity_rpm.....	1747
Modelica.Slunits.Conversions.NonSlunits.Velocity_kmh.....	1747
Modelica.Slunits.Conversions.NonSlunits.Time_day.....	1747
Modelica.Slunits.Conversions.NonSlunits.Time_hour.....	1747
Modelica.Slunits.Conversions.NonSlunits.Time_minute.....	1747
Modelica.Slunits.Conversions.NonSlunits.Volume_litre.....	1747
Modelica.Slunits.Conversions.NonSlunits.Energy_kWh.....	1747

---

Modelica.Slunits.Conversions.NonSlunits.Pressure_bar.....	1747
Modelica.Slunits.Conversions.NonSlunits.MassFlowRate_gps.....	1747
Modelica.Slunits.Conversions.to_degC.....	1747
Modelica.Slunits.Conversions.from_degC.....	1748
Modelica.Slunits.Conversions.to_degF.....	1748
Modelica.Slunits.Conversions.from_degF.....	1748
Modelica.Slunits.Conversions.to_degRk.....	1748
Modelica.Slunits.Conversions.from_degRk.....	1749
Modelica.Slunits.Conversions.to_deg.....	1749
Modelica.Slunits.Conversions.from_deg.....	1749
Modelica.Slunits.Conversions.to_rpm.....	1749
Modelica.Slunits.Conversions.from_rpm.....	1750
Modelica.Slunits.Conversions.to_kmh.....	1750
Modelica.Slunits.Conversions.from_kmh.....	1750
Modelica.Slunits.Conversions.to_day.....	1751
Modelica.Slunits.Conversions.from_day.....	1751
Modelica.Slunits.Conversions.to_hour.....	1751
Modelica.Slunits.Conversions.from_hour.....	1751
Modelica.Slunits.Conversions.to_minute.....	1752
Modelica.Slunits.Conversions.from_minute.....	1752
Modelica.Slunits.Conversions.to_litre.....	1752
Modelica.Slunits.Conversions.from_litre.....	1752
Modelica.Slunits.Conversions.to_kWh.....	1753
Modelica.Slunits.Conversions.from_kWh.....	1753
Modelica.Slunits.Conversions.to_bar.....	1753
Modelica.Slunits.Conversions.from_bar.....	1754
Modelica.Slunits.Conversions.to_gps.....	1754
Modelica.Slunits.Conversions.from_gps.....	1754
Modelica.Slunits.Conversions.ConversionIcon.....	1754
Modelica.Slunits.Angle.....	1755
Modelica.Slunits.SolidAngle.....	1755
Modelica.Slunits.Length.....	1755
Modelica.Slunits.PathLength.....	1755
Modelica.Slunits.Position.....	1755
Modelica.Slunits.Distance.....	1755
Modelica.Slunits.Breadth.....	1755
Modelica.Slunits.Height.....	1755
Modelica.Slunits.Thickness.....	1756
Modelica.Slunits.Radius.....	1756
Modelica.Slunits.Diameter.....	1756
Modelica.Slunits.Area.....	1756
Modelica.Slunits.Volume.....	1756
Modelica.Slunits.Time.....	1756
Modelica.Slunits.Duration.....	1756
Modelica.Slunits.AngularVelocity.....	1756
Modelica.Slunits.AngularAcceleration.....	1756
Modelica.Slunits.Velocity.....	1757
Modelica.Slunits.Acceleration.....	1757
Modelica.Slunits.Period.....	1757
Modelica.Slunits.Frequency.....	1757
Modelica.Slunits.AngularFrequency.....	1757
Modelica.Slunits.Wavelength.....	1757
Modelica.Slunits.Wavelenght.....	1757
Modelica.Slunits.WaveNumber.....	1757
Modelica.Slunits.CircularWaveNumber.....	1757
Modelica.Slunits.AmplitudeLevelDifference.....	1757
Modelica.Slunits.PowerLevelDifference.....	1757
Modelica.Slunits.DampingCoefficient.....	1757

---

Modelica.Slunits.LogarithmicDecrement.....	1757
Modelica.Slunits.AttenuationCoefficient.....	1757
Modelica.Slunits.PhaseCoefficient.....	1758
Modelica.Slunits.PropagationCoefficient.....	1758
Modelica.Slunits.Damping.....	1758
Modelica.Slunits.Mass.....	1758
Modelica.Slunits.Density.....	1758
Modelica.Slunits.RelativeDensity.....	1758
Modelica.Slunits.SpecificVolume.....	1758
Modelica.Slunits.LinearDensity.....	1758
Modelica.Slunits.SurfaceDensity.....	1759
Modelica.Slunits.Momentum.....	1759
Modelica.Slunits.Impulse.....	1759
Modelica.Slunits.AngularMomentum.....	1759
Modelica.Slunits.AngularImpulse.....	1759
Modelica.Slunits.MomentOfInertia.....	1759
Modelica.Slunits.Inertia.....	1759
Modelica.Slunits.Force.....	1759
Modelica.Slunits.Weight.....	1759
Modelica.Slunits.Torque.....	1759
Modelica.Slunits.MomentOfForce.....	1759
Modelica.Slunits.Pressure.....	1760
Modelica.Slunits.AbsolutePressure.....	1760
Modelica.Slunits.BulkModulus.....	1760
Modelica.Slunits.Stress.....	1760
Modelica.Slunits.NormalStress.....	1760
Modelica.Slunits.ShearStress.....	1760
Modelica.Slunits.Strain.....	1760
Modelica.Slunits.LinearStrain.....	1760
Modelica.Slunits.ShearStrain.....	1760
Modelica.Slunits.VolumeStrain.....	1760
Modelica.Slunits.PoissonNumber.....	1760
Modelica.Slunits.ModulusOfElasticity.....	1760
Modelica.Slunits.ShearModulus.....	1761
Modelica.Slunits.SecondMomentOfArea.....	1761
Modelica.Slunits.SecondPolarMomentOfArea.....	1761
Modelica.Slunits.SectionModulus.....	1761
Modelica.Slunits.CoefficientOfFriction.....	1761
Modelica.Slunits.DynamicViscosity.....	1761
Modelica.Slunits.KinematicViscosity.....	1761
Modelica.Slunits.SurfaceTension.....	1761
Modelica.Slunits.Work.....	1761
Modelica.Slunits.Energy.....	1761
Modelica.Slunits.EnergyDensity.....	1761
Modelica.Slunits.PotentialEnergy.....	1761
Modelica.Slunits.KineticEnergy.....	1762
Modelica.Slunits.Power.....	1762
Modelica.Slunits.EnergyFlowRate.....	1762
Modelica.Slunits.EnthalpyFlowRate.....	1762
Modelica.Slunits.Efficiency.....	1762
Modelica.Slunits.MassFlowRate.....	1762
Modelica.Slunits.VolumeFlowRate.....	1762
Modelica.Slunits.MomentumFlux.....	1762
Modelica.Slunits.AngularMomentumFlux.....	1762
Modelica.Slunits.ThermodynamicTemperature.....	1762
Modelica.Slunits.Temp_K.....	1763
Modelica.Slunits.Temperature.....	1763
Modelica.Slunits.TemperatureDifference.....	1763

---

Modelica.Slunits.CelsiusTemperature.....	1763
Modelica.Slunits.Temp_C.....	1763
Modelica.Slunits.LinearExpansionCoefficient.....	1763
Modelica.Slunits.CubicExpansionCoefficient.....	1763
Modelica.Slunits.RelativePressureCoefficient.....	1763
Modelica.Slunits.PressureCoefficient.....	1763
Modelica.Slunits.Compressibility.....	1763
Modelica.Slunits.IsothermalCompressibility.....	1763
Modelica.Slunits.IsentropicCompressibility.....	1763
Modelica.Slunits.Heat.....	1763
Modelica.Slunits.HeatFlowRate.....	1763
Modelica.Slunits.HeatFlux.....	1764
Modelica.Slunits.DensityOfHeatFlowRate.....	1764
Modelica.Slunits.ThermalConductivity.....	1764
Modelica.Slunits.CoefficientOfHeatTransfer.....	1764
Modelica.Slunits.SurfaceCoefficientOfHeatTransfer.....	1764
Modelica.Slunits.ThermalInsulance.....	1764
Modelica.Slunits.ThermalResistance.....	1764
Modelica.Slunits.ThermalConductance.....	1764
Modelica.Slunits.ThermalDiffusivity.....	1764
Modelica.Slunits.HeatCapacity.....	1764
Modelica.Slunits.SpecificHeatCapacity.....	1764
Modelica.Slunits.SpecificHeatCapacityAtConstantPressure.....	1764
Modelica.Slunits.SpecificHeatCapacityAtConstantVolume.....	1764
Modelica.Slunits.SpecificHeatCapacityAtSaturation.....	1764
Modelica.Slunits.RatioOfSpecificHeatCapacities.....	1765
Modelica.Slunits.IsentropicExponent.....	1765
Modelica.Slunits.Entropy.....	1765
Modelica.Slunits.SpecificEntropy.....	1765
Modelica.Slunits.InternalEnergy.....	1765
Modelica.Slunits.Enthalpy.....	1765
Modelica.Slunits.HelmholtzFreeEnergy.....	1765
Modelica.Slunits.GibbsFreeEnergy.....	1765
Modelica.Slunits.SpecificEnergy.....	1765
Modelica.Slunits.SpecificInternalEnergy.....	1765
Modelica.Slunits.SpecificEnthalpy.....	1765
Modelica.Slunits.SpecificHelmholtzFreeEnergy.....	1765
Modelica.Slunits.SpecificGibbsFreeEnergy.....	1765
Modelica.Slunits.MassieuFunction.....	1765
Modelica.Slunits.PlanckFunction.....	1766
Modelica.Slunits.DerDensityByEnthalpy.....	1766
Modelica.Slunits.DerDensityByPressure.....	1766
Modelica.Slunits.DerDensityByTemperature.....	1766
Modelica.Slunits.DerEnthalpyByPressure.....	1766
Modelica.Slunits.DerEnergyByDensity.....	1766
Modelica.Slunits.DerEnergyByPressure.....	1766
Modelica.Slunits.ElectricCurrent.....	1766
Modelica.Slunits.Current.....	1766
Modelica.Slunits.ElectricCharge.....	1766
Modelica.Slunits.Charge.....	1766
Modelica.Slunits.VolumeDensityOfCharge.....	1766
Modelica.Slunits.SurfaceDensityOfCharge.....	1767
Modelica.Slunits.ElectricFieldStrength.....	1767
Modelica.Slunits.ElectricPotential.....	1767
Modelica.Slunits.Voltage.....	1767
Modelica.Slunits.PotentialDifference.....	1767
Modelica.Slunits.ElectromotiveForce.....	1767
Modelica.Slunits.ElectricFluxDensity.....	1767

---

Modelica.SIunits.ElectricFlux.....	1767
Modelica.SIunits.Capacitance.....	1767
Modelica.SIunits.Permittivity.....	1767
Modelica.SIunits.PermittivityOfVacuum.....	1768
Modelica.SIunits.RelativePermittivity.....	1768
Modelica.SIunits.ElectricSusceptibility.....	1768
Modelica.SIunits.ElectricPolarization.....	1768
Modelica.SIunits.Electrization.....	1768
Modelica.SIunits.ElectricDipoleMoment.....	1768
Modelica.SIunits.CurrentDensity.....	1768
Modelica.SIunits.LinearCurrentDensity.....	1768
Modelica.SIunits.MagneticFieldStrength.....	1768
Modelica.SIunits.MagneticPotential.....	1768
Modelica.SIunits.MagneticPotentialDifference.....	1768
Modelica.SIunits.MagnetomotiveForce.....	1768
Modelica.SIunits.CurrentLinkage.....	1768
Modelica.SIunits.MagneticFluxDensity.....	1768
Modelica.SIunits.MagneticFlux.....	1769
Modelica.SIunits.MagneticVectorPotential.....	1769
Modelica.SIunits.Inductance.....	1769
Modelica.SIunits.SelfInductance.....	1769
Modelica.SIunits.MutualInductance.....	1769
Modelica.SIunits.CouplingCoefficient.....	1769
Modelica.SIunits.LeakageCoefficient.....	1769
Modelica.SIunits.Permeability.....	1769
Modelica.SIunits.PermeabilityOfVacuum.....	1769
Modelica.SIunits.RelativePermeability.....	1769
Modelica.SIunits.MagneticSusceptibility.....	1769
Modelica.SIunits.ElectromagneticMoment.....	1769
Modelica.SIunits.MagneticDipoleMoment.....	1769
Modelica.SIunits.Magnetization.....	1770
Modelica.SIunits.MagneticPolarization.....	1770
Modelica.SIunits.ElectromagneticEnergyDensity.....	1770
Modelica.SIunits.PoyntingVector.....	1770
Modelica.SIunits.Resistance.....	1770
Modelica.SIunits.Resistivity.....	1770
Modelica.SIunits.Conductivity.....	1770
Modelica.SIunits.Reluctance.....	1770
Modelica.SIunits.Permeance.....	1770
Modelica.SIunits.PhaseDifference.....	1770
Modelica.SIunits.Impedance.....	1770
Modelica.SIunits.ModulusOfImpedance.....	1770
Modelica.SIunits.Reactance.....	1770
Modelica.SIunits.QualityFactor.....	1771
Modelica.SIunits.LossAngle.....	1771
Modelica.SIunits.Conductance.....	1771
Modelica.SIunits.Admittance.....	1771
Modelica.SIunits.ModulusOfAdmittance.....	1771
Modelica.SIunits.Susceptance.....	1771
Modelica.SIunits.InstantaneousPower.....	1771
Modelica.SIunits.ActivePower.....	1771
Modelica.SIunits.ApparentPower.....	1771
Modelica.SIunits.ReactivePower.....	1771
Modelica.SIunits.PowerFactor.....	1771
Modelica.SIunits.Transconductance.....	1771
Modelica.SIunits.InversePotential.....	1771
Modelica.SIunits.RadiantEnergy.....	1772
Modelica.SIunits.RadiantEnergyDensity.....	1772

---

Modelica.Slunits.SpectralRadiantEnergyDensity.....	1772
Modelica.Slunits.RadiantPower.....	1772
Modelica.Slunits.RadiantEnergyFluenceRate.....	1772
Modelica.Slunits.RadiantIntensity.....	1772
Modelica.Slunits.Radiance.....	1772
Modelica.Slunits.RadiantExtiance.....	1772
Modelica.Slunits.Irradiance.....	1772
Modelica.Slunits.Emissivity.....	1772
Modelica.Slunits.SpectralEmissivity.....	1772
Modelica.Slunits.DirectionalspectralEmissivity.....	1772
Modelica.Slunits.LuminousIntensity.....	1772
Modelica.Slunits.LuminousFlux.....	1772
Modelica.Slunits.QuantityOfLight.....	1773
Modelica.Slunits.Luminance.....	1773
Modelica.Slunits.LuminousExitance.....	1773
Modelica.Slunits.Illuminance.....	1773
Modelica.Slunits.LightExposure.....	1773
Modelica.Slunits.LuminousEfficacy.....	1773
Modelica.Slunits.SpectralLuminousEfficacy.....	1773
Modelica.Slunits.LuminousEfficiency.....	1773
Modelica.Slunits.SpectralLuminousEfficiency.....	1773
Modelica.Slunits.CIESpectralTristimulusValues.....	1773
Modelica.Slunits.ChromaticityCoordinates.....	1773
Modelica.Slunits.SpectralAbsorptionFactor.....	1773
Modelica.Slunits.SpectralReflectionFactor.....	1773
Modelica.Slunits.SpectralTransmissionFactor.....	1773
Modelica.Slunits.SpectralRadianceFactor.....	1774
Modelica.Slunits.LinearAttenuationCoefficient.....	1774
Modelica.Slunits.LinearAbsorptionCoefficient.....	1774
Modelica.Slunits.MolarAbsorptionCoefficient.....	1774
Modelica.Slunits.RefractiveIndex.....	1774
Modelica.Slunits.StaticPressure.....	1774
Modelica.Slunits.SoundPressure.....	1774
Modelica.Slunits.SoundParticleDisplacement.....	1774
Modelica.Slunits.SoundParticleVelocity.....	1774
Modelica.Slunits.SoundParticleAcceleration.....	1774
Modelica.Slunits.VelocityOfSound.....	1774
Modelica.Slunits.SoundEnergyDensity.....	1774
Modelica.Slunits.SoundPower.....	1775
Modelica.Slunits.SoundIntensity.....	1775
Modelica.Slunits.AcousticImpedance.....	1775
Modelica.Slunits.SpecificAcousticImpedance.....	1775
Modelica.Slunits.MechanicalImpedance.....	1775
Modelica.Slunits.SoundPressureLevel.....	1775
Modelica.Slunits.SoundPowerLevel.....	1775
Modelica.Slunits.DissipationCoefficient.....	1775
Modelica.Slunits.ReflectionCoefficient.....	1775
Modelica.Slunits.TransmissionCoefficient.....	1775
Modelica.Slunits.AcousticAbsorptionCoefficient.....	1775
Modelica.Slunits.SoundReductionIndex.....	1775
Modelica.Slunits.EquivalentAbsorptionArea.....	1775
Modelica.Slunits.ReverberationTime.....	1775
Modelica.Slunits.LoundnessLevel.....	1776
Modelica.Slunits.Loundness.....	1776
Modelica.Slunits.RelativeAtomicMass.....	1776
Modelica.Slunits.RelativeMolecularMass.....	1776
Modelica.Slunits.NumberOfMolecules.....	1776
Modelica.Slunits.AmountOfSubstance.....	1776

Modelica.Slunits.MolarMass.....	1776
Modelica.Slunits.MolarVolume.....	1776
Modelica.Slunits.MolarInternalEnergy.....	1776
Modelica.Slunits.MolarHeatCapacity.....	1776
Modelica.Slunits.MolarEntropy.....	1776
Modelica.Slunits.NumberDensityOfMolecules.....	1776
Modelica.Slunits.MolecularConcentration.....	1776
Modelica.Slunits.MassConcentration.....	1777
Modelica.Slunits.MassFraction.....	1777
Modelica.Slunits.Concentration.....	1777
Modelica.Slunits.VolumeFraction.....	1777
Modelica.Slunits.MoleFraction.....	1777
Modelica.Slunits.ChemicalPotential.....	1777
Modelica.Slunits.AbsoluteActivity.....	1777
Modelica.Slunits.PartialPressure.....	1777
Modelica.Slunits.Fugacity.....	1777
Modelica.Slunits.StandardAbsoluteActivity.....	1777
Modelica.Slunits.ActivityCoefficient.....	1777
Modelica.Slunits.ActivityOfSolute.....	1777
Modelica.Slunits.ActivityCoefficientOfSolute.....	1778
Modelica.Slunits.StandardAbsoluteActivityOfSolute.....	1778
Modelica.Slunits.ActivityOfSolvent.....	1778
Modelica.Slunits.OsmoticCoefficientOfSolvent.....	1778
Modelica.Slunits.StandardAbsoluteActivityOfSolvent.....	1778
Modelica.Slunits.OsmoticPressure.....	1778
Modelica.Slunits.StoichiometricNumber.....	1778
Modelica.Slunits.Affinity.....	1778
Modelica.Slunits.MassOfMolecule.....	1778
Modelica.Slunits.ElectricDipoleMomentOfMolecule.....	1778
Modelica.Slunits.ElectricPolarizabilityOfAMolecule.....	1778
Modelica.Slunits.MicrocanonicalPartitionFunction.....	1778
Modelica.Slunits.CanonicalPartitionFunction.....	1779
Modelica.Slunits.GrandCanonicalPartitionFunction.....	1779
Modelica.Slunits.MolecularPartitionFunction.....	1779
Modelica.Slunits.StatisticalWeight.....	1779
Modelica.Slunits.MeanFreePath.....	1779
Modelica.Slunits.DiffusionCoefficient.....	1779
Modelica.Slunits.ThermalDiffusionRatio.....	1779
Modelica.Slunits.ThermalDiffusionFactor.....	1779
Modelica.Slunits.ThermalDiffusionCoefficient.....	1779
Modelica.Slunits.ElementaryCharge.....	1779
Modelica.Slunits.ChargeNumberOfIon.....	1779
Modelica.Slunits.FaradayConstant.....	1779
Modelica.Slunits.IonicStrength.....	1779
Modelica.Slunits.DegreeOfDissociation.....	1779
Modelica.Slunits.ElectrolyticConductivity.....	1780
Modelica.Slunits.MolarConductivity.....	1780
Modelica.Slunits.TransportNumberOfIonic.....	1780
Modelica.Slunits.ProtonNumber.....	1780
Modelica.Slunits.NeutronNumber.....	1780
Modelica.Slunits.NucleonNumber.....	1780
Modelica.Slunits.AtomicMassConstant.....	1780
Modelica.Slunits.MassOfElectron.....	1780
Modelica.Slunits.MassOfProton.....	1780
Modelica.Slunits.MassOfNeutron.....	1780
Modelica.Slunits.HartreeEnergy.....	1780
Modelica.Slunits.MagneticMomentOfParticle.....	1780
Modelica.Slunits.BohrMagneton.....	1780

---

Modelica.Slunits.NuclearMagneton.....	1780
Modelica.Slunits.GyromagneticCoefficient.....	1781
Modelica.Slunits.GFactorOfAtom.....	1781
Modelica.Slunits.GFactorOfNucleus.....	1781
Modelica.Slunits.LarmorAngularFrequency.....	1781
Modelica.Slunits.NuclearPrecessionAngularFrequency.....	1781
Modelica.Slunits.CyclotronAngularFrequency.....	1781
Modelica.Slunits.NuclearQuadrupoleMoment.....	1781
Modelica.Slunits.NuclearRadius.....	1781
Modelica.Slunits.ElectronRadius.....	1781
Modelica.Slunits.ComptonWavelength.....	1781
Modelica.Slunits.MassExcess.....	1781
Modelica.Slunits.MassDefect.....	1781
Modelica.Slunits.RelativeMassExcess.....	1781
Modelica.Slunits.RelativeMassDefect.....	1781
Modelica.Slunits.PackingFraction.....	1782
Modelica.Slunits.BindingFraction.....	1782
Modelica.Slunits.MeanLife.....	1782
Modelica.Slunits.LevelWidth.....	1782
Modelica.Slunits.Activity.....	1782
Modelica.Slunits.SpecificActivity.....	1782
Modelica.Slunits.DecayConstant.....	1782
Modelica.Slunits.HalfLife.....	1782
Modelica.Slunits.AlphaDisintegrationEnergy.....	1782
Modelica.Slunits.MaximumBetaParticleEnergy.....	1782
Modelica.Slunits.BetaDisintegrationEnergy.....	1782
Modelica.Slunits.ReactionEnergy.....	1782
Modelica.Slunits.ResonanceEnergy.....	1782
Modelica.Slunits.CrossSection.....	1782
Modelica.Slunits.TotalCrossSection.....	1783
Modelica.Slunits.AngularCrossSection.....	1783
Modelica.Slunits.SpectralCrossSection.....	1783
Modelica.Slunits.SpectralAngularCrossSection.....	1783
Modelica.Slunits.MacroscopicCrossSection.....	1783
Modelica.Slunits.TotalMacroscopicCrossSection.....	1783
Modelica.Slunits.ParticleFluence.....	1783
Modelica.Slunits.ParticleFluenceRate.....	1783
Modelica.Slunits.EnergyFluence.....	1783
Modelica.Slunits.EnergyFluenceRate.....	1783
Modelica.Slunits.CurrentDensityOfParticles.....	1783
Modelica.Slunits.MassAttenuationCoefficient.....	1783
Modelica.Slunits.MolarAttenuationCoefficient.....	1783
Modelica.Slunits.AtomicAttenuationCoefficient.....	1783
Modelica.Slunits.HalfThickness.....	1784
Modelica.Slunits.TotalLinearStoppingPower.....	1784
Modelica.Slunits.TotalAtomicStoppingPower.....	1784
Modelica.Slunits.TotalMassStoppingPower.....	1784
Modelica.Slunits.MeanLinearRange.....	1784
Modelica.Slunits.MeanMassRange.....	1784
Modelica.Slunits.LinearIonization.....	1784
Modelica.Slunits.TotalIonization.....	1784
Modelica.Slunits.Mobility.....	1784
Modelica.Slunits.IonNumberDensity.....	1784
Modelica.Slunits.RecombinationCoefficient.....	1784
Modelica.Slunits.NeutronNumberDensity.....	1784
Modelica.Slunits.NeutronSpeed.....	1784
Modelica.Slunits.NeutronFluenceRate.....	1784
Modelica.Slunits.TotalNeutronSourceDensity.....	1785

Modelica.Slunits.SlowingDownDensity.....	1785
Modelica.Slunits.ResonanceEscapeProbability.....	1785
Modelica.Slunits.Lethargy.....	1785
Modelica.Slunits.SlowingDownArea.....	1785
Modelica.Slunits.DiffusionArea.....	1785
Modelica.Slunits.MigrationArea.....	1785
Modelica.Slunits.SlowingDownLength.....	1785
Modelica.Slunits.DiffusionLength.....	1785
Modelica.Slunits.MigrationLength.....	1785
Modelica.Slunits.NeutronYieldPerFission.....	1785
Modelica.Slunits.NeutronYieldPerAbsorption.....	1785
Modelica.Slunits.FastFissionFactor.....	1785
Modelica.Slunits.ThermalUtilizationFactor.....	1785
Modelica.Slunits.NonLeakageProbability.....	1786
Modelica.Slunits.Reactivity.....	1786
Modelica.Slunits.ReactorTimeConstant.....	1786
Modelica.Slunits.EnergyImparted.....	1786
Modelica.Slunits.MeanEnergyImparted.....	1786
Modelica.Slunits.SpecificEnergyImparted.....	1786
Modelica.Slunits.AbsorbedDose.....	1786
Modelica.Slunits.DoseEquivalent.....	1786
Modelica.Slunits.AbsorbedDoseRate.....	1786
Modelica.Slunits.LinearEnergyTransfer.....	1786
Modelica.Slunits.Kerma.....	1786
Modelica.Slunits.KermaRate.....	1786
Modelica.Slunits.MassEnergyTransferCoefficient.....	1786
Modelica.Slunits.Exposure.....	1786
Modelica.Slunits.ExposureRate.....	1787
Modelica.Slunits.ReynoldsNumber.....	1787
Modelica.Slunits.EulerNumber.....	1787
Modelica.Slunits.FroudeNumber.....	1787
Modelica.Slunits.GrashofNumber.....	1787
Modelica.Slunits.WeberNumber.....	1787
Modelica.Slunits.MachNumber.....	1787
Modelica.Slunits.KnudsenNumber.....	1787
Modelica.Slunits.StrouhalNumber.....	1787
Modelica.Slunits.FourierNumber.....	1787
Modelica.Slunits.PecletNumber.....	1787
Modelica.Slunits.RayleighNumber.....	1787
Modelica.Slunits.NusseltNumber.....	1787
Modelica.Slunits.BiotNumber.....	1787
Modelica.Slunits.StantonNumber.....	1788
Modelica.Slunits.FourierNumberOfMassTransfer.....	1788
Modelica.Slunits.PecletNumberOfMassTransfer.....	1788
Modelica.Slunits.GrashofNumberOfMassTransfer.....	1788
Modelica.Slunits.NusseltNumberOfMassTransfer.....	1788
Modelica.Slunits.StantonNumberOfMassTransfer.....	1788
Modelica.Slunits.PrandtlNumber.....	1788
Modelica.Slunits.SchmidtNumber.....	1788
Modelica.Slunits.LewisNumber.....	1788
Modelica.Slunits.MagneticReynoldsNumber.....	1788
Modelica.Slunits.AlfvenNumber.....	1788
Modelica.Slunits.HartmannNumber.....	1788
Modelica.Slunits.CowlingNumber.....	1788
Modelica.Slunits.BraggAngle.....	1788
Modelica.Slunits.OrderOfReflexion.....	1789
Modelica.Slunits.ShortRangeOrderParameter.....	1789
Modelica.Slunits.LongRangeOrderParameter.....	1789

Modelica.Slunits.DebyeWallerFactor.....	1789
Modelica.Slunits.CircularWavenumber.....	1789
Modelica.Slunits.FermiCircularWavenumber.....	1789
Modelica.Slunits.DebyeCircularWavenumber.....	1789
Modelica.Slunits.DebyeCircularFrequency.....	1789
Modelica.Slunits.DebyeTemperature.....	1789
Modelica.Slunits.SpectralConcentration.....	1789
Modelica.Slunits.GrueneisenParameter.....	1789
Modelica.Slunits.MadelungConstant.....	1789
Modelica.Slunits.DensityOfStates.....	1789
Modelica.Slunits.ResidualResistivity.....	1789
Modelica.Slunits.LorenzCoefficient.....	1790
Modelica.Slunits.HallCoefficient.....	1790
Modelica.Slunits.ThermoelectromotiveForce.....	1790
Modelica.Slunits.SeebeckCoefficient.....	1790
Modelica.Slunits.PeltierCoefficient.....	1790
Modelica.Slunits.ThomsonCoefficient.....	1790
Modelica.Slunits.RichardsonConstant.....	1790
Modelica.Slunits.FermiEnergy.....	1790
Modelica.Slunits.GapEnergy.....	1790
Modelica.Slunits.DonorIonizationEnergy.....	1790
Modelica.Slunits.AcceptorIonizationEnergy.....	1790
Modelica.Slunits.FermiTemperature.....	1790
Modelica.Slunits.ElectronNumberDensity.....	1790
Modelica.Slunits.HoleNumberDensity.....	1790
Modelica.Slunits.IntrinsicNumberDensity.....	1791
Modelica.Slunits.DonorNumberDensity.....	1791
Modelica.Slunits.AcceptorNumberDensity.....	1791
Modelica.Slunits.EffectiveMass.....	1791
Modelica.Slunits.MobilityRatio.....	1791
Modelica.Slunits.RelaxationTime.....	1791
Modelica.Slunits.CarrierLifeTime.....	1791
Modelica.Slunits.ExchangelIntegral.....	1791
Modelica.Slunits.CurieTemperature.....	1791
Modelica.Slunits.NeelTemperature.....	1791
Modelica.Slunits.LondonPenetrationDepth.....	1791
Modelica.Slunits.CoherenceLength.....	1791
Modelica.Slunits.LandauGinzburgParameter.....	1791
Modelica.Slunits.FluxoidQuantum.....	1791
Modelica.StateGraph.....	1792
Modelica.StateGraph.UsersGuide.....	1793
Modelica.StateGraph.Examples.....	1793
Modelica.StateGraph.Examples.FirstExample.....	1794
Modelica.StateGraph.Examples.FirstExample_Variant2.....	1794
Modelica.StateGraph.Examples.FirstExample_Variant3.....	1794
Modelica.StateGraph.Examples.ExecutionPaths.....	1794
Modelica.StateGraph.Examples.ShowCompositeStep.....	1794
Modelica.StateGraph.Examples.ShowExceptions.....	1794
Modelica.StateGraph.Examples.ControlledTanks.....	1795
Modelica.StateGraph.Examples.Utilities.....	1795
Modelica.StateGraph.Examples.Utilities.TankController.....	1796
Modelica.StateGraph.Examples.Utilities.MakeProduct.....	1796
Modelica.StateGraph.Examples.Utilities.inflow.....	1796
Modelica.StateGraph.Examples.Utilities.outflow.....	1797
Modelica.StateGraph.Examples.Utilities.valve.....	1797
Modelica.StateGraph.Examples.Utilities.Tank.....	1797
Modelica.StateGraph.Examples.Utilities.Source.....	1797
Modelica.StateGraph.Examples.Utilities.CompositeStep.....	1798

Modelica.StateGraph.Examples.Utilities.CompositeStep1.....	1798
Modelica.StateGraph.Examples.Utilities.CompositeStep2.....	1798
Modelica.StateGraph.Interfaces.....	1799
Modelica.StateGraph.Interfaces.Step_in.....	1799
Modelica.StateGraph.Interfaces.Step_out.....	1800
Modelica.StateGraph.Interfaces.Transition_in.....	1800
Modelica.StateGraph.Interfaces.Transition_out.....	1800
Modelica.StateGraph.Interfaces.CompositeStep_resume.....	1800
Modelica.StateGraph.Interfaces.CompositeStep_suspend.....	1801
Modelica.StateGraph.Interfaces.CompositeStepStatePort_in.....	1801
Modelica.StateGraph.Interfaces.CompositeStepStatePort_out.....	1801
Modelica.StateGraph.Interfaces.PartialStep.....	1802
Modelica.StateGraph.Interfaces.PartialTransition.....	1802
Modelica.StateGraph.Interfaces.PartialStateGraphIcon.....	1802
Modelica.StateGraph.Interfaces.CompositeStepState.....	1802
Modelica.StateGraph.InitialStep.....	1803
Modelica.StateGraph.InitialStepWithSignal.....	1803
Modelica.StateGraph.Step.....	1804
Modelica.StateGraph.StepWithSignal.....	1804
Modelica.StateGraph.Transition.....	1804
Modelica.StateGraph.TransitionWithSignal.....	1805
Modelica.StateGraph.Alternative.....	1805
Modelica.StateGraph.Parallel.....	1806
Modelica.StateGraph.PartialCompositeStep.....	1806
Modelica.StateGraph.StateGraphRoot.....	1806
Modelica.StateGraph.Temporary.....	1807
Modelica.StateGraph.Temporary.SetRealParameter.....	1807
Modelica.StateGraph.Temporary.anyTrue.....	1808
Modelica.StateGraph.Temporary.allTrue.....	1808
Modelica.StateGraph.Temporary.RadioButton.....	1808
Modelica.StateGraph.Temporary.NumericValue.....	1809
Modelica.StateGraph.Temporary.IndicatorLamp.....	1809
Modelica.Thermal.....	1809
Modelica.Thermal.FluidHeatFlow.....	1810
Modelica.Thermal.FluidHeatFlow.Examples.....	1811
Modelica.Thermal.FluidHeatFlow.Examples.SimpleCooling.....	1812
Modelica.Thermal.FluidHeatFlow.Examples.ParallelCooling.....	1812
Modelica.Thermal.FluidHeatFlow.Examples.IndirectCooling.....	1813
Modelica.Thermal.FluidHeatFlow.Examples.PumpAndValve.....	1813
Modelica.Thermal.FluidHeatFlow.Examples.PumpDropOut.....	1814
Modelica.Thermal.FluidHeatFlow.Examples.ParallelPumpDropOut.....	1814
Modelica.Thermal.FluidHeatFlow.Examples.OneMass.....	1814
Modelica.Thermal.FluidHeatFlow.Examples.TwoMass.....	1815
Modelica.Thermal.FluidHeatFlow.Examples.Utilities.....	1815
Modelica.Thermal.FluidHeatFlow.Examples.Utilities.DoubleRamp.....	1815
Modelica.Thermal.FluidHeatFlow.Components.....	1816
Modelica.Thermal.FluidHeatFlow.Components.IsolatedPipe.....	1816
Modelica.Thermal.FluidHeatFlow.Components.HeatedPipe.....	1817
Modelica.Thermal.FluidHeatFlow.Components.Valve.....	1818
Modelica.Thermal.FluidHeatFlow.Interfaces.....	1818
Modelica.Thermal.FluidHeatFlow.Interfaces.FlowPort.....	1819
Modelica.Thermal.FluidHeatFlow.Interfaces.FlowPort_a.....	1819
Modelica.Thermal.FluidHeatFlow.Interfaces.FlowPort_b.....	1820
Modelica.Thermal.FluidHeatFlow.Interfaces.Partials.....	1820
Modelica.Thermal.FluidHeatFlow.Interfaces.Partials.SimpleFriction.....	1821
Modelica.Thermal.FluidHeatFlow.Interfaces.Partials.TwoPort.....	1822
Modelica.Thermal.FluidHeatFlow.Interfaces.Partials.Ambient.....	1822
Modelica.Thermal.FluidHeatFlow.Interfaces.Partials.AbsoluteSensor.....	1822

---

Modelica.Thermal.FluidHeatFlow.Interfaces.Partials.RelativeSensor.....	1823
Modelica.Thermal.FluidHeatFlow.Interfaces.Partials.FlowSensor.....	1823
Modelica.Thermal.FluidHeatFlow.Media.....	1824
Modelica.Thermal.FluidHeatFlow.Media.Medium.....	1824
Modelica.Thermal.FluidHeatFlow.Media.Air_30degC.....	1824
Modelica.Thermal.FluidHeatFlow.Media.Air_70degC.....	1825
Modelica.Thermal.FluidHeatFlow.Media.Water.....	1825
Modelica.Thermal.FluidHeatFlow.Sensors.....	1825
Modelica.Thermal.FluidHeatFlow.Sensors.pSensor.....	1826
Modelica.Thermal.FluidHeatFlow.Sensors.TSensor.....	1826
Modelica.Thermal.FluidHeatFlow.Sensors.dpSensor.....	1827
Modelica.Thermal.FluidHeatFlow.Sensors.dTSensor.....	1827
Modelica.Thermal.FluidHeatFlow.Sensors.m_flowSensor.....	1828
Modelica.Thermal.FluidHeatFlow.Sensors.V_flowSensor.....	1828
Modelica.Thermal.FluidHeatFlow.Sensors.H_flowSensor.....	1828
Modelica.Thermal.FluidHeatFlow.Sources.....	1829
Modelica.Thermal.FluidHeatFlow.Sources.Ambient.....	1829
Modelica.Thermal.FluidHeatFlow.Sources.PrescribedAmbient.....	1830
Modelica.Thermal.FluidHeatFlow.Sources.AbsolutePressure.....	1830
Modelica.Thermal.FluidHeatFlow.Sources.ConstantVolumeFlow.....	1831
Modelica.Thermal.FluidHeatFlow.Sources.PrescribedVolumeFlow.....	1831
Modelica.Thermal.FluidHeatFlow.Sources.ConstantPressureIncrease.....	1832
Modelica.Thermal.FluidHeatFlow.Sources.PrescribedPressureIncrease.....	1832
Modelica.Thermal.FluidHeatFlow.Sources.IdealPump.....	1833
Modelica.Thermal.HeatTransfer.....	1833
Modelica.Thermal.HeatTransfer.Examples.....	1835
Modelica.Thermal.HeatTransfer.Examples.TwoMasses.....	1836
Modelica.Thermal.HeatTransfer.Examples.ControlledTemperature.....	1836
Modelica.Thermal.HeatTransfer.Examples.Motor.....	1837
Modelica.Thermal.HeatTransfer.Interfaces.....	1837
Modelica.Thermal.HeatTransfer.Interfaces.HeatPort.....	1837
Modelica.Thermal.HeatTransfer.Interfaces.HeatPort_a.....	1838
Modelica.Thermal.HeatTransfer.Interfaces.HeatPort_b.....	1838
Modelica.Thermal.HeatTransfer.Interfaces.Element1D.....	1839
Modelica.Thermal.HeatTransfer.HeatCapacitor.....	1839
Modelica.Thermal.HeatTransfer.ThermalConductor.....	1840
Modelica.Thermal.HeatTransfer.Convection.....	1841
Modelica.Thermal.HeatTransfer.BodyRadiation.....	1842
Modelica.Thermal.HeatTransfer.FixedTemperature.....	1843
Modelica.Thermal.HeatTransfer.PrescribedTemperature.....	1843
Modelica.Thermal.HeatTransfer.FixedHeatFlow.....	1844
Modelica.Thermal.HeatTransfer.PrescribedHeatFlow.....	1844
Modelica.Thermal.HeatTransfer.TemperatureSensor.....	1844
Modelica.Thermal.HeatTransfer.RelTemperatureSensor.....	1845
Modelica.Thermal.HeatTransfer.HeatFlowSensor.....	1845
Modelica.Thermal.HeatTransfer.Celsius.....	1845
Modelica.Thermal.HeatTransfer.Celsius.ToKelvin.....	1846
Modelica.Thermal.HeatTransfer.Celsius.FromKelvin.....	1846
Modelica.Thermal.HeatTransfer.Celsius.FixedTemperature.....	1847
Modelica.Thermal.HeatTransfer.Celsius.PrescribedTemperature.....	1847
Modelica.Thermal.HeatTransfer.Celsius.TemperatureSensor.....	1847
Modelica.Thermal.HeatTransfer.Fahrenheit.....	1848
Modelica.Thermal.HeatTransfer.Fahrenheit.ToKelvin.....	1848
Modelica.Thermal.HeatTransfer.Fahrenheit.FromKelvin.....	1848
Modelica.Thermal.HeatTransfer.Fahrenheit.FixedTemperature.....	1849
Modelica.Thermal.HeatTransfer.Fahrenheit.PrescribedTemperature.....	1849
Modelica.Thermal.HeatTransfer.Fahrenheit.TemperatureSensor.....	1849
Modelica.Thermal.HeatTransfer.Rankine.....	1850

---

Modelica.Thermal.HeatTransfer.Rankine.ToKelvin.....	1850
Modelica.Thermal.HeatTransfer.Rankine.FromKelvin.....	1851
Modelica.Thermal.HeatTransfer.Rankine.FixedTemperature.....	1851
Modelica.Thermal.HeatTransfer.Rankine.PrescribedTemperature.....	1851
Modelica.Thermal.HeatTransfer.Rankine.TemperatureSensor.....	1852
Modelica.Utilities.....	1852
Modelica.Utilities.UsersGuide.....	1853
Modelica.Utilities.Examples.....	1853
Modelica.Utilities.Examples.calculator.....	1854
Modelica.Utilities.Examples.expression.....	1854
Modelica.Utilities.Examples.readRealParameter.....	1856
Modelica.Utilities.Examples.readRealParameterModel.....	1857
Modelica.Utilities.Files.....	1857
Modelica.Utilities.Files.list.....	1858
Modelica.Utilities.Files.copy.....	1858
Modelica.Utilities.Files.move.....	1859
Modelica.Utilities.Files.remove.....	1860
Modelica.Utilities.Files.removeFile.....	1860
Modelica.Utilities.Files.createDirectory.....	1861
Modelica.Utilities.Files.exist.....	1861
Modelica.Utilities.Files.assertNew.....	1862
Modelica.Utilities.Files fullPathName.....	1862
Modelica.Utilities.Files.splitPathName.....	1863
Modelica.Utilities.Files.temporaryFileName.....	1863
Modelica.Utilities.Streams.....	1864
Modelica.Utilities.Streams.print.....	1865
Modelica.Utilities.Streams.readFile.....	1865
Modelica.Utilities.Streams.readLine.....	1866
Modelica.Utilities.Streams.countLines.....	1866
Modelica.Utilities.Streams.error.....	1867
Modelica.Utilities.Streams.close.....	1868
Modelica.Utilities.Strings.....	1868
Modelica.Utilities.Strings.length.....	1869
Modelica.Utilities.Strings.substring.....	1870
Modelica.Utilities.Strings.repeat.....	1871
Modelica.Utilities.Strings.compare.....	1871
Modelica.Utilities.Strings isEqual.....	1872
Modelica.Utilities.Strings.count.....	1872
Modelica.Utilities.Strings.find.....	1873
Modelica.Utilities.Strings.findLast.....	1874
Modelica.Utilities.Strings.replace.....	1874
Modelica.Utilities.Strings.sort.....	1875
Modelica.Utilities.Strings.scanToken.....	1876
Modelica.Utilities.Strings.scanReal.....	1877
Modelica.Utilities.Strings.scanInteger.....	1878
Modelica.Utilities.Strings.scanBoolean.....	1878
Modelica.Utilities.Strings.scanString.....	1879
Modelica.Utilities.Strings.scanIdentifier.....	1880
Modelica.Utilities.Strings.scanDelimiter.....	1880
Modelica.Utilities.Strings.scanNoToken.....	1881
Modelica.Utilities.Strings.syntaxError.....	1881
Modelica.Utilities.Strings.Advanced.....	1882
Modelica.Utilities.Strings.Advanced.scanReal.....	1883
Modelica.Utilities.Strings.Advanced.scanInteger.....	1884
Modelica.Utilities.Strings.Advanced.scanString.....	1885
Modelica.Utilities.Strings.Advanced.scanIdentifier.....	1886
Modelica.Utilities.Strings.Advanced.skipWhiteSpace.....	1886
Modelica.Utilities.Strings.Advanced.skipLineComments.....	1887

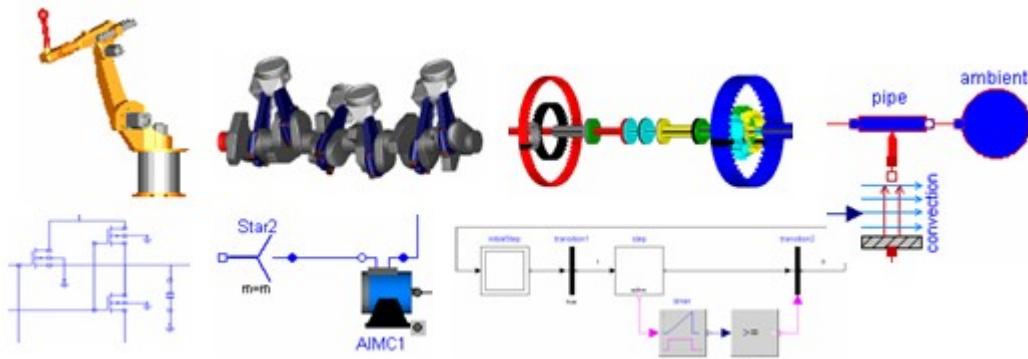
Modelica.Utilities.System.....	1888
Modelica.Utilities.System.getWorkDirectory.....	1888
Modelica.Utilities.System.setWorkDirectory.....	1888
Modelica.Utilities.System.getEnvironmentVariable.....	1888
Modelica.Utilities.System.setEnvironmentVariable.....	1889
Modelica.Utilities.System.command.....	1889
Modelica.Utilities.System.exit.....	1889
Modelica.Utilities.Types.....	1890
Modelica.Utilities.Types.Compare.....	1890
Modelica.Utilities.Types.Compare.Type.....	1891
Modelica.Utilities.Types.FileType.....	1891
Modelica.Utilities.Types.FileType.Type.....	1892
Modelica.Utilities.Types.TokenType.....	1892
Modelica.Utilities.Types.TokenType.Type.....	1893
Modelica.Utilities.Types.TokenValue.....	1893

## Modelica

### Modelica Standard Library

#### Information

Package **Modelica** is a **standardized** and **free** package that is developed together with the Modelica language from the Modelica Association, see <http://www.Modelica.org>. It is also called **Modelica Standard Library**. It provides model components in many domains that are based on standardized interface definitions. Some typical examples are shown in the next figure:



For an introduction, have especially a look at:

- [Overview](#) provides an overview of the Modelica Standard Library inside the [User's Guide](#),
- [Release Notes](#) summarizes the changes of new versions of this package,
- [Contact](#) lists the contributors of the Modelica Standard Library.
- The [Examples](#) packages in the various libraries, demonstrate how to use the components of the corresponding sublibrary.

This version of the Modelica Standard Library consists of

- **739** models and blocks, and
- **539** functions

that are directly usable (= number of public, non-partial classes).

Copyright © 1998-2007, Modelica Association.

*This Modelica package is **free** software; it can be redistributed and/or modified under the terms of the **Modelica license**, see the license conditions and the accompanying [disclaimer](#) [here](#).*

#### Package Content

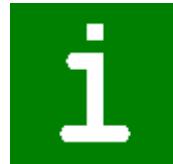
Name	Description
<a href="#">UsersGuide</a>	User's Guide of Modelica library
<a href="#">Blocks</a>	Library of basic input/output control blocks (continuous, discrete, logical, table blocks)
<a href="#">Constants</a>	Library of mathematical constants and constants of nature (e.g., pi, eps, R, sigma)
<a href="#">Electrical</a>	Library of electrical models (analog, digital, machines, multi-phase)
<a href="#">Icons</a>	Library of icons
<a href="#">Math</a>	Library of mathematical functions (e.g., sin, cos) and of functions operating on vectors and matrices
<a href="#">Mechanics</a>	Library of 1-dim. and 3-dim. mechanical components (multi-body, rotational, translational)

 Media	Library of media property models
 Slunits	Library of type and unit definitions based on SI units according to ISO 31-1992
 StateGraph	Library of hierarchical state machine components to model discrete event and reactive systems
 Thermal	Library of thermal system components to model heat transfer and simple thermo-fluid pipe flow
 Utilities	Library of utility functions dedicated to scripting (operating on files, streams, strings, system)

## Modelica.UsersGuide

### User's Guide of Modelica library

Package **Modelica** is a **standardized** and **pre-defined** package that is developed together with the Modelica language from the Modelica Association, see <http://www.Modelica.org>. It is also called **Modelica Standard Library**. It provides constants, types, connectors, partial models and model components in various disciplines.



This is a short **user's guide** for the overall library. Some of the main sublibraries have their own user's guides that can be accessed by the following links:

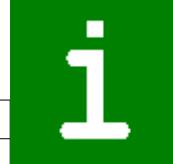
Digital	Library for digital electrical components based on the VHDL standard (2-,3-,4-,9-valued logic)
MultiBody	Library to model 3-dimensional mechanical systems
Rotational	Library to model 1-dimensional mechanical systems
Media	Property models of media
Slunits	Type definitions based on SI units according to ISO 31-1992
StateGraph	Library to model discrete event and reactive systems by hierarchical state machines
Utilities	Utility functions especially for scripting (Files, Streams, Strings, System)

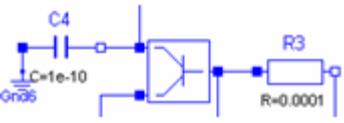
### Package Content

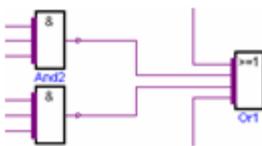
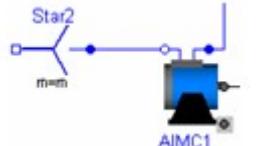
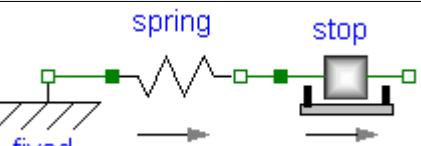
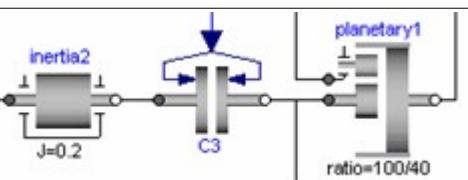
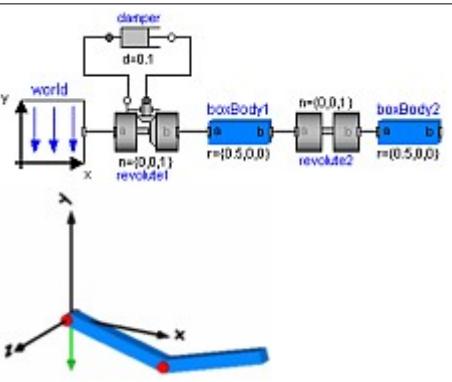
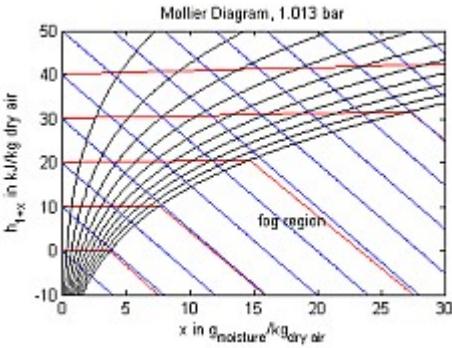
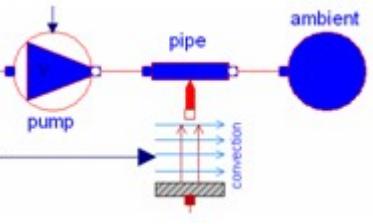
Name	Description
 Overview	Overview of Modelica Library
 Connectors	Connectors
 Conventions	Conventions
 ReleaseNotes	Release notes
 Contact	Contact
 ModelicaLicense	Modelica License (Version 1.1 of June 30, 2000)

## Modelica.UsersGuide.Overview

The Modelica Standard Library consists of the following main sub-libraries:



Library Components	Description
	<b>Analog</b> Analog electric and electronic components, such as resistor, capacitor, transformers, diodes, transistors, transmission lines, switches, sources, sensors.

	<b>Digital</b> Digital electrical components based on the VHDL standard, like basic logic blocks with 9-value logic, delays, gates, sources, converters between 2-, 3-, 4-, and 9-valued logic.
	<b>Machines</b> Electrical asynchronous-, synchronous-, and DC-machines (motors and generators) as well as 3-phase transformers.
	<b>Translational</b> 1-dim. mechanical, translational systems, e.g., sliding mass, mass with stops, spring, damper.
	<b>Rotational</b> 1-dim. mechanical, rotational systems, e.g., inertias, gears, planetary gears, convenient definition of speed/torque dependent friction (clutches, brakes, bearings, ...)
	<b>MultiBody</b> 3-dim. mechanical systems consisting of joints, bodies, force and sensor elements. Joints can be driven by drive trains defined by 1-dim. mechanical system library (Rotational). Every component has a default animation. Components can be arbitrarily connected together.
	<b>Media</b> Large media library providing models and functions to compute media properties, such as $h = h(p, T)$ , $d = d(p, T)$ , for the following media: <ul style="list-style-type: none"> <li>• 1240 gases and mixtures between these gases.</li> <li>• incompressible, table based liquids (<math>h = h(T)</math>, etc.).</li> <li>• compressible liquids</li> <li>• dry and moist air</li> <li>• high precision model for water (IF97).</li> </ul>
	<b>FluidHeatFlow, HeatTransfer</b> Simple thermo-fluid pipe flow, especially to model cooling of machines with air or water (pipes, pumps, valves, ambient, sensors, sources) and lumped heat transfer with heat capacitors, thermal conductors, convection, body radiation, sources and sensors.

<pre> graph LR     subgraph "kinematicPTP"         acc[acc]         deltaq1[deltaq=(1)]     end     subgraph "integrator"         I[integrator k=1]     end     subgraph "PI"         PID[PID]     end     subgraph "Logic"         table2[table2]         Not1[Not1]         And1[And1]         Pre1[Pre1]         pre[pre]     end     acc --&gt; deltaq1     deltaq1 --&gt; I     I --&gt; PID     table2 --&gt; Not1     Not1 --&gt; And1     And1 --&gt; Pre1     Pre1 --&gt; pre     pre --&gt; PID </pre>	<p><b>Blocks</b> Input/output blocks to model block diagrams and logical networks, e.g., integrator, PI, PID, transfer function, linear state space system, sampler, unit delay, discrete transfer function, and/or blocks, timer, hysteresis, nonlinear and routing blocks, sources, tables.</p>
<pre> graph LR     initialStep[initialStep] --&gt; transition1[transition1]     transition1 --&gt; step[step]     step --&gt; transition2[transition2]     transition2 --&gt; initialStep </pre>	<p><b>StateGraph</b> Hierarchical state machines with the same modeling power as Statecharts. Modelica is used as synchronous action language, i.e. deterministic behavior is guaranteed (not the case for Statecharts)</p>
<pre> A = [1,2,3;       3,4,5;       2,1,4]; b = {10,22,12}; x = Matrices.solve(A,b); Matrices.eigenValues(A); </pre>	<p><b>Math, Utilities</b> Functions operating on vectors and matrices, such as for solving linear systems, eigen and singular values etc., and functions operating on strings, streams, files, e.g., to copy and remove a file or sort a vector of strings.</p>

## Modelica.UsersGuide.Connectors

The Modelica standard library defines the most important **elementary connectors** in various domains. If any possible, a user should utilize these connectors in order that components from the Modelica Standard Library and from other libraries can be combined without problems. The following elementary connectors are defined (potential variables are connector variables without the flow attribute, flow variables are connector variables that have the flow attribute):



domain	pot. variables	flow variables	connector definition	icons
electrical analog	electrical potential	electrical current	Modelica.Electrical.Analog.Interfaces Pin, PositivePin, NegativePin	
electrical multi-phase	vector of electrical pins		Modelica.Electrical.MultiPhase.Interfaces Plug, PositivePlug, NegativePlug	
electrical sphase phasor	2 electrical potentials	2 electrical currents	Modelica.Electrical.Machines.Interfaces SpacePhasor	
electrical digital	Integer (1..9)	---	Modelica.Electrical.Digital.Interfaces DigitalSignal, DigitalInput, DigitalOutput	
translational	distance	cut-force	Modelica.Mechanics.Translational.Interfaces Flange_a, Flange_b	
rotational	angle	cut-torque	Modelica.Mechanics.Rotational.Interfaces Flange_a, Flange_b	
3-dim. mechanics	position vector orientation object	cut-force vector cut-torque vector	Modelica.Mechanics.MultiBody.Interfaces Frame, Frame_a, Frame_b, Frame_resolve	

<b>simple fluid flow</b>	pressure specific enthalpy	mass flow rate enthalpy flow rate	<a href="#">Modelica.Thermal.FluidHeatFlow.Interfaces</a> FlowPort, FlowPort_a, FlowPort_b	
<b>heat transfer</b>	temperature	heat flow rate	<a href="#">Modelica.Thermal.HeatTransfer.Interfaces</a> HeatPort, HeatPort_a, HeatPort_b	
<b>block diagram</b>	Real variable Integer variable Boolean variable	---	<a href="#">Modelica.Blocks.Interfaces</a> RealSignal, RealInput, RealOutput IntegerSignal, IntegerInput, IntegerOutput BooleanSignal, BooleanInput, BooleanOutput	  
<b>state machine</b>	Boolean variables (occupied, set, available, reset)	---	<a href="#">Modelica.StateGraph.Interfaces</a> Step_in, Step_out, Transition_in, Transition_out	

**Connectors from libraries that will be included in one of the next releases of package Modelica**

<b>thermo fluid flow</b>	pressure specific enthalpy mass fractions	mass flow rate enthalpy flow rate subst. mass flow rates	<a href="#">Modelica_Fluid.Interfaces</a> FluidPort, FluidPort_a, FluidPort_b	
<b>magnetic</b>	magnetic potential	magnetic flux	<a href="#">Magnetic.Interfaces</a> MagneticPort, PositiveMagneticPort, NegativeMagneticPort	

**Connectors from other libraries**

<b>hydraulic</b>	pressure	volume flow rate	<a href="#">HyLibLight.Interfaces</a> Port_A, Port_b	
<b>pneumatic</b>	pressure	mass flow rate	<a href="#">PneuLibLight.Interfaces</a> Port_1, Port_2	

In all domains, usually 2 connectors are defined. The variable declarations are **identical**, only the icons are different in order that it is easy to distinguish connectors of the same domain that are attached at the same component.

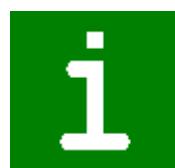
Modelica supports also hierarchical connectors, in a similar way as hierarchical models. As a result, it is, e.g., possible, to collect elementary connectors together. For example, an electrical plug consisting of two electrical pins can be defined as:

```
connector Plug
  import Modelica.Electrical.Analog.Interfaces;
  Interfaces.PositivePin phase;
  Interfaces.NegativePin ground;
end Plug;
```

With one connect(..) equation, either two plugs can be connected (and therefore implicitly also the phase and ground pins) or a Pin connector can be directly connected to the phase or ground of a Plug connector, such as "connect(resistor.p, plug.phase)".

**Modelica.UsersGuide.Conventions**

Note, in the html documentation of any Modelica library, the headings "h1, h2, h3" should not be used, because they are utilized from the automatically generated documentation/headings. Additional headings in the html documentation should start with "h4".



In the Modelica package the following conventions are used:

1. Class and instance names are written in upper and lower case letters, e.g., "ElectricCurrent". An underscore is only used at the end of a name to characterize a lower or upper index, e.g., "pin\_a".
2. **Class names** start always with an upper case letter.
3. **Instance names**, i.e., names of component instances and of variables (with the exception of constants), start usually with a lower case letter with only a few exceptions if this is common sense (such as "T" for a temperature variable).
4. **Constant names**, i.e., names of variables declared with the "constant" prefix, follow the usual naming conventions (= upper and lower case letters) and start usually with an upper case letter, e.g. UniformGravity, SteadyState.
5. The two connectors of a domain that have identical declarations and different icons are usually distinguished by "\_a", "\_b" or "\_p", "\_n", e.g., Flange\_a/Flange\_b, HeatPort\_a, HeatPort\_b.
6. The **instance name** of a component is always displayed in its icon (= text string "%name") in **blue color**. A connector class has the instance name definition in the diagram layer and not in the icon layer. **Parameter** values, e.g., resistance, mass, gear ratio, are displayed in the icon in **black color** in a smaller font size as the instance name.
7. A main package has usually the following subpackages:
  - **UsersGuide** containing an overall description of the library and how to use it.
  - **Examples** containing models demonstrating the usage of the library.
  - **Interfaces** containing connectors and partial models.
  - **Types** containing type, enumeration and choice definitions.

**Enumerations** are defined in the Modelica language since release 2.0. However, they are not yet supported in the most important Modelica simulation environment Dymola. For this reason, this language element is not used in the Modelica standard library. Instead, enumerations are emulated with packages and constants. For example, the enumeration

```
type Init = enumeration (NoInit, InitializeStates, SteadyState);

parameter Init initType = Init.NoInit "Type of initialization";
```

is emulated in the following way:

```
package Init "Enumeration emulation"
  extends Modelica.Icons.Enumeration;

  constant Integer NoInit=1;
  constant Integer InitializeStates=2;
  constant Integer SteadyState=3;

  type Temp
    extends Modelica.Icons.TypeInteger;
    annotation (choices(
      choice=Init.NoInit           "NoInit (no initialization)",
      choice=Init.InitializeStates "InitializeStates (initialize
states)",
      choice=Init.SteadyState      "SteadyState (initialize in steady
state)"));
    end Temp;
  end Init;

parameter Init.Temp initType = Init.NoInit;
```

## Modelica.UsersGuide.ReleaseNotes

This section summarizes the changes that have been performed on the Modelica standard library.



- [Version 2.2.2](#) (Aug. 12, 2007)
- [Version 2.2.1](#) (March 24, 2006)
- [Version 2.2](#) (April 6, 2005)
- [Version 2.1](#) (Nov. 11, 2004)
- [Version 1.6](#) (June 21, 2004)
- [Version 1.5](#) (Dec. 16, 2002)
- [Version 1.4](#) (June 28, 2001 and previous versions)

Maintenance of the Modelica Standard Library is performed with three branches on the subversion server of the Modelica Association:

### Released branch

Example: "Modelica/tags/V2\_2\_1/Modelica"

This branch contains the released Modelica versions (e.g. version 2.2.1), where all available test cases and compatibility checks with other Modelica libraries have been performed on the respective release. This version is usually shipped with a Modelica modelling and simulation environment and utilized by a Modelica user.

### Development branch

Example: "Modelica/trunk/Modelica"

This branch contains the actual development version, i.e., all bug fixes and new features based on the last Modelica release. New features should have been tested before including them. However, the exhaustive tests for a new version are (usually) not performed. This version is usually only be used by the developers of the Modelica Standard Library and is not utilized by Modelica users.

### Maintenance branch

Example: "Modelica/branches/maintenance/2.2.1/Modelica"

This branch contains the released Modelica version (e.g. version 2.2.1) where all bug fixes since this release date are included (up to a new release, when becoming available; i.e., after a new release, the previous maintenance versions are no longer changed). These bug fixes might be not yet tested with all test cases or with other Modelica libraries. The goal is that a vendor may take this version at any time for a new release of its software, in order to incorporate the latest bug fixes, without changing the version number of the Modelica Standard Library.

Incorporation of bug fixes (subversion "commit") shall be performed in the following way:

- One person is fixing the bug and another person is checking whether the fix is fine.
- It is up to the library developer, whether he opens a new branch for testing and then merges it with the "head" maintenance branch or not.
- Every change to the maintenance branch has to be done at the development branch (see above) as well.
- Every change to the maintenance branch requires introducing a description of the bug fix under Modelica.UsersGuide.ReleaseNotes.<release-number>\_bugFixes.
- Every change to the maintenance branch requires changing the date of the Modelica library in the version annotation.

When including the library in a distribution, the vendor has to add the subversion build number of the corresponding release in the version annotation. Example:

```
annotation(version="2.2.1", versionBuild="436", versionDate="2007-05-13")
```

The goal is to include the version build and version date information automatically from the subversion server, but this is not yet the case.

- If time does not permit, a vendor makes the bug fix in its local version and then has to include it in the maintenance version. It would be best to make these changes at a new branch in order to get a unique release number.

A valid "commit" to the maintenance branch may contain one or more of the following changes.

- Correcting an equation.
- Correcting attributes quantity/unit/defaultUnit in a declaration.
- Improving/fixing the documentation.
- Introducing a new name in the public section of a class (model, package, ...) or in any section of a partial class is **not** allowed. Since otherwise, a user might use this new name and when storing its model and loading it with an older build-version, an error would occur.
- Introducing a new name in the protected section of a non-partial class should only be done if absolutely necessary to fix a bug. The problem is that this might be non-backward compatible, because a user might already extend from this class and already using the same name.

---

## Modelica.UsersGuide.Contact

The development of the Modelica package is organized by

**Martin Otter**

Deutsches Zentrum für Luft und Raumfahrt e.V. (DLR)  
Institut für Robotik und Mechatronik  
Abteilung für Entwurfsorientierte Regelungstechnik  
Postfach 1116  
D-82230 Wessling  
Germany  
email: [Martin.Otter@dlr.de](mailto:Martin.Otter@dlr.de)



This library is developed by several people from the Modelica Association, see <http://www.Modelica.org>. In particular, the following people have directly contributed (many more people have contributed to the design):

<b>Peter Beater</b> University of Paderborn, Germany	Modelica.Mechanics.Translational
<b>Dag Brück</b> Dynasim AB, Lund, Sweden	Modelica.Utilities
<b>Francesco Casella</b> Politecnico di Milano, Milano, Italy	Modelica.Media
<b>Christoph Clauss</b> Fraunhofer Institute for Integrated Circuits, Dresden, Germany	Modelica.Electrical.Analog Modelica.Electrical.Digital
<b>Jonas Eborn</b> Modelon AB, Lund, Sweden	Modelica.Media
<b>Hilding Elmqvist</b> Dynasim AB, Lund, Sweden	Modelica.Mechanics.MultiBody Modelica.Media Modelica.StateGraph Modelica.Utilities Conversion from 1.6 to 2.0
<b>Rüdiger Franke</b> ABB Corporate Research, Ladenburg, German	Modelica.Media
<b>Anton Haumer</b> Consultant, St.Andrae-Woerdern, Austria	Modelica.Electrical.Machines Modelica.Electrical.Multiphase Modelica.Mechanics.Rotational Modelica.Thermal.FluidHeatFlow
<b>Hans-Dieter Joos</b> Institute of Robotics and Mechatronics DLR, German Aerospace Center, Oberpfaffenhofen, Germany	Modelica.Math.Matrices
<b>Christian Kral</b> arsenal research, Vienna, Austria	Modelica.Electrical.Machines

	Modelica.Thermal.FluidHeatFlow
<b>Sven Erik Mattsson</b> Dynasim AB, Lund, Sweden	Modelica.Mechanics.MultiBody
<b>Hans Olsson</b> Dynasim AB, Lund, Sweden	Modelica.Blocks Modelica.Math.Matrices Modelica.Utilities Conversion from 1.6 to 2.0
<b>Martin Otter</b> Institute of Robotics and Mechatronics DLR, German Aerospace Center, Oberpfaffenhofen, Germany	Modelica.Blocks Modelica.Mechanics.MultiBody Modelica.Mechanics.Rotational Modelica.Math Modelica.Media Modelica.Slunits Modelica.StateGraph Modelica.Thermal Modelica.Utilities ModelicaReference Conversion from 1.6 to 2.0
<b>Katrin Prölß</b> Department of Technical Thermodynamics, Technical University Hamburg-Harburg, Germany	Modelica.Media
<b>Christoph C. Richter</b> Institut für Thermodynamik, Technische Universität Braunschweig, Germany	Modelica.Media
<b>André Schneider</b> Fraunhofer Institute for Integrated Circuits, Dresden, Germany	Modelica.Electrical.Analog Modelica.Electrical.Digital
<b>Christian Schweiger</b> Institute of Robotics and Mechatronics, DLR, German Aerospace Center, Oberpfaffenhofen, Germany	Modelica.Mechanics.Rotational ModelicaReference Conversion from 1.6 to 2.0
<b>Michael Tiller</b> Emmeskay, Inc., Dearborn, MI, U.S.A, (previously Ford Motor Company, Dearborn)	Modelica.Media Modelica.Thermal
<b>Hubertus Tummescheit</b> Modelon AB, Lund, Sweden	Modelica.Media Modelica.Thermal
<b>Nico Walter</b> Master thesis at HTWK Leipzig (Prof. R. Müller) and DLR Oberpfaffenhofen, Germany	Modelica.Math.Matrices

## Modelica.UsersGuide.ModelicaLicense

Redistribution and use in source and binary forms, with or without modification are permitted, provided that the following conditions are met:

1. The author and copyright notices in the source files, these license conditions and the disclaimer below are (a) retained and (b) reproduced in the documentation provided with the distribution.
2. Modifications of the original source files are allowed, provided that a prominent notice is inserted in each changed file and the accompanying documentation, stating how and when the file was modified, and provided that the conditions under (1) are met.
3. It is not allowed to charge a fee for the original version or a modified version of the software, besides a reasonable fee for distribution and support. Distribution in aggregate with other (possibly commercial) programs as part of a larger (possibly commercial) software distribution is permitted, provided that it is not advertised as a product of your own.



## Disclaimer

The software (sources, binaries, etc.) in their original or in a modified form are provided "as is" and the copyright holders assume no responsibility for its contents what so ever. Any express or implied warranties, including, but not limited to, the implied warranties of merchantability and fitness for a particular purpose are **disclaimed**. In no event shall the copyright holders, or any party who modify and/or redistribute the package, **be liable** for any direct, indirect, incidental, special, exemplary, or consequential damages, arising in any way out of the use of this software, even if advised of the possibility of such damage.

---

## Modelica.Blocks

**Library of basic input/output control blocks (continuous, discrete, logical, table blocks)**

### Information

This library contains input/output blocks to build up block diagrams.

#### Main Author:

Martin Otter  
Deutsches Zentrum für Luft und Raumfahrt e. V. (DLR)  
Oberpfaffenhofen  
Postfach 1116  
D-82230 Wessling  
email: [Martin.Otter@dlr.de](mailto:Martin.Otter@dlr.de)

Copyright © 1998-2007, Modelica Association and DLR.

*This Modelica package is free software; it can be redistributed and/or modified under the terms of the Modelica license, see the license conditions and the accompanying disclaimer [here](#).*

### Package Content

Name	Description
 Examples	Library of examples to demonstrate the usage of package Blocks
 Continuous	Library of continuous control blocks with internal states
 Discrete	Library of discrete input/output blocks with fixed sample period
 Interfaces	Library of connectors and partial models for input/output blocks
 Logical	Library of components with Boolean input and output signals
 Math	Library of mathematical functions as input/output blocks
 Nonlinear	Library of discontinuous or non-differentiable algebraic control blocks
 Routing	Library of blocks to combine and extract signals
 Sources	Library of signal source blocks generating Real and Boolean signals
 Tables	Library of blocks to interpolate in one and two-dimensional tables
 Types	Library of constants and types with choices, especially to build menus

---

## Modelica.Blocks.Examples

**Library of examples to demonstrate the usage of package Blocks**

## Information

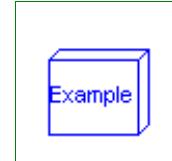
This package contains example models to demonstrate the usage of package blocks.

## Package Content

Name	Description
PID_Controller	Demonstrate usage of the Continuous.LimPID controller
ShowLogicalSources	Show logical sources and demonstrate their diagram animation
LogicalNetwork1	Example for a network of logical blocks
BusUsage	Demonstration of signal bus usage
BusUsage_Utilities	Utility models and connectors for the demonstration example Modelica.Blocks.Examples.BusUsage

### Modelica.Blocks.Examples.PID\_Controller

Demonstrate usage of the Continuous.LimPID controller



## Information

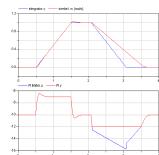
This is a simple drive train controlled by a PID controller:

- The two blocks "kinematic\_PTP" and "integrator" are used to generate the reference speed (= constant acceleration phase, constant speed phase, constant deceleration phase until inertia is at rest). To check whether the system starts in steady state, the reference speed is zero until time = 0.5 s and then follows the sketched trajectory.
- The block "PI" is an instance of "Blocks.Continuous.LimPID" which is a PID controller where several practical important aspects, such as anti-windup-compensation has been added. In this case, the control block is used as PI controller.
- The output of the controller is a torque that drives a motor inertia "inertia1". Via a compliant spring/damper component, the load inertia "inertia2" is attached. A constant external torque of 10 Nm is acting on the load inertia.

The PI controller settings included "limitAtInit=false", in order that the controller output limits of 12 Nm are removed from the initialization problem.

The PI controller is initialized in steady state (initType=SteadyState) and the drive shall also be initialized in steady state. However, it is not possible to initialize "inertia1" in SteadyState, because "der(inertia1.phi)=inertia1.w=0" is an input to the PI controller that defines that the derivative of the integrator state is zero (= the same condition that was already defined by option SteadyState of the PI controller). Furthermore, one initial condition is missing, because the absolute position of inertia1 or inertia2 is not defined. The solution shown in this examples is to initialize the angle and the angular acceleration of "inertia1".

In the following figure, results of a typical simulation are shown:



In the upper figure the reference speed (= integrator.y) and the actual speed (= inertia1.w) are shown. As can be seen, the system initializes in steady state, since no transients are present. The inertia follows the reference speed quite good until the end of the constant speed phase. Then there is a deviation. In the lower figure the reason can be seen: The output of the controller (PI.y) is in its limits. The anti-windup compensation works reasonably, since the input to the limiter (PI.limiter.u) is forced back to its limit after a

transient phase.

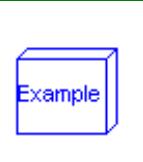
## Parameters

Name	Default	Description
driveAngle	1.57	Reference distance to move [rad]

---

## Modelica.Blocks.Examples.ShowLogicalSources

Show logical sources and demonstrate their diagram animation



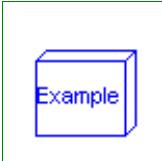
### Information

This simple example demonstrates the logical sources in [Modelica.Blocks.Sources](#) and demonstrate their diagram animation (see "small circle" close to the output connector). The "booleanExpression" source shows how a logical expression can be defined in its parameter menu referring to variables available on this level of the model.

---

## Modelica.Blocks.Examples.LogicalNetwork1

Example for a network of logical blocks



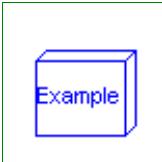
### Information

This example demonstrates a network of logical blocks. Note, that the Boolean values of the input and output signals are visualized in the diagram animation, by the small "circles" close to the connectors. If a "circle" is "white", the signal is **false**. If a "circle" is "green", the signal is **true**.

---

## Modelica.Blocks.Examples.BusUsage

Demonstration of signal bus usage



### Information

#### Signal bus concept

In technical systems, such as vehicles, robots or satellites, many signals are exchanged between components. In a simulation system, these signals are usually modelled by signal connections of input/output blocks. Unfortunately, the signal connection structure may become very complicated, especially for hierarchical models.

The same is also true for real technical systems. To reduce complexity and get higher flexibility, many technical systems use data buses to exchange data between components. For the same reasons, it is often better to use a "signal bus" concept also in a Modelica model. This is demonstrated at hand of this model ([Modelica.Blocks.Examples.BusUsage](#)):



- Connector instance "controlBus" is a hierarchical connector that is used to exchange signals between different components. It is defined as "expandable connector" in order that **no** central definition of the connector is needed but is automatically constructed by the signals connected to it (see also Modelica specification 2.2.1).
- Input/output signals can be directly connected to the "controlBus". When connecting, it is optionally

possible that a **label** is displayed at the connecting line, that contains the name of the variable on the controlBus to which the signal is connected.

- A component, such as "part", can be directly connected to the "controlBus", provided it has also a bus connector, or the "part" connector is a sub-connector contained in the "controlBus".

The control and sub-control bus icons are provided within Modelica.Icons. In [Modelica.Blocks.Examples.BusUsage\\_Utility.Interfaces](#) the buses for this example are defined. Both the "ControlBus" and the "SubControlBus" are **expandable** connectors that do not define any variable. For example, [Interfaces.ControlBus](#) is defined as:

```
expandable connector ControlBus
  extends Modelica.Icons.ControlBus;
  annotation (Icon(Rectangle(extent=[-20, 2; 22, -2],
    style(rgbcolor={255,204,51}, thickness=2))) );
end ControlBus;
```

Note, the "annotation" in the connector is important since the color and thickness of a connector line are taken from the first line element in the icon annotation of a connector class. Above, a small rectangle in the color of the bus is defined (and therefore this rectangle is not visible). As a result, when connecting from an instance of this connector to another connector instance, the connecting line has the color of the "ControlBus" with double width (due to "thickness=2").

An **expandable** connector is a connector where the content of the connector is constructed by the variables connected to instances of this connector. For example, if "sine.y" is connected to the "controlBus", the following menu pops-up in Dymola:



The "Add variable/New name" field allows the user to define the name of the signal on the "controlBus". When typing "realSignal1" as "New name", a connection of the form:

```
connect(sine.y, controlBus.realSignal1)
```

is generated and the "controlBus" contains the new signal "realSignal1". Modelica tools may give more support in order to list potential signals for a connection. For example, in Dymola all variables are listed in the menu that are contained in connectors which are derived by inheritance from "controlBus". Therefore, in [Modelica.Blocks.Examples.BusUsage\\_Utility.Interfaces.Internal](#) the expected implementation of the "ControlBus" and of the "SubControlBus" are given. For example "Internal.ControlBus" is defined as:

```
expandable connector StandardControlBus
  extends BusUsage_Utility.Interfaces.ControlBus;

  import SI = Modelica.SIunits;
  SI.AngularVelocity realSignal1 "First Real signal";
  SI.Velocity realSignal2 "Second Real signal";
  Integer integerSignal "Integer signal";
  Boolean booleanSignal "Boolean signal";
  StandardSubControlBus subControlBus "Combined signal";
end StandardControlBus;
```

Consequently, when connecting now from "sine.y" to "controlBus", the menu looks differently:



Note, even if the signals from "Internal.StandardControlBus" are listed, these are just potential signals. The user might still add different signal names.

**Modelica.Blocks.Examples.BusUsage\_Utility****Utility models and connectors for the demonstration example Modelica.Blocks.Examples.BusUsage****Information**

This package contains utility models and bus definitions needed for the [BusUsage](#) example.

**Package Content**

Name	Description
 Interfaces	Interfaces specialised for this example
 Part	Component with sub-control bus

---

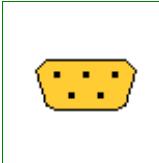
**Modelica.Blocks.Examples.BusUsage\_Utility.Interfaces****Interfaces specialised for this example****Information**

This package contains the bus definitions needed for the [BusUsage](#) example.

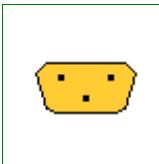
**Package Content**

Name	Description
 ControlBus	Empty control bus that is adapted to the signals connected to it
 SubControlBus	Empty sub-control bus that is adapted to the signals connected to it

---

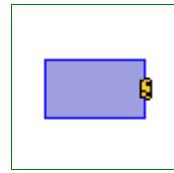
**Modelica.Blocks.Examples.BusUsage\_Utility.Interfaces.ControlBus****Empty control bus that is adapted to the signals connected to it****Information**

This connector defines the "expandable connector" ControlBus that is used as bus in the [BusUsage](#) example. Note, this connector is "empty". When using it, the actual content is constructed by the signals connected to this bus.

**Modelica.Blocks.Examples.BusUsage\_Utility.Interfaces.SubControlBus****Empty sub-control bus that is adapted to the signals connected to it****Information**

This connector defines the "expandable connector" SubControlBus that is used as sub-bus in the [BusUsage](#) example. Note, this connector is "empty". When using it, the actual content is constructed by the signals connected to this bus.

---

**Modelica.Blocks.Examples.BusUsage\_Utilitys.Part****Component with sub-control bus****Information**

This model is used to demonstrate the bus usage in example BusUsage.

**Connectors**

Name	Description
subControlBus	

**Modelica.Blocks.Continuous****Library of continuous control blocks with internal states****Information**

This package contains basic **continuous** input/output blocks described by differential equations.

All blocks of this package can be initialized in different ways controlled by parameter **initType**. The possible values of initType are defined in [Modelica.Blocks.Types.Init](#):

Name	Description
<b>Init.NoInit</b>	no initialization (start values are used as guess values with fixed=false)
<b>Init.SteadyState</b>	steady state initialization (derivatives of states are zero)
<b>Init.InitialState</b>	Initialization with initial states
<b>Init.InitialOutput</b>	Initialization with initial outputs (and steady state of the states if possible)

For backward compatibility reasons the default of all blocks is **Init.NoInit**, with the exception of Integrator and LimIntegrator where the default is **Init.InitialState** (this was the initialization defined in version 2.2 of the Modelica standard library).

In many cases, the most useful initial condition is **Init.SteadyState** because initial transients are then no longer present. The drawback is that in combination with a non-linear plant, non-linear algebraic equations occur that might be difficult to solve if appropriate guess values for the iteration variables are not provided (i.e. start values with fixed=false). However, it is often already useful to just initialize the linear blocks from the Continuous blocks library in SteadyState. This is uncritical, because only linear algebraic equations occur. If **Init.NoInit** is set, then the start values for the states are interpreted as **guess** values and are propagated to the states with fixed=false.

Note, initialization with **Init.SteadyState** is usually difficult for a block that contains an integrator (Integrator, LimIntegrator, PI, PID, LimPID). This is due to the basic equation of an integrator:

```

initial equation
  der(y) = 0;    // Init.SteadyState
equation
  der(y) = k*u;

```

The steady state equation leads to the condition that the input to the integrator is zero. If the input u is already (directly or indirectly) defined by another initial condition, then the initialization problem is **singular** (has none or infinitely many solutions). This situation occurs often for mechanical systems, where, e.g.,  $u = \text{desiredSpeed} - \text{measuredSpeed}$  and since speed is both a state and a derivative, it is always defined by **Init.InitialState** or **Init.SteadyState** initialization.

In such a case, **Init.NoInit** has to be selected for the integrator and an additional initial equation has to be added to the system to which the integrator is connected. E.g., useful initial conditions for a 1-dim. rotational

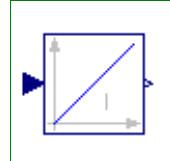
inertia controlled by a PI controller are that **angle**, **speed**, and **acceleration** of the inertia are zero.

## Package Content

Name	Description
 Integrator	Output the integral of the input signal
 LimIntegrator	Integrator with limited value of the output
 Derivative	Approximated derivative block
 FirstOrder	First order transfer function block (= 1 pole)
 SecondOrder	Second order transfer function block (= 2 poles)
 PI	Proportional-Integral controller
 PID	PID-controller in additive description form
 LimPID	P, PI, PD, and PID controller with limited output, anti-windup compensation and setpoint weighting
 TransferFunction	Linear transfer function
 StateSpace	Linear state space system
 Der	Derivative of input (= analytic differentiations)
 LowpassButterworth	Output the input signal filtered with a low pass Butterworth filter of any order
 CriticalDamping	Output the input signal filtered with an n-th order filter with critical damping

## Modelica.Blocks.Continuous.Integrator

Output the integral of the input signal



### Information

This block computes output **y** (element-wise) as *integral* of the input **u** multiplied with the gain **k**:

$$y = \frac{k}{s} u$$

It might be difficult to initialize the integrator in steady state. This is discussed in the description of package [Continuous](#).

### Parameters

Name	Default	Description
k	1	Integrator gain
<b>Initialization</b>		
initType	Modelica.Blocks.Types.Init.I...	Type of initialization (InitialState and InitialOutput are identical)
y_start	0	Initial or guess value of output (= state)
y.start	y_start	Connector of Real output signal

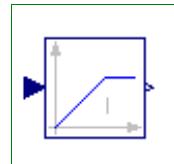
### Connectors

Name	Description

u	Connector of Real input signal
---	--------------------------------

## Modelica.Blocks.Continuous.LimIntegrator

Integrator with limited value of the output



### Information

This blocks computes **y** (element-wise) as *integral* of the input **u** multiplied with the gain **k**. If the integral reaches a given upper or lower *limit* and the input will drive the integral outside of this bound, the integration is halted and only restarted if the input drives the integral away from the bounds.

It might be difficult to initialize the integrator in steady state. This is discussed in the description of package **Continuous**.

If parameter **limitAtInit = false**, the limits of the integrator are removed from the initialization problem which leads to a much simpler equation system. After initialization has been performed, it is checked via an assert whether the output is in the defined limits. For backward compatibility reasons **limitAtInit = true**. In most cases it is best to use **limitAtInit = false**.

### Parameters

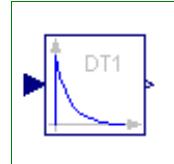
Name	Default	Description
k	1	Integrator gain
outMax	1	Upper limit of output
outMin	-outMax	Lower limit of output
<b>Initialization</b>		
initType	Modelica.Blocks.Types.Init.I...	Type of initialization
limitsAtInit	true	= false, if limits are ignored during initialization (i.e., $\text{der}(y)=k*u$ )
y_start	0	Initial or guess value of output (must be in the limits outMin .. outMax)
y.start	y_start	Connector of Real output signal

### Connectors

Name	Description
u	Connector of Real input signal

## Modelica.Blocks.Continuous.Derivative

Approximated derivative block



### Information

This blocks defines the transfer function between the input **u** and the output **y** (element-wise) as *approximated derivative*:

$$y = \frac{k * s}{T * s + 1} * u$$

If you would like to be able to change easily between different transfer functions (FirstOrder, SecondOrder, ...) by changing parameters, use the general block **TransferFunction** instead and model a derivative block with parameters

## 88 Modelica.Blocks.Continuous.Derivative

---

$b = \{k, 0\}$ ,  $a = \{T, 1\}$ .

If  $k=0$ , the block reduces to  $y=0$ .

### Parameters

Name	Default	Description
k	1	Gains
T	0.01	Time constants ( $T>0$ required; $T=0$ is ideal derivative block) [s]
Initialization		
initType	Modelica.Blocks.Types.Init.N...	Type of initialization
x_start	0	Initial or guess value of state
y_start	0	Initial value of output (= state)

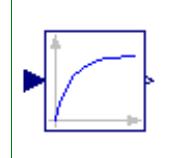
### Connectors

Name	Description
u	Connector of Real input signal
y	Connector of Real output signal

---

## Modelica.Blocks.Continuous.FirstOrder

First order transfer function block (= 1 pole)



### Information

This block defines the transfer function between the input  $u$  and the output  $y$  (element-wise) as *first order* system:

$$y = \frac{k}{T * s + 1} * u$$

If you would like to be able to change easily between different transfer functions (FirstOrder, SecondOrder, ...) by changing parameters, use the general block **TransferFunction** instead and model a first order SISO system with parameters

$b = \{k\}$ ,  $a = \{T, 1\}$ .

Example:

```
parameter: k = 0.3, T = 0.4
results in:
0.3
y = ----- * u
0.4 s + 1.0
```

### Parameters

Name	Default	Description
k	1	Gain
T	1	Time Constant [s]
Initialization		
initType	Modelica.Blocks.Types.Init.N...	Type of initialization (InitialState and InitialOutput are identical)
y_start	0	Initial or guess value of output (= state)

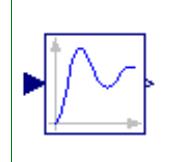
y.start  y_start	Connector of Real output signal
------------------	---------------------------------

## Connectors

Name	Description
u	Connector of Real input signal

## Modelica.Blocks.Continuous.SecondOrder

Second order transfer function block (= 2 poles)



## Information

This blocks defines the transfer function between the input u and the output y (element-wise) as *second order* system:

$$y = \frac{k}{(s / w)^2 + 2*D*(s / w) + 1} * u$$

If you would like to be able to change easily between different transfer functions (FirstOrder, SecondOrder, ...) by changing parameters, use the general model class **TransferFunction** instead and model a second order SISO system with parameters

b = {k}, a = {1/w^2, 2\*D/w, 1}.

Example:

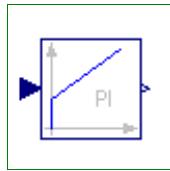
```
parameter: k = 0.3, w = 0.5, D = 0.4
results in:
          0.3
y = ----- * u
        4.0 s^2 + 1.6 s + 1
```

## Parameters

Name	Default	Description
k	1	Gain
w	1	Angular frequency
D	1	Damping
Initialization		
initType	Modelica.Blocks.Types.Init.N...	Type of initialization (InitialState and InitialOutput are identical)
y_start	0	Initial or guess value of output (= state)
yd_start	0	Initial or guess value of derivative of output (= state)
y.start	y_start	Connector of Real output signal

## Connectors

Name	Description
u	Connector of Real input signal

**Modelica.Blocks.Continuous.PI****Proportional-Integral controller****Information**

This blocks defines the transfer function between the input  $u$  and the output  $y$  (element-wise) as *PI* system:

$$\begin{aligned} y &= k * \left(1 + \frac{1}{T*s}\right) * u \\ &= k * \frac{T*s + 1}{T*s} * u \end{aligned}$$

If you would like to be able to change easily between different transfer functions (FirstOrder, SecondOrder, ...) by changing parameters, use the general model class **TransferFunction** instead and model a PI SISO system with parameters

$$b = \{k*T, k\}, a = \{T, 0\}.$$

Example:

```
parameter: k = 0.3, T = 0.4
results in:
y = 0.3 * (0.4 s + 1) / 0.4 s * u
```

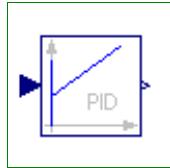
It might be difficult to initialize the PI component in steady state due to the integrator part. This is discussed in the description of package **Continuous**.

**Parameters**

Name	Default	Description
k	1	Gain
T	1	Time Constant ( $T>0$ required) [s]
Initialization		
initType	Modelica.Blocks.Types.Init.N...	Type of initialization (SteadyState and InitialOutput are identical)
x_start	0	Initial or guess value of state
y_start	0	Initial value of output

**Connectors**

Name	Description
u	Connector of Real input signal
y	Connector of Real output signal

**Modelica.Blocks.Continuous.PID****PID-controller in additive description form****Information**

This is the text-book version of a PID-controller. For a more practically useful PID-controller, use block

LimPID.

The PID block can be initialized in different ways controlled by parameter **initType**. The possible values of **initType** are defined in [Modelica.Blocks.Types.InitPID](#). This type is identical to [Types.Init](#), with the only exception that the additional option **DoNotUse\_InitialIntegratorState** is added for backward compatibility reasons (= integrator is initialized with **InitialState** whereas differential part is initialized with **NoInit** which was the initialization in version 2.2 of the Modelica standard library).

Based on the setting of **initType**, the integrator (I) and derivative (D) blocks inside the PID controller are initialized according to the following table:

<b>initType</b>	<b>I.initType</b>	<b>D.initType</b>
<b>NoInit</b>	NoInit	NoInit
<b>SteadyState</b>	SteadyState	SteadyState
<b>InitialState</b>	InitialState	InitialState
<b>InitialOutput</b> and initial equation: $y = y_{start}$	NoInit	SteadyState
<b>DoNotUse_InitialIntegratorState</b>	InitialState	NoInit

In many cases, the most useful initial condition is **SteadyState** because initial transients are then no longer present. If **initType** = **InitPID.SteadyState**, then in some cases difficulties might occur. The reason is the equation of the integrator:

$$\text{der}(y) = k * u;$$

The steady state equation "der(x)=0" leads to the condition that the input **u** to the integrator is zero. If the input **u** is already (directly or indirectly) defined by another initial condition, then the initialization problem is **singular** (has none or infinitely many solutions). This situation occurs often for mechanical systems, where, e.g., **u** = **desiredSpeed - measuredSpeed** and since speed is both a state and a derivative, it is natural to initialize it with zero. As sketched this is, however, not possible. The solution is to not initialize **u** or the variable that is used to compute **u** by an algebraic equation.

## Parameters

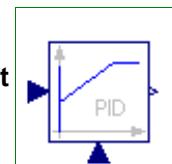
Name	Default	Description
k	1	Gain
Ti	0.5	Time Constant of Integrator [s]
Td	0.1	Time Constant of Derivative block [s]
Nd	10	The higher Nd, the more ideal the derivative block
<b>Initialization</b>		
initType	Modelica.Blocks.Types.InitPI...	Type of initialization
xi_start	0	Initial or guess value value for integrator output (= integrator state)
xd_start	0	Initial or guess value for state of derivative block
y_start	0	Initial value of output

## Connectors

Name	Description
u	Connector of Real input signal
y	Connector of Real output signal

## Modelica.Blocks.Continuous.LimPID

P, PI, PD, and PID controller with limited output, anti-windup compensation and setpoint weighting



## Information

Via parameter **controllerType** either **P**, **PI**, **PD**, or **PID** can be selected. If, e.g., PI is selected, all components belonging to the D-part are removed from the block (via conditional declarations). The example model [Modelica.Blocks.Examples.PID\\_Controller](#) demonstrates the usage of this controller. Several practical aspects of PID controller design are incorporated according to chapter 3 of the book:

Astroem K.J., and Haeggglund T.:

**PID Controllers: Theory, Design, and Tuning.** Instrument Society of America, 2nd edition, 1995.  
Information from: <http://www.control.lth.se/publications/books/asthagg95.html>

Besides the additive **proportional**, **integral** and **derivative** part of this controller, the following features are present:

- The output of this controller is limited. If the controller is in its limits, anti-windup compensation is activated to drive the integrator state to zero.
- The high-frequency gain of the derivative part is limited to avoid excessive amplification of measurement noise.
- Setpoint weighting is present, which allows to weight the setpoint in the proportional and the derivative part independantly from the measurement. The controller will respond to load disturbances and measurement noise independantly of this setting (parameters wp, wd). However, setpoint changes will depend on this setting. For example, it is useful to set the setpoint weight wd for the derivative part to zero, if steps may occur in the setpoint signal.

The parameters of the controller can be manually adjusted by performing simulations of the closed loop system (= controller + plant connected together) and using the following strategy:

1. Set very large limits, e.g., yMax = Modelica.Constants.inf
2. Select a **P**-controller and manually enlarge parameter **k** (the total gain of the controller) until the closed-loop response cannot be improved any more.
3. Select a **PI**-controller and manually adjust parameters **k** and **Ti** (the time constant of the integrator). The first value of **Ti** can be selected, such that it is in the order of the time constant of the oscillations occurring with the P-controller. If, e.g., vibrations in the order of T=10 ms occur in the previous step, start with **Ti**=0.01 s.
4. If you want to make the reaction of the control loop faster (but probably less robust against disturbances and measurement noise) select a **PID**-Controller and manually adjust parameters **k**, **Ti**, **Td** (time constant of derivative block).
5. Set the limits yMax and yMin according to your specification.
6. Perform simulations such that the output of the PID controller goes in its limits. Tune **Ni** (**Ni**\***Ti** is the time constant of the anti-windup compensation) such that the input to the limiter block (= limiter.u) goes quickly enough back to its limits. If **Ni** is decreased, this happens faster. If **Ni**=infinity, the anti-windup compensation is switched off and the controller works bad.

## Initialization

This block can be initialized in different ways controlled by parameter **initType**. The possible values of **initType** are defined in [Modelica.Blocks.Types.InitPID](#). This type is identical to [Types.Init](#), with the only exception that the additional option **DoNotUse\_InitialIntegratorState** is added for backward compatibility reasons (= integrator is initialized with **InitialState** whereas differential part is initialized with **NoInit** which was the initialization in version 2.2 of the Modelica standard library).

Based on the setting of **initType**, the integrator (I) and derivative (D) blocks inside the PID controller are initialized according to the following table:

<b>initType</b>	<b>I.initType</b>	<b>D.initType</b>
<b>NoInit</b>	NoInit	NoInit
<b>SteadyState</b>	SteadyState	SteadyState
<b>InitialState</b>	InitialState	InitialState
<b>InitialOutput</b> and initial equation: $y = y_{\text{start}}$	NoInit	SteadyState

<b>DoNotUse_InitialIntegratorState</b>	InitialState	NoInit
--	--------------	--------

In many cases, the most useful initial condition is **SteadyState** because initial transients are then no longer present. If initType = InitPID.SteadyState, then in some cases difficulties might occur. The reason is the equation of the integrator:

```
der(y) = k*u;
```

The steady state equation "der(x)=0" leads to the condition that the input u to the integrator is zero. If the input u is already (directly or indirectly) defined by another initial condition, then the initialization problem is **singular** (has none or infinitely many solutions). This situation occurs often for mechanical systems, where, e.g.,  $u = \text{desiredSpeed} - \text{measuredSpeed}$  and since speed is both a state and a derivative, it is natural to initialize it with zero. As sketched this is, however, not possible. The solution is to not initialize  $u_m$  or the variable that is used to compute  $u_m$  by an algebraic equation.

If parameter **limitAtInit = false**, the limits at the output of this controller block are removed from the initialization problem which leads to a much simpler equation system. After initialization has been performed, it is checked via an assert whether the output is in the defined limits. For backward compatibility reasons **limitAtInit = true**. In most cases it is best to use **limitAtInit = false**.

## Parameters

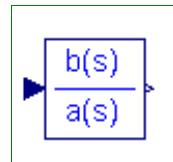
Name	Default	Description
controllerType	Modelica.Blocks.Types.Simple...	Type of controller
k	1	Gain of controller
Ti	0.5	Time constant of Integrator block [s]
Td	0.1	Time constant of Derivative block [s]
yMax	1	Upper limit of output
yMin	-yMax	Lower limit of output
wp	1	Set-point weight for Proportional block (0..1)
wd	0	Set-point weight for Derivative block (0..1)
Ni	0.9	$Ni \cdot Ti$ is time constant of anti-windup compensation
Nd	10	The higher Nd, the more ideal the derivative block
<b>Initialization</b>		
initType	Modelica.Blocks.Types.InitPi...	Type of initialization
limitsAtInit	true	= false, if limits are ignored during initialization
xi_start	0	Initial or guess value for integrator output (= integrator state)
xd_start	0	Initial or guess value for state of derivative block
y_start	0	Initial value of output

## Connectors

Name	Description
u_s	Connector of setpoint input signal
u_m	Connector of measurement input signal
y	Connector of actuator output signal

## Modelica.Blocks.Continuous.TransferFunction

### Linear transfer function



## Information

This block defines the transfer function between the input  $u$  and the output  $y$  as (nb = dimension of b, na = dimension of a):

$$y(s) = \frac{b[1]*s^{[nb-1]} + b[2]*s^{[nb-2]} + \dots + b[nb]}{a[1]*s^{[na-1]} + a[2]*s^{[na-2]} + \dots + a[na]} * u(s)$$

State variables  $x$  are defined according to **controller canonical** form. Initial values of the states can be set as start values of  $x$ .

Example:

```
TransferFunction g(b = {2, 4}, a = {1, 3});
```

results in the following transfer function:

$$y = \frac{2*s + 4}{s + 3} * u$$

## Parameters

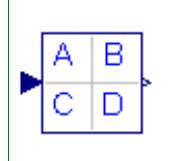
Name	Default	Description
b[:]	{1}	Numerator coefficients of transfer function.
a[:]	{1,1}	Denominator coefficients of transfer function.
Initialization		
initType	Modelica.Blocks.Types.Init.N...	Type of initialization
x_start[size(a, 1) - 1]	zeros(nx)	Initial or guess values of states
y_start	0	Initial value of output (derivatives of y are zero upto nx-1-th derivative)

## Connectors

Name	Description
u	Connector of Real input signal
y	Connector of Real output signal

## Modelica.Blocks.Continuous.StateSpace

Linear state space system



## Information

The State Space block defines the relation between the input  $u$  and the output  $y$  in state space form:

$$\begin{aligned} \text{der}(x) &= A * x + B * u \\ y &= C * x + D * u \end{aligned}$$

The input is a vector of length nu, the output is a vector of length ny and nx is the number of states.  
Accordingly

A has the dimension: A(nx, nx),

B has the dimension: B(nx,nu),  
 C has the dimension: C(ny,nx),  
 D has the dimension: D(ny,nu)

Example:

```

parameter: A = [0.12, 2;3, 1.5]
parameter: B = [2, 7;3, 1]
parameter: C = [0.1, 2]
parameter: D = zeros(ny,nu)
results in the following equations:
[der(x[1])]  [0.12  2.00] [x[1]]  [2.0  7.0] [u[1]]
[ ] = [ ] * [ ] + [ ] * [ ]
[der(x[2])]  [3.00  1.50] [x[2]]  [0.1  2.0] [u[2]]
[x[1]]          [u[1]]
y[1] = [0.1  2.0] * [ ] + [0  0] * [ ]
[x[2]]          [u[2]]

```

## Parameters

Name	Default	Description
A[:, size(A, 1)]	[1, 0; 0, 1]	Matrix A of state space model
B[size(A, 1), :]	[1; 1]	Matrix B of state space model
C[:, size(A, 1)]	[1, 1]	Matrix C of state space model
D[size(C, 1), size(B, 2)]	zeros(size(C, 1), size(B, 2))	Matrix D of state space model
<b>Initialization</b>		
initType	Modelica.Blocks.Types.Init.N...	Type of initialization
x_start[nx]	zeros(nx)	Initial or guess values of states
y_start[ny]	zeros(ny)	Initial values of outputs (remaining states are in steady state if possible)

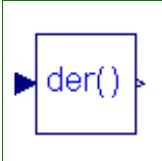
## Connectors

Name	Description
u[nin]	Connector of Real input signals
y[nout]	Connector of Real output signals

---

## Modelica.Blocks.Continuous.Der

Derivative of input (= analytic differentiations)



## Information

Defines that the output y is the *derivative* of the input u. Note, that Modelica.Blocks.Continuous.Derivative computes the derivative in an approximate sense, where as this block computes the derivative exactly. This requires that the input u is differentiated by the Modelica translator, if this derivative is not yet present in the model.

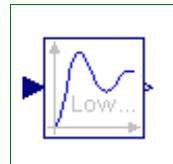
## Connectors

Name	Description
u	Connector of Real input signal

y	Connector of Real output signal
---	---------------------------------

## Modelica.Blocks.Continuous.LowpassButterworth

Output the input signal filtered with a low pass Butterworth filter of any order



### Information

This block defines the transfer function between the input  $u$  and the output  $y$  as an  $n$ -th order low pass filter with *Butterworth* characteristics and cut-off frequency  $f$ . It is implemented as a series of second order filters and a first order filter.

If transients at the simulation start shall be avoided the states  $x_1$  and  $x_r$  need to be initialized with the start value of the input signal and the states  $x_2$  need to be initialized with zeros.

$$y = PT_1 * PT_2 * \dots * PT_{n/2} * PT_1 u$$

### Parameters

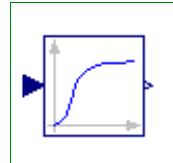
Name	Default	Description
n	2	Order of filter
f	1	Cut-off frequency [Hz]
Initialization		
initType	Modelica.Blocks.Types.Init.N...	Type of initialization
x1_start[m]	zeros(m)	Initial or guess values of states 1 ( $\text{der}(x_1)=x_2$ )
x2_start[m]	zeros(m)	Initial or guess values of states 2
xr_start	0.0	Initial or guess value of real pole for uneven order otherwise dummy
y_start	0.0	Initial value of output (states are initialized in steady state if possible)

### Connectors

Name	Description
u	Connector of Real input signal
y	Connector of Real output signal

## Modelica.Blocks.Continuous.CriticalDamping

Output the input signal filtered with an  $n$ -th order filter with critical damping



### Information

This block defines the transfer function between the input  $u$  and the output  $y$  as an  $n$ -th order filter with *critical damping* characteristics and cut-off frequency  $f=1/T$ . It is implemented as a series of first order filters.

If transients at the simulation start shall be avoided the states  $x$  need to be initialized with the start value of the input.

$$y = \frac{k}{(T * s + 1)^n} * u$$

## Parameters

Name	Default	Description
n	2	Order of filter
f	1	Cut-off frequency [Hz]
<b>Initialization</b>		
initType	Modelica.Blocks.Types.Init.N...	Type of initialization
x_start[n]	zeros(n)	Initial or guess values of states
y_start	0.0	Initial value of output (remaining states are in steady state)

## Connectors

Name	Description
u	Connector of Real input signal
y	Connector of Real output signal

---

## Modelica.Blocks.Discrete

Library of discrete input/output blocks with fixed sample period

## Information

This package contains **discrete control blocks** with **fixed sample period**. Every component of this package is structured in the following way:

1. A component has **continuous real** input and output signals.
2. The **input** signals are **sampled** by the given sample period defined via parameter **samplePeriod**. The first sample instant is defined by parameter **startTime**.
3. The **output** signals are computed from the sampled input signals.

A **sampled data system** may consist of components of package **Discrete** and of every other purely **algebraic** input/output block, such as the components of packages **Modelica.Blocks.Math**, **Modelica.Blocks.Nonlinear** or **Modelica.Blocks.Sources**.

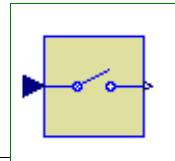
## Package Content

Name	Description
 Sampler	Ideal sampling of continuous signals
 ZeroOrderHold	Zero order hold of a sampled-data system
 FirstOrderHold	First order hold of a sampled-data system
 UnitDelay	Unit Delay Block
 TransferFunction	Discrete Transfer Function block
 StateSpace	Discrete State Space block
 TriggeredSampler	Triggered sampling of continuous signals
 TriggeredMax	Compute maximum, absolute value of continuous signal at trigger instants

---

## Modelica.Blocks.Discrete.Sampler

Ideal sampling of continuous signals



## Information

Samples the continues input signal with a sampling rate defined via parameter **samplePeriod**.

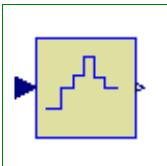
## Parameters

Name	Default	Description
samplePeriod	0.1	Sample period of component [s]
startTime	0	First sample time instant [s]

## Connectors

Name	Description
u	Continuous input signal
y	Continuous output signal

## Modelica.Blocks.Discrete.ZeroOrderHold



Zero order hold of a sampled-data system

## Information

The output is identical to the sampled input signal at sample time instants and holds the output at the value of the last sample instant during the sample points.

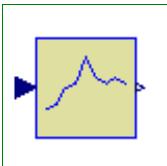
## Parameters

Name	Default	Description
samplePeriod	0.1	Sample period of component [s]
startTime	0	First sample time instant [s]

## Connectors

Name	Description
u	Continuous input signal
y	Continuous output signal

## Modelica.Blocks.Discrete.FirstOrderHold



First order hold of a sampled-data system

## Information

The output signal is the extrapolation through the values of the last two sampled input signals.

## Parameters

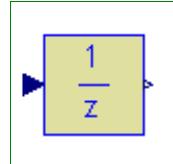
Name	Default	Description
samplePeriod	0.1	Sample period of component [s]
startTime	0	First sample time instant [s]

## Connectors

Name	Description
u	Continuous input signal
y	Continuous output signal

## Modelica.Blocks.Discrete.UnitDelay

### Unit Delay Block



### Information

This block describes a unit delay:

$$y = \frac{1}{z} * u$$

that is, the output signal y is the input signal u of the previous sample instant. Before the second sample instant, the output y is identical to parameter yStart.

## Parameters

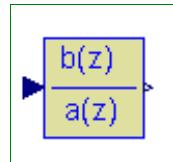
Name	Default	Description
y_start	0	Initial value of output signal
samplePeriod	0.1	Sample period of component [s]
startTime	0	First sample time instant [s]

## Connectors

Name	Description
u	Continuous input signal
y	Continuous output signal

## Modelica.Blocks.Discrete.TransferFunction

### Discrete Transfer Function block



### Information

The **discrete transfer function** block defines the transfer function between the input signal u and the output signal y. The numerator has the order nb-1, the denominator has the order na-1.

$$y(z) = \frac{b(1)*z^{(nb-1)} + b(2)*z^{(nb-2)} + \dots + b(nb)}{a(1)*z^{(na-1)} + a(2)*z^{(na-2)} + \dots + a(na)} * u(z)$$

State variables **x** are defined according to **controller canonical** form. Initial values of the states can be set as start values of **x**.

Example:

```
Blocks.Discrete.TransferFunction g(b = {2, 4}, a = {1, 3});
```

results in the following transfer function:

$$y = \frac{2z + 4}{z + 3} * u$$

## Parameters

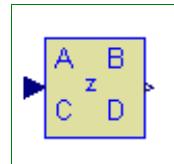
Name	Default	Description
b[:]	{1}	Numerator coefficients of transfer function.
a[:]	{1,1}	Denominator coefficients of transfer function.
samplePeriod	0.1	Sample period of component [s]
startTime	0	First sample time instant [s]

## Connectors

Name	Description
u	Continuous input signal
y	Continuous output signal

## Modelica.Blocks.Discrete.StateSpace

### Discrete State Space block



### Information

The **discrete state space** block defines the relation between the input  $u=\text{inPort.signal}$  and the output  $y=\text{outPort.signal}$  in state space form:

$$\begin{aligned} x &= A * \text{pre}(x) + B * u \\ y &= C * \text{pre}(x) + D * u \end{aligned}$$

where  $\text{pre}(x)$  is the value of the discrete state  $x$  at the previous sample time instant. The input is a vector of length  $n_u$ , the output is a vector of length  $n_y$  and  $n_x$  is the number of states. Accordingly

A has the dimension:  $A(n_x, n_x)$ ,  
B has the dimension:  $B(n_x, n_u)$ ,  
C has the dimension:  $C(n_y, n_x)$ ,  
D has the dimension:  $D(n_y, n_u)$

### Example:

```
parameter: A = [0.12, 2;3, 1.5]
parameter: B = [2, 7;3, 1]
parameter: C = [0.1, 2]
parameter: D = zeros(ny,nu)

results in the following equations:
[x[1]]  [0.12  2.00] [pre(x[1])]  [2.0  7.0]  [u[1]]
[     ] = [           ]*[           ] + [           ]*[           ]
[x[2]]  [3.00  1.50] [pre(x[2])]  [0.1  2.0]  [u[2]]
                           [pre(x[1])]  [u[1]]
y[1]    = [0.1  2.0] * [           ] + [0  0] * [           ]
                           [pre(x[2])]  [u[2]]
```

## Parameters

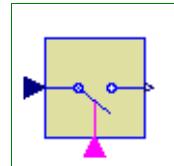
Name	Default	Description
A[:, size(A, 1)]	[1, 0; 0, 1]	Matrix A of state space model
B[size(A, 1), :]	[1; 1]	Matrix B of state space model
C[:, size(A, 1)]	[1, 1]	Matrix C of state space model
D[size(C, 1), size(B, 2)]	zeros(size(C, 1), size(B, 2))	Matrix D of state space model
samplePeriod	0.1	Sample period of component [s]
startTime	0	First sample time instant [s]

## Connectors

Name	Description
u[nin]	Continuous input signals
y[nout]	Continuous output signals

## Modelica.Blocks.Discrete.TriggeredSampler

Triggered sampling of continuous signals



## Information

Samples the continuous input signal whenever the trigger input signal is rising (i.e., trigger changes from **false** to **true**) and provides the sampled input signal as output. Before the first sampling, the output signal is equal to the initial value defined via parameter **y0**.

## Parameters

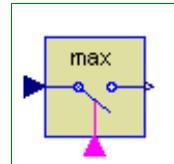
Name	Default	Description
y_start	extends Real	initial value of output signal

## Connectors

Name	Description
u	Connector with an input signal of type SignalType
y	Connector with an output signal of type SignalType
trigger	

## Modelica.Blocks.Discrete.TriggeredMax

Compute maximum, absolute value of continuous signal at trigger instants



## Information

Samples the continuous input signal whenever the trigger input signal is rising (i.e., trigger changes from **false** to **true**). The maximum, absolute value of the input signal at the sampling point is provided as output signal.

## Connectors

Name	Description

u	Connector with an input signal of type SignalType
y	Connector with an output signal of type SignalType
trigger	

## Modelica.Blocks.Interfaces

Library of connectors and partial models for input/output blocks

### Information

This package contains interface definitions for **continuous** input/output blocks with Real, Integer and Boolean signals. Furthermore, it contains partial models for continuous and discrete blocks.

### Package Content

Name	Description
 RealSignal	Real port (both input/output possible)
 BooleanSignal	Boolean port (both input/output possible)
 IntegerSignal	Integer port (both input/output possible)
 RealInput	'input Real' as connector
 RealOutput	'output Real' as connector
 BooleanInput	'input Boolean' as connector
 BooleanOutput	'output Boolean' as connector
 IntegerInput	'input Integer' as connector
 IntegerOutput	'output Integer' as connector
 BlockIcon	Basic graphical layout of input/output block
 SO	Single Output continuous control block
 MO	Multiple Output continuous control block
 SISO	Single Input Single Output continuous control block
 SI2SO	2 Single Input / 1 Single Output continuous control block
 SIMO	Single Input Multiple Output continuous control block
 MISO	Multiple Input Single Output continuous control block
 MIMO	Multiple Input Multiple Output continuous control block
 MIMOs	Multiple Input Multiple Output continuous control block with same number of inputs and outputs
 MI2MO	2 Multiple Input / Multiple Output continuous control block
 SignalSource	Base class for continuous signal source
 SVcontrol	Single-Variable continuous controller
 MVcontrol	Multi-Variable continuous controller
 DiscreteBlockIcon	Graphical layout of discrete block component icon
 DiscreteBlock	Base class of discrete control blocks

 DiscreteSISO	Single Input Single Output discrete control block
 DiscreteMIMO	Multiple Input Multiple Output discrete control block
 DiscreteMIMOs	Multiple Input Multiple Output discrete control block
 SVdiscrete	Discrete Single-Variable controller
 MVdiscrete	Discrete Multi-Variable controller
 BooleanBlockIcon	Basic graphical layout of Boolean block
 BooleanSISO	Single Input Single Output control block with signals of type Boolean
 BooleanMIMOs	Multiple Input Multiple Output continuous control block with same number of inputs and outputs of boolean type
 MI2BooleanMOs	2 Multiple Input / Boolean Multiple Output block with same signal lengths
 SI2BooleanSO	2 Single Input / Boolean Single Output block
 BooleanSignalSource	Base class for Boolean signal sources
 IntegerBlockIcon	Basic graphical layout of Integer block
 IntegerSO	Single Integer Output continuous control block
 IntegerMO	Multiple Integer Output continuous control block
 IntegerSignalSource	Base class for continuous Integer signal source
 IntegerSIBooleanSO	Integer Input Boolean Output continuous control block
 IntegerMIBooleanMOs	Multiple Integer Input Multiple Boolean Output continuous control block with same number of inputs and outputs
 partialBooleanBlockIcon	Basic graphical layout of logical block
 partialBooleanSISO	Partial block with 1 input and 1 output Boolean signal
 partialBooleanSI2SO	Partial block with 2 input and 1 output Boolean signal
 partialBooleanSI3SO	Partial block with 3 input and 1 output Boolean signal
 partialBooleanSI	Partial block with 1 input Boolean signal
 partialBooleanSO	Partial block with 1 output Boolean signal
 partialBooleanSource	Partial source block (has 1 output Boolean signal and an appropriate default icon)
 partialBooleanThresholdComparison	Partial block to compare the Real input u with a threshold and provide the result as 1 Boolean output signal
 partialBooleanComparison	Partial block with 2 Real input and 1 Boolean output signal (the result of a comparison of the two Real inputs)
 Adaptors	Obsolete package with components to send signals to a bus or receive signals from a bus (only for backward compatibility)
 PartialConversionBlock	Partial block defining the interface for conversion blocks

## Modelica.Blocks.Interfaces.RealSignal

Real port (both input/output possible)

## Information

Connector with one signal of type Real (no icon, no input/output prefix).

---

## Modelica.Blocks.Interfaces.BooleanSignal

**Boolean port (both input/output possible)**

## Information

Connector with one signal of type Boolean (no icon, no input/output prefix).

---

## Modelica.Blocks.Interfaces.IntegerSignal

**Integer port (both input/output possible)**

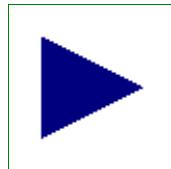
## Information

Connector with one signal of type Icon (no icon, no input/output prefix).

---

## Modelica.Blocks.Interfaces.RealInput

**'input Real' as connector**



## Information

Connector with one input signal of type Real.

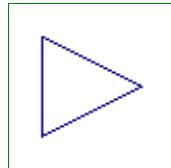
## Parameters

Name	Default	Description
replaceable type SignalType	Real	

---

## Modelica.Blocks.Interfaces.RealOutput

**'output Real' as connector**



## Information

Connector with one output signal of type Real.

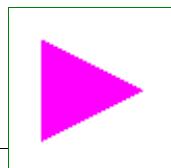
## Parameters

Name	Default	Description
replaceable type SignalType	Real	

---

## Modelica.Blocks.Interfaces.BooleanInput

**'input Boolean' as connector**

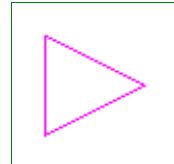


**Information**

Connector with one input signal of type Boolean.

**Modelica.Blocks.Interfaces.BooleanOutput**

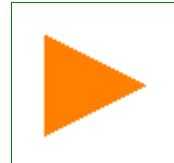
'output Boolean' as connector

**Information**

Connector with one output signal of type Boolean.

**Modelica.Blocks.Interfaces.IntegerInput**

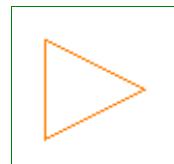
'input Integer' as connector

**Information**

Connector with one input signal of type Integer.

**Modelica.Blocks.Interfaces.IntegerOutput**

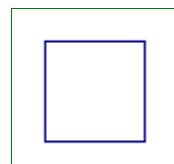
'output Integer' as connector

**Information**

Connector with one output signal of type Integer.

**Modelica.Blocks.Interfaces.BlockIcon**

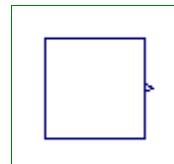
**Basic graphical layout of input/output block**

**Information**

Block that has only the basic icon for an input/output block (no declarations, no equations). Most blocks of package Modelica.Blocks inherit directly or indirectly from this block.

**Modelica.Blocks.Interfaces.SO**

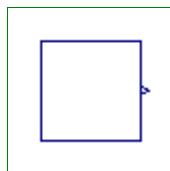
**Single Output continuous control block**

**Information**

Block has one continuous Real output signal.

**Connectors**

Name	Description
y	Connector of Real output signal

**Modelica.Blocks.Interfaces.MO****Multiple Output continuous control block****Information**

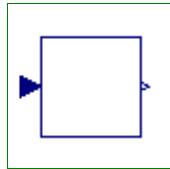
Block has one continuous Real output signal vector.

**Parameters**

Name	Default	Description
nout	1	Number of outputs

**Connectors**

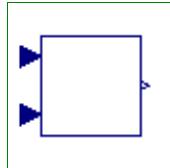
Name	Description
y[nout]	Connector of Real output signals

**Modelica.Blocks.Interfaces.SISO****Single Input Single Output continuous control block****Information**

Block has one continuous Real input and one continuous Real output signal.

**Connectors**

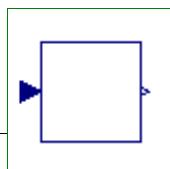
Name	Description
u	Connector of Real input signal
y	Connector of Real output signal

**Modelica.Blocks.Interfaces.SI2SO****2 Single Input / 1 Single Output continuous control block****Information**

Block has two continuous Real input signals u1 and u2 and one continuous Real output signal y.

**Connectors**

Name	Description
u1	Connector of Real input signal 1
u2	Connector of Real input signal 2
y	Connector of Real output signal

**Modelica.Blocks.Interfaces.SIMO****Single Input Multiple Output continuous control block**

## Information

Block has one continuous Real input signal and a vector of continuous Real output signals.

## Parameters

Name	Default	Description
nout	1	Number of outputs

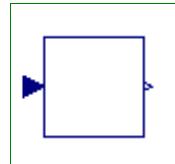
## Connectors

Name	Description
u	Connector of Real input signal
y[nout]	Connector of Real output signals

---

## Modelica.Blocks.Interfaces.MISO

Multiple Input Single Output continuous control block



## Information

Block has a vector of continuous Real input signals and one continuous Real output signal.

## Parameters

Name	Default	Description
nin	1	Number of inputs

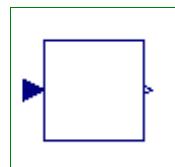
## Connectors

Name	Description
u[nin]	Connector of Real input signals
y	Connector of Real output signal

---

## Modelica.Blocks.Interfaces.MIMO

Multiple Input Multiple Output continuous control block



## Information

Block has a continuous Real input and a continuous Real output signal vector. The signal sizes of the input and output vector may be different.

## Parameters

Name	Default	Description
nin	1	Number of inputs
nout	1	Number of outputs

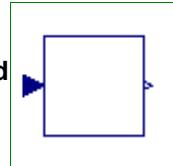
## Connectors

Name	Description

u[nin]	Connector of Real input signals
y[nout]	Connector of Real output signals

## Modelica.Blocks.Interfaces.MIMOs

Multiple Input Multiple Output continuous control block with same number of inputs and outputs



### Information

Block has a continuous Real input and a continuous Real output signal vector where the signal sizes of the input and output vector are identical.

### Parameters

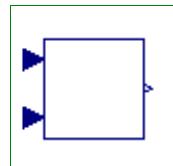
Name	Default	Description
n	1	Number of inputs (= number of outputs)

### Connectors

Name	Description
u[n]	Connector of Real input signals
y[n]	Connector of Real output signals

## Modelica.Blocks.Interfaces.MI2MO

2 Multiple Input / Multiple Output continuous control block



### Information

Block has two continuous Real input vectors  $u_1$  and  $u_2$  and one continuous Real output vector  $y$ . All vectors have the same number of elements.

### Parameters

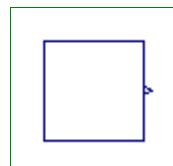
Name	Default	Description
n	1	Dimension of input and output vectors.

### Connectors

Name	Description
u1[n]	Connector 1 of Real input signals
u2[n]	Connector 2 of Real input signals
y[n]	Connector of Real output signals

## Modelica.Blocks.Interfaces.SignalSource

Base class for continuous signal source



## Information

Basic block for Real sources of package Blocks.Sources. This component has one continuous Real output signal  $y$  and two parameters (offset, startTime) to shift the generated signal.

## Parameters

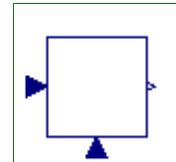
Name	Default	Description
offset	0	Offset of output signal $y$
startTime	0	Output $y = \text{offset}$ for time < startTime [s]

## Connectors

Name	Description
y	Connector of Real output signal

## Modelica.Blocks.Interfaces.SVcontrol

Single-Variable continuous controller



## Information

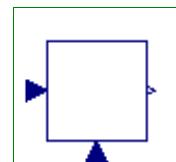
Block has two continuous Real input signals and one continuous Real output signal. The block is designed to be used as base class for a corresponding controller.

## Connectors

Name	Description
u_s	Connector of setpoint input signal
u_m	Connector of measurement input signal
y	Connector of actuator output signal

## Modelica.Blocks.Interfaces.MVcontrol

Multi-Variable continuous controller



## Information

Block has two continuous Real input signal vectors and one continuous Real output signal vector. The block is designed to be used as base class for a corresponding controller.

## Parameters

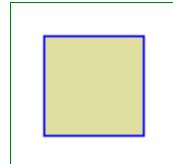
Name	Default	Description
nu_s	1	Number of setpoint inputs
nu_m	1	Number of measurement inputs
ny	1	Number of actuator outputs

## Connectors

Name	Description
u_s[nu_s]	Connector of setpoint input signals

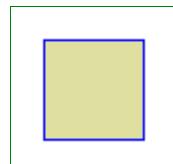
u_m[nu_m]	Connector of measurement input signals
y[ny]	Connector of actuator output signals

---

**Modelica.Blocks.Interfaces.DiscreteBlockIcon****Graphical layout of discrete block component icon****Information**

Block that has only the basic icon for an input/output, discrete block (no declarations, no equations), e.g., from Blocks.Discrete.

---

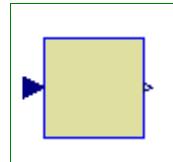
**Modelica.Blocks.Interfaces.DiscreteBlock****Base class of discrete control blocks****Information**

Basic definitions of a discrete block of library Blocks.Discrete.

**Parameters**

Name	Default	Description
samplePeriod	0.1	Sample period of component [s]
startTime	0	First sample time instant [s]

---

**Modelica.Blocks.Interfaces.DiscreteSISO****Single Input Single Output discrete control block****Information**

Block has one continuous input and one continuous output signal which are sampled due to the defined **samplePeriod** parameter.

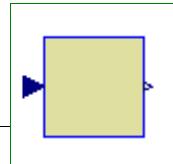
**Parameters**

Name	Default	Description
samplePeriod	0.1	Sample period of component [s]
startTime	0	First sample time instant [s]

**Connectors**

Name	Description
u	Continuous input signal
y	Continuous output signal

---

**Modelica.Blocks.Interfaces.DiscreteMIMO****Multiple Input Multiple Output discrete control block**

## Information

Block has a continuous input and a continuous output signal vector which are sampled due to the defined **samplePeriod** parameter.

## Parameters

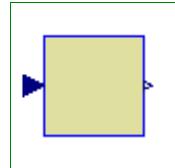
Name	Default	Description
samplePeriod	0.1	Sample period of component [s]
startTime	0	First sample time instant [s]
nin	1	Number of inputs
nout	1	Number of outputs

## Connectors

Name	Description
u[nin]	Continuous input signals
y[nout]	Continuous output signals

## Modelica.Blocks.Interfaces.DiscreteMIMOs

Multiple Input Multiple Output discrete control block



## Information

Block has a continuous input and a continuous output signal vector where the signal sizes of the input and output vector are identical. These signals are sampled due to the defined **samplePeriod** parameter.

## Parameters

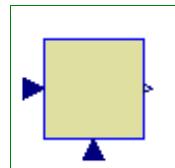
Name	Default	Description
n	1	Number of inputs (= number of outputs)
samplePeriod	0.1	Sample period of component [s]
startTime	0	First sample time instant [s]

## Connectors

Name	Description
u[n]	Continuous input signals
y[n]	Continuous output signals

## Modelica.Blocks.Interfaces.SVdiscrete

Discrete Single-Variable controller



## Information

Block has two continuous Real input signals and one continuous Real output signal that are sampled due to the defined **samplePeriod** parameter. The block is designed to be used as base class for a corresponding controller.

## Parameters

Name	Default	Description
samplePeriod	0.1	Sample period of component [s]
startTime	0	First sample time instant [s]

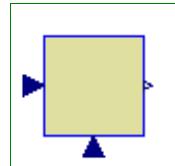
## Connectors

Name	Description
u_s	Continuous scalar setpoint input signal
u_m	Continuous scalar measurement input signal
y	Continuous scalar actuator output signal

---

## Modelica.Blocks.Interfaces.MVdiscrete

Discrete Multi-Variable controller



## Information

Block has two continuous Real input signal vectors and one continuous Real output signal vector. The vector signals are sampled due to the defined **samplePeriod** parameter. The block is designed to be used as base class for a corresponding controller.

## Parameters

Name	Default	Description
samplePeriod	0.1	Sample period of component [s]
startTime	0	First sample time instant [s]
nu_s	1	Number of setpoint inputs
nu_m	1	Number of measurement inputs
ny	1	Number of actuator outputs

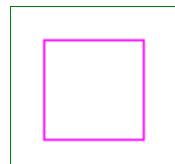
## Connectors

Name	Description
u_s[nu_s]	Continuous setpoint input signals
u_m[nu_m]	Continuous measurement input signals
y[ny]	Continuous actuator output signals

---

## Modelica.Blocks.Interfaces.BooleanBlockIcon

Basic graphical layout of Boolean block



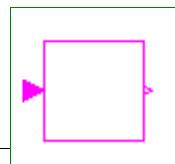
## Information

Block that has only the basic icon for an input/output, Boolean block (no declarations, no equations).

---

## Modelica.Blocks.Interfaces.BooleanSISO

Single Input Single Output control block with signals of type Boolean



## Information

Block has one continuous Boolean input and one continuous Boolean output signal.

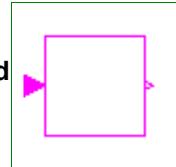
## Connectors

Name	Description
u	Connector of Boolean input signal
y	Connector of Boolean output signal

---

## Modelica.Blocks.Interfaces.BooleanMIMOs

**Multiple Input Multiple Output continuous control block with same number of inputs and outputs of boolean type**



## Information

Block has a continuous Boolean input and a continuous Boolean output signal vector where the signal sizes of the input and output vector are identical.

## Parameters

Name	Default	Description
n	1	Number of inputs (= number of outputs)

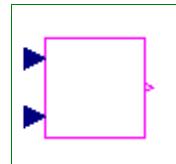
## Connectors

Name	Description
u[n]	Connector of Boolean input signals
y[n]	Connector of Boolean output signals

---

## Modelica.Blocks.Interfaces.MI2BooleanMOs

**2 Multiple Input / Boolean Multiple Output block with same signal lengths**



## Information

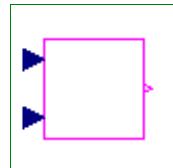
Block has two Boolean input vectors u1 and u2 and one Boolean output vector y. All vectors have the same number of elements.

## Parameters

Name	Default	Description
n	1	Dimension of input and output vectors.

## Connectors

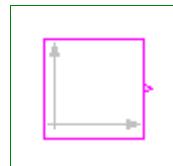
Name	Description
u1[n]	Connector 1 of Boolean input signals
u2[n]	Connector 2 of Boolean input signals
y[n]	Connector of Boolean output signals

**Modelica.Blocks.Interfaces.SI2BooleanSO****2 Single Input / Boolean Single Output block****Information**

Block has two Boolean input signals u1 and u2 and one Boolean output signal y.

**Connectors**

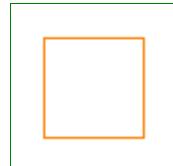
Name	Description
u1	Connector 1 of Boolean input signals
u2	Connector 2 of Boolean input signals
y	Connector of Boolean output signals

**Modelica.Blocks.Interfaces.BooleanSignalSource****Base class for Boolean signal sources****Information**

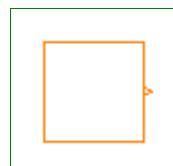
Basic block for Boolean sources of package Blocks.Sources. This component has one continuous Boolean output signal y.

**Connectors**

Name	Description
y	Connector of Boolean output signal

**Modelica.Blocks.Interfaces.IntegerBlockIcon****Basic graphical layout of Integer block****Information**

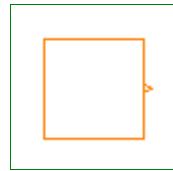
Block that has only the basic icon for an input/output, Integer block (no declarations, no equations).

**Modelica.Blocks.Interfaces.IntegerSO****Single Integer Output continuous control block****Information**

Block has one continuous Integer output signal.

**Connectors**

Name	Description
y	Connector of Integer output signal

**Modelica.Blocks.Interfaces.IntegerMO****Multiple Integer Output continuous control block****Information**

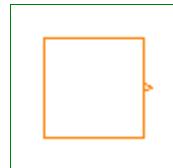
Block has one continuous Integer output signal vector.

**Parameters**

Name	Default	Description
nout	1	Number of outputs

**Connectors**

Name	Description
y[nout]	Connector of Integer output signals

**Modelica.Blocks.Interfaces.IntegerSignalSource****Base class for continuous Integer signal source****Information**

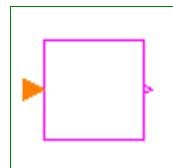
Basic block for Integer sources of package Blocks.Sources. This component has one continuous Integer output signal y and two parameters (offset, startTime) to shift the generated signal.

**Parameters**

Name	Default	Description
offset	0	Offset of output signal y
startTime	0	Output y = offset for time < startTime [s]

**Connectors**

Name	Description
y	Connector of Integer output signal

**Modelica.Blocks.Interfaces.IntegerSIBooleanSO****Integer Input Boolean Output continuous control block****Information**

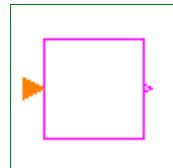
Block has a continuous Integer input and a continuous Boolean output signal.

**Connectors**

Name	Description
u	Connector of Integer input signal
y	Connector of Boolean output signal

**Modelica.Blocks.Interfaces.IntegerMIBooleanMOs**

**Multiple Integer Input Multiple Boolean Output continuous control block with same number of inputs and outputs**

**Information**

Block has a continuous Integer input and a continuous Boolean output signal vector where the signal sizes of the input and output vector are identical.

**Parameters**

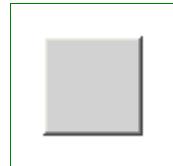
Name	Default	Description
n	1	Number of inputs (= number of outputs)

**Connectors**

Name	Description
u[n]	Connector of Integer input signals
y[n]	Connector of Boolean output signals

**Modelica.Blocks.Interfaces.partialBooleanBlockIcon**

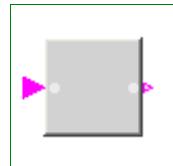
**Basic graphical layout of logical block**

**Information**

Block that has only the basic icon for an input/output, Boolean block (no declarations, no equations) used especially in the Blocks.Logical library.

**Modelica.Blocks.Interfaces.partialBooleanSISO**

**Partial block with 1 input and 1 output Boolean signal**

**Information**

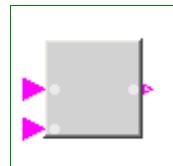
Block has one continuous Boolean input and one continuous Boolean output signal with a 3D icon (e.g. used in Blocks.Logical library).

**Connectors**

Name	Description
u	Connector of Boolean input signal
y	Connector of Boolean output signal

**Modelica.Blocks.Interfaces.partialBooleanSI2SO**

**Partial block with 2 input and 1 output Boolean signal**

**Information**

Block has two continuous Boolean input and one continuous Boolean output signal with a 3D icon (e.g. used

in Blocks.Logical library).

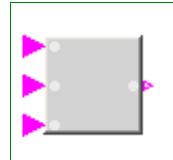
### Connectors

Name	Description
u1	Connector of first Boolean input signal
u2	Connector of second Boolean input signal
y	Connector of Boolean output signal

---

## Modelica.Blocks.Interfaces.partialBooleanSI3SO

Partial block with 3 input and 1 output Boolean signal



### Information

Block has three continuous Boolean input and one continuous Boolean output signal with a 3D icon (e.g. used in Blocks.Logical library).

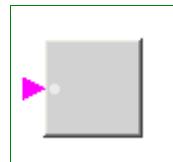
### Connectors

Name	Description
u1	Connector of first Boolean input signal
u2	Connector of second Boolean input signal
u3	Connector of third Boolean input signal
y	Connector of Boolean output signal

---

## Modelica.Blocks.Interfaces.partialBooleanSI

Partial block with 1 input Boolean signal



### Information

Block has one continuous Boolean input signal with a 3D icon (e.g. used in Blocks.Logical library).

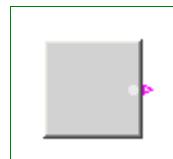
### Connectors

Name	Description
u	Connector of Boolean input signal

---

## Modelica.Blocks.Interfaces.partialBooleanSO

Partial block with 1 output Boolean signal



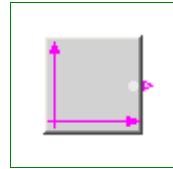
### Information

Block has one continuous Boolean output signal with a 3D icon (e.g. used in Blocks.Logical library).

### Connectors

Name	Description

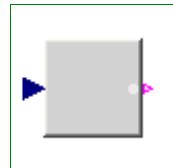
y	Connector of Boolean output signal
---	------------------------------------

**Modelica.Blocks.Interfaces.partialBooleanSource****Partial source block (has 1 output Boolean signal and an appropriate default icon)****Information**

Basic block for Boolean sources of package Blocks.Sources. This component has one continuous Boolean output signal y and a 3D icon (e.g. used in Blocks.Logical library).

**Connectors**

Name	Description
y	Connector of Boolean output signal

**Modelica.Blocks.Interfaces.partialBooleanThresholdComparison****Partial block to compare the Real input u with a threshold and provide the result as 1 Boolean output signal****Information**

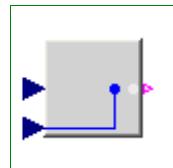
Block has one continuous Real input and one continuous Boolean output signal as well as a 3D icon (e.g. used in Blocks.Logical library).

**Parameters**

Name	Default	Description
threshold	0	Comparison with respect to threshold

**Connectors**

Name	Description
u	Connector of Boolean input signal
y	Connector of Boolean output signal

**Modelica.Blocks.Interfaces.partialBooleanComparison****Partial block with 2 Real input and 1 Boolean output signal (the result of a comparison of the two Real inputs)****Information**

Block has two continuous Real input and one continuous Boolean output signal as a result of the comparison of the two input signals. The block has a 3D icon (e.g. used in Blocks.Logical library).

**Connectors**

Name	Description
u1	Connector of first Boolean input signal

u2	Connector of second Boolean input signal
y	Connector of Boolean output signal

## Modelica.Blocks.Interfaces.Adaptors

Obsolete package with components to send signals to a bus or receive signals from a bus (only for backward compatibility)

### Information

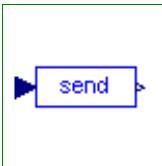
The components of this package should no longer be used. They are only provided for backward compatibility. It is much more convenient and more powerful to use "expandable connectors" for signal buses, see example [BusUsage](#).

### Package Content

Name	Description
 SendReal	Obsolete block to send Real signal to bus
 SendBoolean	Obsolete block to send Boolean signal to bus
 SendInteger	Obsolete block to send Integer signal to bus
 ReceiveReal	Obsolete block to receive Real signal from bus
 ReceiveBoolean	Obsolete block to receive Boolean signal from bus
 ReceiveInteger	Obsolete block to receive Integer signal from bus
 AdaptorReal ...	Completely obsolete adaptor between 'old' and 'new' Real signal connectors (only for backward compatibility)
 AdaptorBoolean ...	Completely obsolete adaptor between 'old' and 'new' Boolean signal connectors (only for backward compatibility)
 AdaptorInteger ...	Completely obsolete adaptor between 'old' and 'new' Integer signal connectors (only for backward compatibility)

## Modelica.Blocks.Interfaces.Adaptors.SendReal

Obsolete block to send Real signal to bus



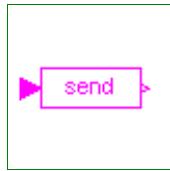
### Information

Obsolete block that was previously used to connect a Real signal to a signal in a connector. This block is only provided for backward compatibility.

It is much more convenient and more powerful to use "expandable connectors" for signal buses, see example [BusUsage](#).

### Connectors

Name	Description
toBus	Output signal to be connected to bus
u	Input signal to be send to bus

**Modelica.Blocks.Interfaces.Adaptors.SendBoolean****Obsolete block to send Boolean signal to bus****Information**

Obsolete block that was previously used to connect a Boolean signal to a signal in a connector. This block is only provided for backward compatibility.

It is much more convenient and more powerful to use "expandable connectors" for signal buses, see example [BusUsage](#).

**Connectors**

Name	Description
toBus	Output signal to be connected to bus
u	Input signal to be send to bus

---

**Modelica.Blocks.Interfaces.Adaptors.SendInteger****Obsolete block to send Integer signal to bus****Information**

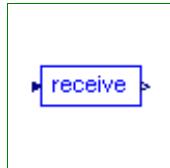
Obsolete block that was previously used to connect an Integer signal to a signal in a connector. This block is only provided for backward compatibility.

It is much more convenient and more powerful to use "expandable connectors" for signal buses, see example [BusUsage](#).

**Connectors**

Name	Description
toBus	Output signal to be connected to bus
u	Input signal to be send to bus

---

**Modelica.Blocks.Interfaces.Adaptors.ReceiveReal****Obsolete block to receive Real signal from bus****Information**

Obsolete block that was previously used to connect a Real signal in a connector to an input of a block. This block is only provided for backward compatibility.

It is much more convenient and more powerful to use "expandable connectors" for signal buses, see example [BusUsage](#).

**Connectors**

Name	Description
fromBus	To be connected with signal on bus
y	Output signal to be received from bus

---

**Modelica.Blocks.Interfaces.Adaptors.ReceiveBoolean****Obsolete block to receive Boolean signal from bus****Information**

Obsolete block that was previously used to connect a Boolean signal in a connector to an input of a block. This block is only provided for backward compatibility.

It is much more convenient and more powerful to use "expandable connectors" for signal buses, see example [BusUsage](#).

**Connectors**

Name	Description
fromBus	To be connected with signal on bus
y	Output signal to be received from bus

**Modelica.Blocks.Interfaces.Adaptors.ReceiveInteger****Obsolete block to receive Integer signal from bus****Information**

Obsolete block that was previously used to connect an Integer signal in a connector to an input of a block. This block is only provided for backward compatibility.

It is much more convenient and more powerful to use "expandable connectors" for signal buses, see example [BusUsage](#).

**Connectors**

Name	Description
fromBus	To be connected with signal on bus
y	Output signal to be received from bus

**Modelica.Blocks.Interfaces.Adaptors.AdaptorReal****Completely obsolete adaptor between 'old' and 'new' Real signal connectors (only for backward compatibility)****Information**

Completely obsolete adaptor between the Real signal connector of version 1.6 and version  $\geq 2.1$  of the Modelica Standard Library. This block is only provided for backward compatibility.

**Connectors**

Name	Description
newReal	Connector of Modelica version 2.1
oldReal	Connector of Modelica version 1.6

**Modelica.Blocks.Interfaces.Adaptors.AdaptorBoolean**

Completely obsolete adaptor between 'old' and 'new' Boolean signal connectors (only for backward compatibility)

**Information**

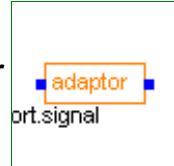
Completely obsolete adaptor between the Real signal connector of version 1.6 and version  $\geq 2.1$  of the Modelica Standard Library. This block is only provided for backward compatibility.

**Connectors**

Name	Description
newBoolean	Connector of Modelica version 2.1
oldBoolean	Connector of Modelica version 1.6

**Modelica.Blocks.Interfaces.Adaptors.AdaptorInteger**

Completely obsolete adaptor between 'old' and 'new' Integer signal connectors (only for backward compatibility)

**Information**

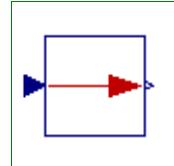
Completely obsolete adaptor between the Real signal connector of version 1.6 and version  $\geq 2.1$  of the Modelica Standard Library. This block is only provided for backward compatibility.

**Connectors**

Name	Description
newInteger	Connector of Modelica version 2.1
oldInteger	Connector of Modelica version 1.6

**Modelica.Blocks.Interfaces.PartialConversionBlock**

Partial block defining the interface for conversion blocks

**Information**

This block defines the interface of a conversion block that converts from one unit into another one.

**Connectors**

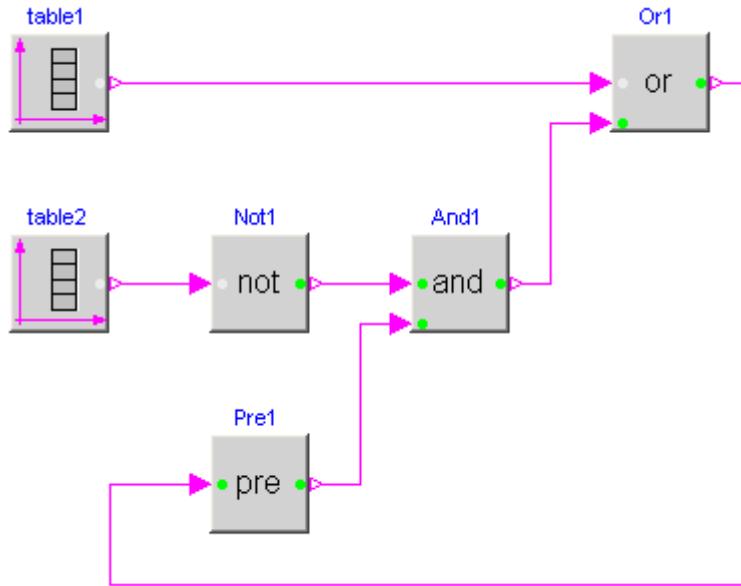
Name	Description
u	Connector of Real input signal to be converted
y	Connector of Real output signal containing input signal u in another unit

**Modelica.Blocks.Logical**

Library of components with Boolean input and output signals

## Information

This package provides blocks with Boolean input and output signals to describe logical networks. A typical example for a logical network built with package Logical is shown in the next figure:



The actual value of Boolean input and/or output signals is displayed in the respective block icon as "circle", where "white" color means value **false** and "green" color means value **true**. These values are visualized in a diagram animation.

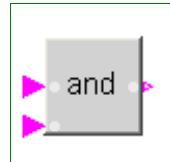
## Package Content

Name	Description
And	Logical 'and': $y = u_1 \text{ and } u_2$
Or	Logical 'or': $y = u_1 \text{ or } u_2$
Xor	Logical 'xor': $y = u_1 \text{ xor } u_2$
Nor	Logical 'nor': $y = \text{not}(u_1 \text{ or } u_2)$
Nand	Logical 'nand': $y = \text{not}(u_1 \text{ and } u_2)$
Not	Logical 'not': $y = \text{not } u$
Pre	Breaks algebraic loops by an infinitesimal small time delay ( $y = \text{pre}(u)$ : event iteration continues until $u = \text{pre}(u)$ )
Edge	Output $y$ is true, if the input $u$ has a rising edge ( $y = \text{edge}(u)$ )
FallingEdge	Output $y$ is true, if the input $u$ has a falling edge ( $y = \text{edge}(\text{not } u)$ )
Change	Output $y$ is true, if the input $u$ has a rising or falling edge ( $y = \text{change}(u)$ )
GreaterThreshold	Output $y$ is true, if input $u$ is greater than threshold
GreaterEqualThreshold	Output $y$ is true, if input $u$ is greater or equal than threshold
LessThreshold	Output $y$ is true, if input $u$ is less than threshold
LessEqualThreshold	Output $y$ is true, if input $u$ is less or equal than threshold
Greater	Output $y$ is true, if input $u_1$ is greater as input $u_2$
GreaterEqual	Output $y$ is true, if input $u_1$ is greater or equal as input $u_2$

 Less	Output y is true, if input u1 is less as input u2
 LessEqual	Output y is true, if input u1 is less or equal as input u2
 ZeroCrossing	Trigger zero crossing of input u
 LogicalSwitch	Logical Switch
 Switch	Switch between two Real signals
 Hysteresis	Transform Real to Boolean signal with Hysteresis
 OnOffController	On-off controller
 TriggeredTrapezoid	Triggered trapezoid generator
 Timer	Timer measuring the time from the time instant where the Boolean input became true
TerminateSimulation	Terminate simulation if condition is fulfilled

## Modelica.Blocks.Logical.And

Logical 'and':  $y = u1 \text{ and } u2$



### Information

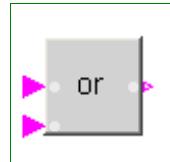
The output is **true** if all inputs are **true**, otherwise the output is **false**.

### Connectors

Name	Description
u1	Connector of first Boolean input signal
u2	Connector of second Boolean input signal
y	Connector of Boolean output signal

## Modelica.Blocks.Logical.Or

Logical 'or':  $y = u1 \text{ or } u2$



### Information

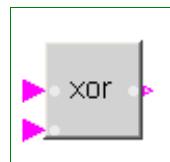
The output is **true** if at least one input is **true**, otherwise the output is **false**.

### Connectors

Name	Description
u1	Connector of first Boolean input signal
u2	Connector of second Boolean input signal
y	Connector of Boolean output signal

## Modelica.Blocks.Logical.Xor

Logical 'xor':  $y = u1 \text{ xor } u2$



## Information

The output is **true** if exactly one input is **true**, otherwise the output is **false**.

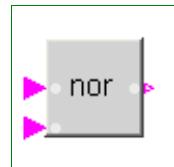
## Connectors

Name	Description
u1	Connector of first Boolean input signal
u2	Connector of second Boolean input signal
y	Connector of Boolean output signal

---

## Modelica.Blocks.Logical.Nor

Logical 'nor':  $y = \text{not}(u1 \text{ or } u2)$



## Information

The output is **true** if none of the inputs is **true**, otherwise the output is **false**.

## Connectors

Name	Description
u1	Connector of first Boolean input signal
u2	Connector of second Boolean input signal
y	Connector of Boolean output signal

---

## Modelica.Blocks.Logical.Nand

Logical 'nand':  $y = \text{not}(u1 \text{ and } u2)$



## Information

The output is **true** if at least one input is **false**, otherwise the output is **false**.

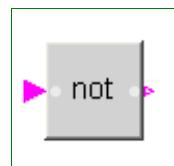
## Connectors

Name	Description
u1	Connector of first Boolean input signal
u2	Connector of second Boolean input signal
y	Connector of Boolean output signal

---

## Modelica.Blocks.Logical.Not

Logical 'not':  $y = \text{not } u$



## Information

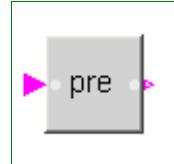
The output is **true** if the input is **false**, otherwise the output is **false**.

## Connectors

Name	Description
u	Connector of Boolean input signal
y	Connector of Boolean output signal

## Modelica.Blocks.Logical.Pre

**Breaks algebraic loops by an infinitesimal small time delay ( $y = \text{pre}(u)$ : event iteration continues until  $u = \text{pre}(u)$ )**



## Information

This block delays the Boolean input by an infinitesimal small time delay and therefore breaks algebraic loops. In a network of logical blocks, in every "closed connection loop" at least one logical block must have a delay, since algebraic systems of Boolean equations are not solveable.

The "Pre" block returns the value of the "input" signal from the last "event iteration". The "event iteration" stops, once both values are identical ( $u = \text{pre}(u)$ ).

## Parameters

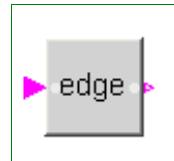
Name	Default	Description
pre_u_start	false	Start value of pre(u) at initial time

## Connectors

Name	Description
u	Connector of Boolean input signal
y	Connector of Boolean output signal

## Modelica.Blocks.Logical.Edge

**Output y is true, if the input u has a rising edge ( $y = \text{edge}(u)$ )**



## Information

The output is **true** if the Boolean input has a rising edge from **false** to **true**, otherwise the output is **false**.

## Parameters

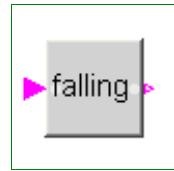
Name	Default	Description
pre_u_start	false	Start value of pre(u) at initial time

## Connectors

Name	Description
u	Connector of Boolean input signal
y	Connector of Boolean output signal

## Modelica.Blocks.Logical.FallingEdge

Output  $y$  is true, if the input  $u$  has a falling edge ( $y = \text{edge}(\text{not } u)$ )



### Information

The output is **true** if the Boolean input has a falling edge from **true** to **false**, otherwise the output is **false**.

### Parameters

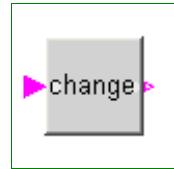
Name	Default	Description
pre_u_start	false	Start value of pre( $u$ ) at initial time

### Connectors

Name	Description
u	Connector of Boolean input signal
y	Connector of Boolean output signal

## Modelica.Blocks.Logical.Change

Output  $y$  is true, if the input  $u$  has a rising or falling edge ( $y = \text{change}(u)$ )



### Information

The output is **true** if the Boolean input has either a rising edge from **false** to **true** or a falling edge from **true** to **false**, otherwise the output is **false**.

### Parameters

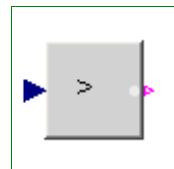
Name	Default	Description
pre_u_start	false	Start value of pre( $u$ ) at initial time

### Connectors

Name	Description
u	Connector of Boolean input signal
y	Connector of Boolean output signal

## Modelica.Blocks.Logical.GreaterThreshold

Output  $y$  is true, if input  $u$  is greater than threshold



### Information

The output is **true** if the Real input is greater than parameter **threshold**, otherwise the output is **false**.

### Parameters

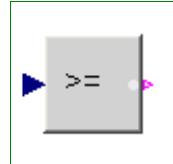
Name	Default	Description
threshold	0	Comparison with respect to threshold

## Connectors

Name	Description
u	Connector of Boolean input signal
y	Connector of Boolean output signal

## Modelica.Blocks.Logical.GreaterEqualThreshold

Output y is true, if input u is greater or equal than threshold



## Information

The output is **true** if the Real input is greater than or equal to parameter **threshold**, otherwise the output is **false**.

## Parameters

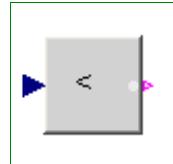
Name	Default	Description
threshold	0	Comparison with respect to threshold

## Connectors

Name	Description
u	Connector of Boolean input signal
y	Connector of Boolean output signal

## Modelica.Blocks.Logical.LessThreshold

Output y is true, if input u is less than threshold



## Information

The output is **true** if the Real input is less than parameter **threshold**, otherwise the output is **false**.

## Parameters

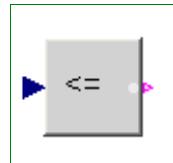
Name	Default	Description
threshold	0	Comparison with respect to threshold

## Connectors

Name	Description
u	Connector of Boolean input signal
y	Connector of Boolean output signal

## Modelica.Blocks.Logical.LessEqualThreshold

Output y is true, if input u is less or equal than threshold



## Information

The output is **true** if the Real input is less than or equal to parameter **threshold**, otherwise the output is **false**.

## Parameters

Name	Default	Description
threshold	0	Comparison with respect to threshold

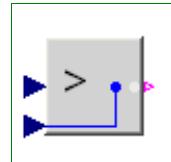
## Connectors

Name	Description
u	Connector of Boolean input signal
y	Connector of Boolean output signal

---

## Modelica.Blocks.Logical.Greater

Output **y** is **true**, if input **u1** is greater as input **u2**



## Information

The output is **true** if Real input **u1** is greater than Real input **u2**, otherwise the output is **false**.

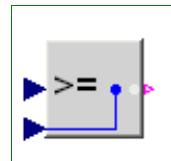
## Connectors

Name	Description
u1	Connector of first Boolean input signal
u2	Connector of second Boolean input signal
y	Connector of Boolean output signal

---

## Modelica.Blocks.Logical.GreaterEqual

Output **y** is **true**, if input **u1** is greater or equal as input **u2**



## Information

The output is **true** if Real input **u1** is greater than or equal to Real input **u2**, otherwise the output is **false**.

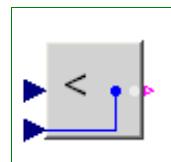
## Connectors

Name	Description
u1	Connector of first Boolean input signal
u2	Connector of second Boolean input signal
y	Connector of Boolean output signal

---

## Modelica.Blocks.Logical.Less

Output **y** is **true**, if input **u1** is less as input **u2**



## Information

The output is **true** if Real input u1 is less than Real input u2, otherwise the output is **false**.

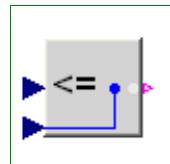
## Connectors

Name	Description
u1	Connector of first Boolean input signal
u2	Connector of second Boolean input signal
y	Connector of Boolean output signal

---

## Modelica.Blocks.Logical.LessEqual

Output y is true, if input u1 is less or equal as input u2



## Information

The output is **true** if Real input u1 is less than or equal to Real input u2, otherwise the output is **false**.

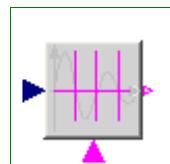
## Connectors

Name	Description
u1	Connector of first Boolean input signal
u2	Connector of second Boolean input signal
y	Connector of Boolean output signal

---

## Modelica.Blocks.Logical.ZeroCrossing

Trigger zero crossing of input u



## Information

The output "y" is **true** at the time instant when the input "u" becomes zero, provided the input "enable" is **true**. At all other time instants, the output "y" is **false**. If the input "u" is zero at a time instant when the "enable" input changes its value, then the output y is **false**.

Note, that in the plot window of a Modelica simulator, the output of this block is usually identically to **false**, because the output may only be **true** at an event instant, but not during continuous integration. In order to check that this component is actually working as expected, one should connect its output to, e.g., component *ModelicaAdditions.Blocks.Discrete.TriggeredSampler*.

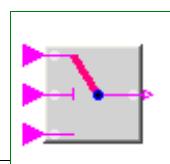
## Connectors

Name	Description
y	Connector of Boolean output signal
u	
enable	Zero input crossing is triggered if the enable input signal is true

---

## Modelica.Blocks.Logical.LogicalSwitch

Logical Switch



## Information

The LogicalSwitch switches, depending on the Boolean u2 connector (the middle connector), between the two possible input signals u1 (upper connector) and u3 (lower connector).

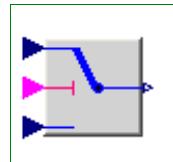
If u2 is true, connector y is set equal to u1, else it is set equal to u2.

## Connectors

Name	Description
u1	Connector of first Boolean input signal
u2	Connector of second Boolean input signal
u3	Connector of third Boolean input signal
y	Connector of Boolean output signal

## Modelica.Blocks.Logical.Switch

### Switch between two Real signals



## Information

The Logical.Switch switches, depending on the logical connector u2 (the middle connector) between the two possible input signals u1 (upper connector) and u3 (lower connector).

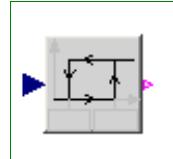
If u2 is **true**, the output signal y is set equal to u1, else it is set equal to u3.

## Connectors

Name	Description
u1	Connector of first Real input signal
u2	Connector of Boolean input signal
u3	Connector of second Real input signal
y	Connector of Real output signal

## Modelica.Blocks.Logical.Hysteresis

### Transform Real to Boolean signal with Hysteresis



## Information

This block transforms a **Real** input signal into a **Boolean** output signal:

- When the output was **false** and the input becomes **greater** than parameter **uHigh**, the output switches to **true**.
- When the output was **true** and the input becomes **less** than parameter **uLow**, the output switches to **false**.

The start value of the output is defined via parameter **pre\_y\_start** (= value of pre(y) at initial time). The default value of this parameter is **false**.

## Parameters

Name	Default	Description
uLow	0	if y=true and u<=uLow, switch to y=false

## 132 Modelica.Blocks.Logical.Hysteresis

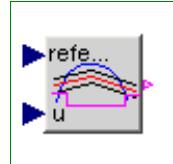
uHigh	1	if $y=\text{false}$ and $u>=u\text{High}$ , switch to $y=\text{true}$
pre_y_start	false	Value of pre( $y$ ) at initial time

### Connectors

Name	Description
u	
y	

## Modelica.Blocks.Logical.OnOffController

On-off controller



### Information

The block OnOffController sets the output signal  $y$  to **true** when the input signal  $u$  falls below the **reference** signal minus half of the bandwidth and sets the output signal  $y$  to **false** when the input signal  $u$  exceeds the **reference** signal plus half of the bandwidth.

### Parameters

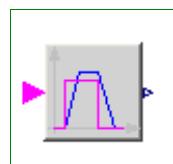
Name	Default	Description
bandwidth	0.1	Bandwidth around reference signal
pre_y_start	false	Value of pre( $y$ ) at initial time

### Connectors

Name	Description
reference	Connector of Real input signal used as reference signal
u	Connector of Real input signal used as measurement signal
y	Connector of Real output signal used as actuator signal

## Modelica.Blocks.Logical.TriggeredTrapezoid

Triggered trapezoid generator



### Information

The block TriggeredTrapezoid has a boolean input and a real output signal and requires the parameters *amplitude*, *rising*, *falling* and *offset*. The output signal  $y$  represents a trapezoidal signal dependent on the input signal  $u$ .

The behaviour is as follows: Assume the initial input to be false. In this case, the output will be *offset*. After a rising edge (i.e. the input changes from false to true), the output is rising during *rising* to the sum of *offset* and *amplitude*. In contrast, after a falling edge (i.e. the input changes from true to false), the output is falling during *falling* to a value of *offset*.

Note, that the case of edges before expiration of rising or falling is handled properly.

### Parameters

Name	Default	Description
amplitude	1	Amplitude of trapezoid

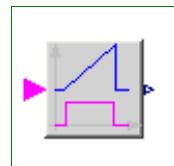
rising	0	Rising duration of trapezoid [s]
falling	rising	Falling duration of trapezoid [s]
offset	0	Offset of output signal

## Connectors

Name	Description
u	Connector of Boolean input signal
y	Connector of Real output signal

## Modelica.Blocks.Logical.Timer

Timer measuring the time from the time instant where the Boolean input became true



## Information

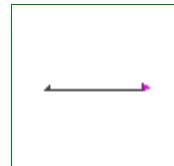
When the Boolean input "u" becomes **true**, the timer is started and the output "y" is the time from the time instant where u became true. The timer is stopped and the output is reset to zero, once the input becomes false.

## Connectors

Name	Description
u	Connector of Boolean input signal
y	Connector of Real output signal

## Modelica.Blocks.Logical.TerminateSimulation

Terminate simulation if condition is fulfilled



## Information

In the parameter menu, a **time varying** expression can be defined via variable **condition**, for example "condition = x < 0", where "x" is a variable that is declared in the model in which the "TerminateSimulation" block is present. If this expression becomes **true**, the simulation is (successfully) terminated. A termination message explaining the reason for the termination can be given via parameter "terminationText".

## Parameters

Name	Default	Description
condition	false	Terminate simulation when condition becomes true
terminationText	"... End condition reached"	Text that will be displayed when simulation is terminated

## Connectors

Name	Description
condition	Terminate simulation when condition becomes true

## Modelica.Blocks.Math

Library of mathematical functions as input/output blocks

## Information

This package contains basic **mathematical operations**, such as summation and multiplication, and basic **mathematical functions**, such as `sqrt` and `sin`, as input/output blocks. All blocks of this library can be either connected with continuous blocks or with sampled-data blocks.

## Package Content

Name	Description
 UnitConversions	Conversion blocks to convert between SI and non-SI unit signals
 TwoInputs	Change causality of input signals by defining that two input signals are identical (e.g. for inverse models)
 TwoOutputs	Change causality of output signals by defining that two output signals are identical (e.g. for inverse models)
 Gain	Output the product of a gain value with the input signal
 MatrixGain	Output the product of a gain matrix with the input signal vector
 Sum	Output the sum of the elements of the input vector
 Feedback	Output difference between commanded and feedback input
 Add	Output the sum of the two inputs
 Add3	Output the sum of the three inputs
 Product	Output product of the two inputs
 Division	Output first input divided by second input
 Abs	Output the absolute value of the input
 Sign	Output the sign of the input
 Sqrt	Output the square root of the input (input $\geq 0$ required)
 Sin	Output the sine of the input
 Cos	Output the cosine of the input
 Tan	Output the tangent of the input
 Asin	Output the arc sine of the input
 Acos	Output the arc cosine of the input
 Atan	Output the arc tangent of the input
 Atan2	Output $\text{atan}(u_1/u_2)$ of the inputs $u_1$ and $u_2$
 Sinh	Output the hyperbolic sine of the input
 Cosh	Output the hyperbolic cosine of the input
 Tanh	Output the hyperbolic tangent of the input
 Exp	Output the exponential (base e) of the input
 Log	Output the natural (base e) logarithm of the input (input $> 0$ required)
 Log10	Output the base 10 logarithm of the input (input $> 0$ required)
 RealToInteger	Convert Real to Integer signal
 IntegerToReal	Convert integer to real signals
 BooleanToReal	Convert Boolean to Real signal

 BooleanToInteger	Convert Boolean to Integer signal
 RealToBoolean	Convert Real to Boolean signal
 IntegerToBoolean	Convert Integer to Boolean signal
 Max	Pass through the largest signal
 Min	Pass through the smallest signal
 Edge	Indicates rising edge of boolean signal
 BooleanChange	Indicates boolean signal changing
 IntegerChange	Indicates integer signal changing

## Modelica.Blocks.Math.UnitConversions

Conversion blocks to convert between SI and non-SI unit signals

### Information

This package consists of blocks that convert an input signal with a specific unit to an output signal in another unit (e.g. conversion of an angle signal from "deg" to "rad"). Block "ConvertAllUnits" converts between a set of units that can be selected in a pull-down menu of the parameter menu. All other blocks convert exactly between two different units.

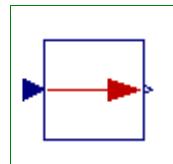
### Package Content

Name	Description
 ConvertAllUnits	Convert signal to a signal with different unit
 To_degC	Convert from Kelvin to °Celsius
 From_degC	Convert from °Celsius to Kelvin
 To_degF	Convert from Kelvin to °Fahrenheit
 From_degF	Convert from °Fahrenheit to Kelvin
 To_degRk	Convert from Kelvin to °Rankine
 From_degRk	Convert from °Rankine to Kelvin
 To_deg	Convert from radian to degree
 From_deg	Convert from degree to radian
 To_rpm	Convert from radian per second to revolutions per minute
 From_rpm	Convert from revolutions per minute to radian per second
 To_kmh	Convert from metre per second to kilometre per hour
 From_kmh	Convert from kilometre per hour to metre per second
 To_day	Convert from second to day
 From_day	Convert from day to second
 To_hour	Convert from second to hour
 From_hour	Convert from hour to second
 To_minute	Convert from second to minute

 <a href="#">From_minute</a>	Convert from minute to second
 <a href="#">To_litre</a>	Convert from cubic metre to litre
 <a href="#">From_litre</a>	Convert from litre to cubic metre
 <a href="#">To_kWh</a>	Convert from Joule to kilo Watt hour
 <a href="#">From_kWh</a>	Convert from kilo Watt hour to Joule
 <a href="#">To_bar</a>	Convert from Pascal to bar
 <a href="#">From_bar</a>	Convert from bar to Pascal
 <a href="#">To_gps</a>	Convert from kilogram per second to gram per second
 <a href="#">From_gps</a>	Convert from gram per second to kilogram per second

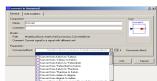
## Modelica.Blocks.Math.UnitConversions.ConvertAllUnits

Convert signal to a signal with different unit



### Information

This block implements the Modelica.SIunits.Conversions functions as a fixed causality block to simplify their use. The block contains a replaceable block class **ConversionBlock** that can be changed to be any of the blocks defined in Modelica.Blocks.Math.UnitConversions, and more generally, any blocks that extend from Modelica.Blocks.Interfaces.PartialConversionBlock.

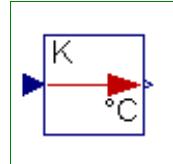


### Connectors

Name	Description
u	Connector of Real input signal to be converted
y	Connector of Real output signal containing input signal u in another unit

## Modelica.Blocks.Math.UnitConversions.To\_degC

Convert from Kelvin to °Celsius

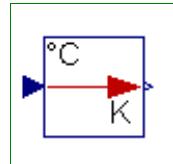


### Information

This block converts the input signal from Kelvin to °Celsius and returns the result as output signal.

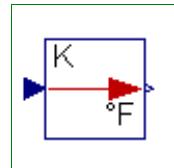
## Modelica.Blocks.Math.UnitConversions.From\_degC

Convert from °Celsius to Kelvin

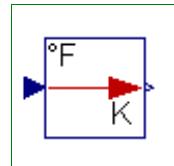


### Information

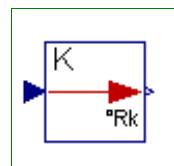
This block converts the input signal from °Celsius to Kelvin and returns the result as output signal.

**Modelica.Blocks.Math.UnitConversions.To\_degF****Convert from Kelvin to °Fahrenheit****Information**

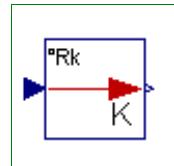
This block converts the input signal from Kelvin to °Fahrenheit and returns the result as output signal.

**Modelica.Blocks.Math.UnitConversions.From\_degF****Convert from °Fahrenheit to Kelvin****Information**

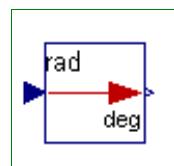
This block converts the input signal from °Fahrenheit to Kelvin and returns the result as output signal.

**Modelica.Blocks.Math.UnitConversions.To\_degRk****Convert from Kelvin to °Rankine****Information**

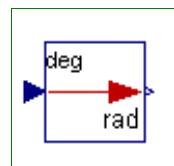
This block converts the input signal from Kelvin to °Rankine and returns the result as output signal.

**Modelica.Blocks.Math.UnitConversions.From\_degRk****Convert from °Rankine to Kelvin****Information**

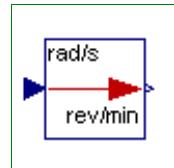
This block converts the input signal from °Rankine to Kelvin and returns the result as output signal.

**Modelica.Blocks.Math.UnitConversions.To\_deg****Convert from radian to degree****Information**

This block converts the input signal from radian to degree and returns the result as output signal.

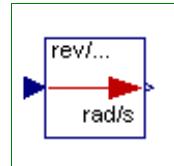
**Modelica.Blocks.Math.UnitConversions.From\_deg****Convert from degree to radian****Information**

This block converts the input signal from degree to radian and returns the result as output signal.

**Modelica.Blocks.Math.UnitConversions.To\_rpm****Convert from radian per second to revolutions per minute****Information**

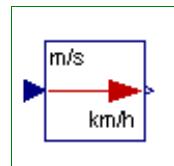
This block converts the input signal from radian per second to revolutions per minute and returns the result as output signal.

---

**Modelica.Blocks.Math.UnitConversions.From\_rpm****Convert from revolutions per minute to radian per second****Information**

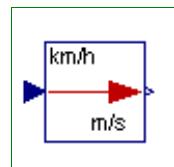
This block converts the input signal from revolutions per minute to radian per second and returns the result as output signal.

---

**Modelica.Blocks.Math.UnitConversions.To\_kmh****Convert from metre per second to kilometre per hour****Information**

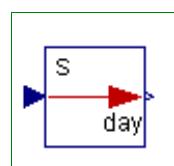
This block converts the input signal from metre per second to kilometre per hour and returns the result as output signal.

---

**Modelica.Blocks.Math.UnitConversions.From\_kmh****Convert from kilometre per hour to metre per second****Information**

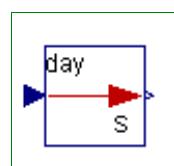
This block converts the input signal from kilometre per hour to metre per second and returns the result as output signal.

---

**Modelica.Blocks.Math.UnitConversions.To\_day****Convert from second to day****Information**

This block converts the input signal from second to day and returns the result as output signal.

---

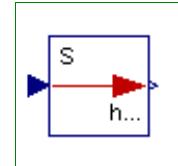
**Modelica.Blocks.Math.UnitConversions.From\_day****Convert from day to second**

**Information**

This block converts the input signal from day to second and returns the result as output signal.

**Modelica.Blocks.Math.UnitConversions.To\_hour**

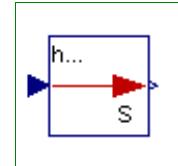
**Convert from second to hour**

**Information**

This block converts the input signal from second to hour and returns the result as output signal.

**Modelica.Blocks.Math.UnitConversions.From\_hour**

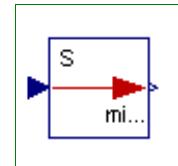
**Convert from hour to second**

**Information**

This block converts the input signal from hour to second and returns the result as output signal.

**Modelica.Blocks.Math.UnitConversions.To\_minute**

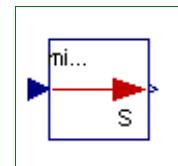
**Convert from second to minute**

**Information**

This block converts the input signal from second to minute and returns the result as output signal.

**Modelica.Blocks.Math.UnitConversions.From\_minute**

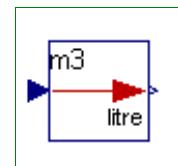
**Convert from minute to second**

**Information**

This block converts the input signal from minute to second and returns the result as output signal.

**Modelica.Blocks.Math.UnitConversions.To\_litre**

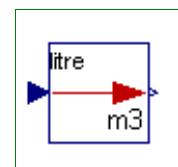
**Convert from cubic metre to litre**

**Information**

This block converts the input signal from metre to litre and returns the result as output signal.

**Modelica.Blocks.Math.UnitConversions.From\_litre**

**Convert from litre to cubic metre**



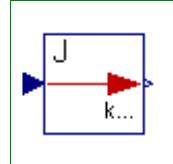
### Information

This block converts the input signal from litre to cubic metre and returns the result as output signal.

---

### Modelica.Blocks.Math.UnitConversions.To\_kWh

Convert from Joule to kilo Watt hour



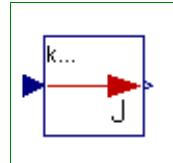
### Information

This block converts the input signal from Joule to kilo Watt hour and returns the result as output signal.

---

### Modelica.Blocks.Math.UnitConversions.From\_kWh

Convert from kilo Watt hour to Joule



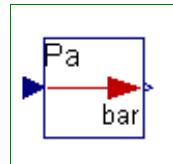
### Information

This block converts the input signal from kilo Watt hour to Joule and returns the result as output signal.

---

### Modelica.Blocks.Math.UnitConversions.To\_bar

Convert from Pascal to bar



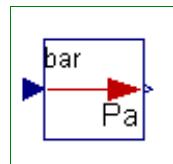
### Information

This block converts the input signal from Pascal to bar and returns the result as output signal.

---

### Modelica.Blocks.Math.UnitConversions.From\_bar

Convert from bar to Pascal



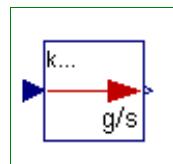
### Information

This block converts the input signal from bar to Pascal and returns the result as output signal.

---

### Modelica.Blocks.Math.UnitConversions.To\_gps

Convert from kilogram per second to gram per second



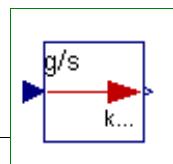
### Information

This block converts the input signal from kilogram per second to gram per seconds and returns the result as output signal.

---

### Modelica.Blocks.Math.UnitConversions.From\_gps

Convert from gram per second to kilogram per second

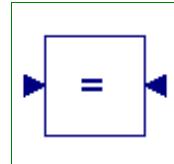


## Information

This block converts the input signal from gram per second to kilogram per second and returns the result as output signal.

## Modelica.Blocks.Math.TwoInputs

Change causality of input signals by defining that two input signals are identical (e.g. for inverse models)



## Information

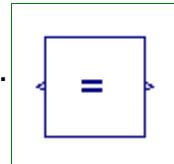
This block is used to enable assignment of values to variables preliminary defined as outputs (e.g. useful for inverse model generation).

## Connectors

Name	Description
u1	Connector of first Real input signal
u2	Connector of second Real input signal ( $u1=u2$ )

## Modelica.Blocks.Math.TwoOutputs

Change causality of output signals by defining that two output signals are identical (e.g. for inverse models)



## Information

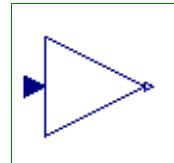
This block is used to enable calculation of values preliminary defined as inputs. (e.g. useful for inverse model generation).

## Connectors

Name	Description
y1	Connector of first Real output signal
y2	Connector of second Real output signal ( $y1=y2$ )

## Modelica.Blocks.Math.Gain

Output the product of a gain value with the input signal



## Information

This block computes output  $y$  as *product* of gain  $k$  with the input  $u$ :

$$y = k * u;$$

## Parameters

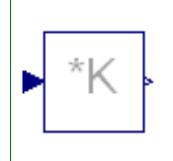
Name	Default	Description
k	1	Gain value multiplied with input signal

## Connectors

Name	Description
u	Input signal connector
y	Output signal connector

## Modelica.Blocks.Math.MatrixGain

Output the product of a gain matrix with the input signal vector



## Information

This blocks computes output vector **y** as *product* of the gain matrix **K** with the input signal vector **u**:

$$\mathbf{y} = \mathbf{K} * \mathbf{u};$$

Example:

```
parameter: K = [0.12 2; 3 1.5]
```

results in the following equations:

$$\begin{vmatrix} | & y[1] & | \\ | & & = & | & 0.12 & 2.00 & | & * & | & u[1] & | \\ | & y[2] & | & & | & 3.00 & 1.50 & | & | & u[2] & | \end{vmatrix}$$

## Parameters

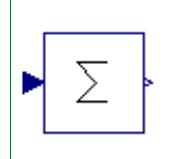
Name	Default	Description
K[:, :]	[1, 0; 0, 1]	Gain matrix which is multiplied with the input

## Connectors

Name	Description
u[nin]	Connector of Real input signals
y[nout]	Connector of Real output signals

## Modelica.Blocks.Math.Sum

Output the sum of the elements of the input vector



## Information

This blocks computes output **y** as *sum* of the elements of the input signal vector **u**:

$$\mathbf{y} = \mathbf{u}[1] + \mathbf{u}[2] + \dots;$$

Example:

```
parameter: nin = 3;
```

results in the following equations:

$$y = u[1] + u[2] + u[3];$$

## Parameters

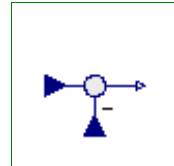
Name	Default	Description
nin	1	Number of inputs
k[nin]	ones(nin)	Optional: sum coefficients

## Connectors

Name	Description
u[nin]	Connector of Real input signals
y	Connector of Real output signal

## Modelica.Blocks.Math.Feedback

Output difference between commanded and feedback input



## Information

This blocks computes output **y** as *difference* of the commanded input **u1** and the feedback input **u2**:

$$y = u1 - u2;$$

Example:

```
parameter: n = 2
```

results in the following equations:

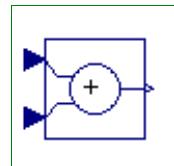
$$y = u1 - u2$$

## Connectors

Name	Description
u1	
u2	
y	

## Modelica.Blocks.Math.Add

Output the sum of the two inputs



## Information

This blocks computes output **y** as *sum* of the two input signals **u1** and **u2**:

$$y = k1*u1 + k2*u2;$$

Example:

```
parameter: k1= +2, k2= -3
```

results in the following equations:

$$y = 2 * u1 - 3 * u2$$

## Parameters

Name	Default	Description
k1	+1	Gain of upper input
k2	+1	Gain of lower input

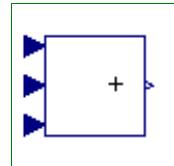
## Connectors

Name	Description
u1	Connector of Real input signal 1
u2	Connector of Real input signal 2
y	Connector of Real output signal

---

## Modelica.Blocks.Math.Add3

Output the sum of the three inputs



## Information

This blocks computes output **y** as *sum* of the three input signals **u1**, **u2** and **u3**:

$$y = k1*u1 + k2*u2 + k3*u3;$$

Example:

```
parameter:    k1= +2, k2= -3, k3=1;
```

results in the following equations:

$$y = 2 * u1 - 3 * u2 + u3;$$

## Parameters

Name	Default	Description
k1	+1	Gain of upper input
k2	+1	Gain of middle input
k3	+1	Gain of lower input

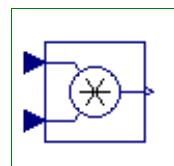
## Connectors

Name	Description
u1	Connector 1 of Real input signals
u2	Connector 2 of Real input signals
u3	Connector 3 of Real input signals
y	Connector of Real output signals

---

## Modelica.Blocks.Math.Product

Output product of the two inputs



## Information

This blocks computes the output **y** (element-wise) as *product* of the corresponding elements of the two inputs **u1** and **u2**:

$$y = u1 * u2;$$

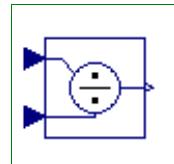
## Connectors

Name	Description
u1	Connector of Real input signal 1
u2	Connector of Real input signal 2
y	Connector of Real output signal

---

## Modelica.Blocks.Math.Division

Output first input divided by second input



## Information

This block computes the output **y** (element-wise) by *dividing* the corresponding elements of the two inputs **u1** and **u2**:

$$y = u1 / u2;$$

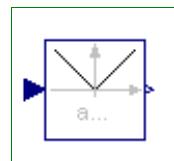
## Connectors

Name	Description
u1	Connector of Real input signal 1
u2	Connector of Real input signal 2
y	Connector of Real output signal

---

## Modelica.Blocks.Math.Abs

Output the absolute value of the input



## Information

This blocks computes the output **y** as *absolute value* of the input **u**:

$$y = \text{abs}(u);$$

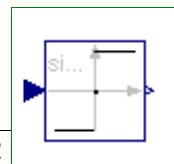
## Connectors

Name	Description
u	Connector of Real input signal
y	Connector of Real output signal

---

## Modelica.Blocks.Math.Sign

Output the sign of the input



## Information

This blocks computes the output **y** as **sign** of the input **u**:

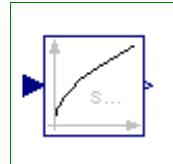
$$\begin{aligned} 1 & \quad \text{if } u > 0 \\ y = 0 & \quad \text{if } u == 0 \\ -1 & \quad \text{if } u < 0 \end{aligned}$$

## Connectors

Name	Description
u	Connector of Real input signal
y	Connector of Real output signal

## Modelica.Blocks.Math.Sqrt

Output the square root of the input (input  $\geq 0$  required)



## Information

This blocks computes the output **y** as *square root* of the input **u**:

$$y = \sqrt(u);$$

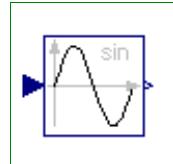
All elements of the input vector shall be zero or positive. Otherwise an error occurs.

## Connectors

Name	Description
u	Connector of Real input signal
y	Connector of Real output signal

## Modelica.Blocks.Math.Sin

Output the sine of the input



## Information

This blocks computes the output **y** as **sine** of the input **u**:

$$y = \sin(u);$$



## Connectors

Name	Description
u	Connector of Real input signal
y	Connector of Real output signal

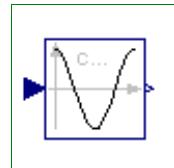
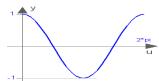
## Modelica.Blocks.Math.Cos

Output the cosine of the input

### Information

This blocks computes the output **y** as **cos** of the input **u**:

$$y = \cos(u);$$

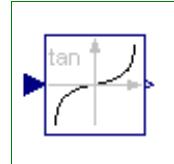


### Connectors

Name	Description
u	Connector of Real input signal
y	Connector of Real output signal

## Modelica.Blocks.Math.Tan

Output the tangent of the input



### Information

This blocks computes the output **y** as **tan** of the input **u**:

$$y = \tan(u);$$

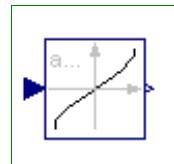


### Connectors

Name	Description
u	Connector of Real input signal
y	Connector of Real output signal

## Modelica.Blocks.Math.Asin

Output the arc sine of the input



### Information

This blocks computes the output **y** as the *sine-inverse* of the input **u**:

$$y = \text{asin}(u);$$

The absolute values of the elements of the input **u** need to be less or equal to one (**abs(u) <= 1**). Otherwise an error occurs.

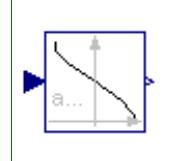


## Connectors

Name	Description
u	Connector of Real input signal
y	Connector of Real output signal

## Modelica.Blocks.Math.Acos

Output the arc cosine of the input



## Information

This blocks computes the output **y** as the *cosine-inverse* of the input **u**:

$$y = \text{acos}(u);$$

The absolute values of the elements of the input **u** need to be less or equal to one ( $\text{abs}(u) \leq 1$ ). Otherwise an error occurs.

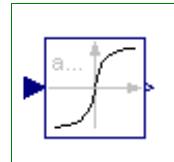


## Connectors

Name	Description
u	Connector of Real input signal
y	Connector of Real output signal

## Modelica.Blocks.Math.Atan

Output the arc tangent of the input



## Information

This blocks computes the output **y** as the *tangent-inverse* of the input **u**:

$$y = \text{atan}(u);$$

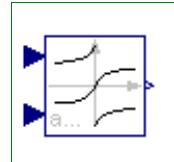


## Connectors

Name	Description
u	Connector of Real input signal
y	Connector of Real output signal

## Modelica.Blocks.Math.Atan2

Output  $\text{atan}(u_1/u_2)$  of the inputs **u1** and **u2**

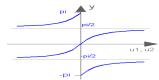


## Information

This blocks computes the output **y** as the *tangent-inverse* of the input **u1** divided by input **u2**:

$$y = \text{atan2}( u1, u2 );$$

**u1** and **u2** shall not be zero at the same time instant. **Atan2** uses the sign of **u1** and **u2** in order to construct the solution in the range  $-180 \text{ deg} \leq y \leq 180 \text{ deg}$ , whereas block **Atan** gives a solution in the range  $-90 \text{ deg} \leq y \leq 90 \text{ deg}$ .

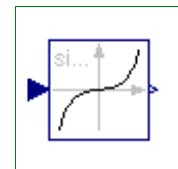


## Connectors

Name	Description
u1	Connector of Real input signal 1
u2	Connector of Real input signal 2
y	Connector of Real output signal

## Modelica.Blocks.Math.Sinh

Output the hyperbolic sine of the input



## Information

This blocks computes the output **y** as the *hyperbolic sine* of the input **u**:

$$y = \sinh( u );$$

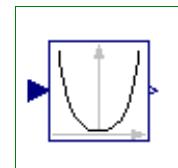


## Connectors

Name	Description
u	Connector of Real input signal
y	Connector of Real output signal

## Modelica.Blocks.Math.Cosh

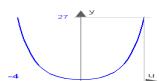
Output the hyperbolic cosine of the input



## Information

This blocks computes the output **y** as the *hyperbolic cosine* of the input **u**:

$$y = \cosh( u );$$



## 150 Modelica.Blocks.Math.Cosh

---

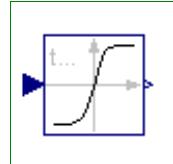
### Connectors

Name	Description
u	Connector of Real input signal
y	Connector of Real output signal

---

## Modelica.Blocks.Math.Tanh

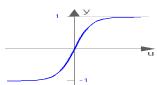
Output the hyperbolic tangent of the input



### Information

This blocks computes the output **y** as the *hyperbolic tangent* of the input **u**:

$$y = \tanh(u);$$



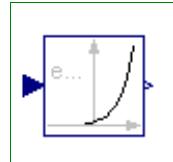
### Connectors

Name	Description
u	Connector of Real input signal
y	Connector of Real output signal

---

## Modelica.Blocks.Math.Exp

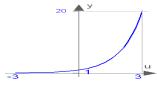
Output the exponential (base e) of the input



### Information

This blocks computes the output **y** as the *exponential* (of base e) of the input **u**:

$$y = \exp(u);$$



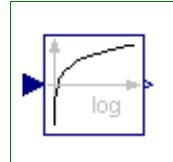
### Connectors

Name	Description
u	Connector of Real input signal
y	Connector of Real output signal

---

## Modelica.Blocks.Math.Log

Output the natural (base e) logarithm of the input (input > 0 required)

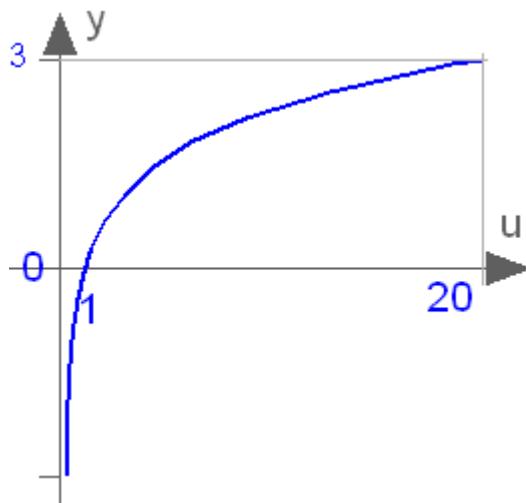


### Information

This blocks computes the output **y** as the *natural (base e) logarithm* of the input **u**:

```
y = log( u );
```

An error occurs if the elements of the input **u** are zero or negative.

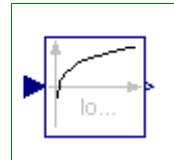


## Connectors

Name	Description
u	Connector of Real input signal
y	Connector of Real output signal

## Modelica.Blocks.Math.Log10

Output the base 10 logarithm of the input (input > 0 required)

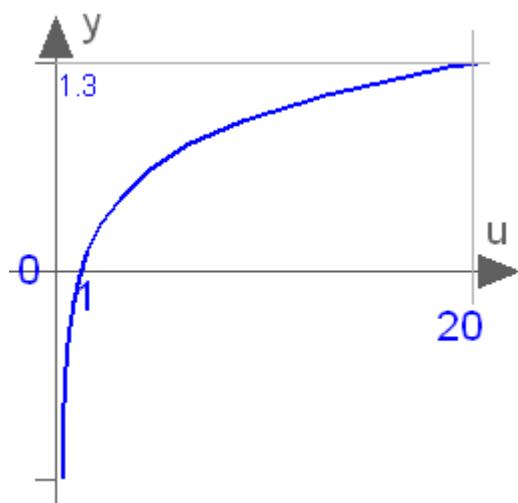


## Information

This blocks computes the output **y** as the *base 10 logarithm* of the input **u**:

```
y = log10( u );
```

An error occurs if the elements of the input **u** are zero or negative.

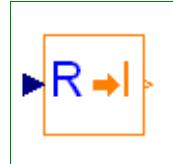


## Connectors

Name	Description
u	Connector of Real input signal
y	Connector of Real output signal

## Modelica.Blocks.Math.RealToInteger

Convert Real to Integer signal



## Information

This block computes the output **y** as *nearest integer value* of the input **u**:

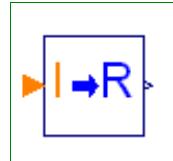
$$\begin{aligned} y &= \text{integer}(\text{floor}(u + 0.5)) \quad \text{for } u > 0; \\ y &= \text{integer}(\text{ceil}(u - 0.5)) \quad \text{for } u < 0; \end{aligned}$$

## Connectors

Name	Description
u	Connector of Real input signal
y	Connector of Integer output signal

## Modelica.Blocks.Math.IntegerToReal

Convert integer to real signals



## Information

This block computes the output **y** as *Real equivalent* of the Integer input **u**:

$$y = u;$$

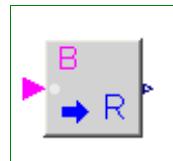
where **u** is of Integer and **y** of Real type.

## Connectors

Name	Description
u	Connector of Integer input signal
y	Connector of Real output signal

## Modelica.Blocks.Math.BooleanToReal

Convert Boolean to Real signal



## Information

This block computes the output **y** as *Real equivalent* of the Boolean input **u**:

$$y = \text{if } u \text{ then realTrue } \text{ else realFalse};$$

where **u** is of Boolean and **y** of Real type, and **realTrue** and **realFalse** are parameters.

## Parameters

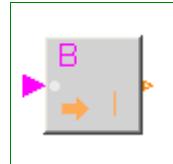
Name	Default	Description
realTrue	1.0	Output signal for true Boolean input
realFalse	0.0	Output signal for false Boolean input

## Connectors

Name	Description
u	Connector of Boolean input signal
y	Connector of Real output signal

## Modelica.Blocks.Math.BooleanToInteger

Convert Boolean to Integer signal



## Information

This block computes the output **y** as *Integer equivalent* of the Boolean input **u**:

$$y = \text{if } u \text{ then integerTrue else integerFalse;}$$

where **u** is of Boolean and **y** of Integer type, and **integerTrue** and **integerFalse** are parameters.

## Parameters

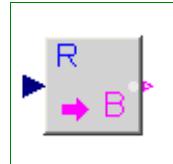
Name	Default	Description
integerTrue	1	Output signal for true Boolean input
integerFalse	0	Output signal for false Boolean input

## Connectors

Name	Description
u	Connector of Boolean input signal
y	Connector of Integer output signal

## Modelica.Blocks.Math.RealToBoolean

Convert Real to Boolean signal



## Information

This block computes the Boolean output **y** from the Real input **u** by the equation:

$$y = u \geq \text{threshold};$$

where **threshold** is a parameter.

## Parameters

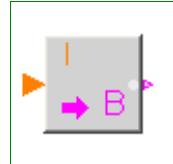
Name	Default	Description
threshold	0.5	Output signal y is true, if input u >= threshold

## Connectors

Name	Description
u	Connector of Real input signal
y	Connector of Boolean output signal

## Modelica.Blocks.Math.IntegerToBoolean

Convert Integer to Boolean signal



## Information

This block computes the Boolean output **y** from the Integer input **u** by the equation:

$$y = u \geq \text{threshold};$$

where **threshold** is a parameter.

## Parameters

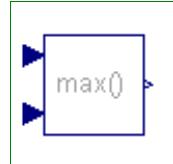
Name	Default	Description
threshold	1	Output signal y is true, if input $u \geq \text{threshold}$

## Connectors

Name	Description
u	Connector of Integer input signal
y	Connector of Boolean output signal

## Modelica.Blocks.Math.Max

Pass through the largest signal



## Information

This block computes the output **y** as *maximum* of the two Real inputs **u1** and **u2**:

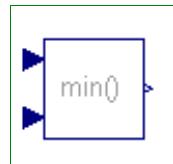
$$y = \max(u1, u2);$$

## Connectors

Name	Description
u1	Connector of Real input signal 1
u2	Connector of Real input signal 2
y	Connector of Real output signal

## Modelica.Blocks.Math.Min

Pass through the smallest signal



## Information

This block computes the output **y** as *minimum* of the two Real inputs **u1** and **u2**:

```
y = min ( u1 , u2 );
```

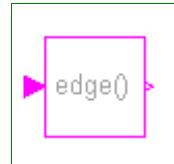
## Connectors

Name	Description
u1	Connector of Real input signal 1
u2	Connector of Real input signal 2
y	Connector of Real output signal

---

## Modelica.Blocks.Math.Edge

Indicates rising edge of boolean signal



## Information

This block sets the Boolean output **y** to true, when the Boolean input **u** shows a *rising edge*:

```
y = edge ( u );
```

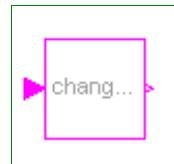
## Connectors

Name	Description
u	Connector of Boolean input signal
y	Connector of Boolean output signal

---

## Modelica.Blocks.Math.BooleanChange

Indicates boolean signal changing



## Information

This block sets the Boolean output **y** to true, when the Boolean input **u** shows a *rising or falling edge*, i.e., when the signal changes:

```
y = change ( u );
```

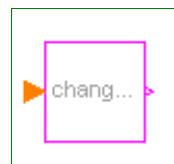
## Connectors

Name	Description
u	Connector of Boolean input signal
y	Connector of Boolean output signal

---

## Modelica.Blocks.Math.IntegerChange

Indicates integer signal changing



## 156 Modelica.Blocks.Math.IntegerChange

---

### Information

This block sets the Boolean output **y** to true, when the Integer input **u** changes:

```
y = change( u );
```

### Connectors

Name	Description
u	Connector of Integer input signal
y	Connector of Boolean output signal

---

## Modelica.Blocks.Nonlinear

### Library of discontinuous or non-differentiable algebraic control blocks

### Information

This package contains **discontinuous** and **non-differentiable, algebraic** input/output blocks.

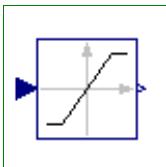
### Package Content

Name	Description
Limiter	Limit the range of a signal
VariableLimiter	Limit the range of a signal with variable limits
DeadZone	Provide a region of zero output
FixedDelay	Delay block with fixed DelayTime
PadeDelay	Pade approximation of delay block with fixed DelayTime
VariableDelay	Delay block with variable DelayTime

---

## Modelica.Blocks.Nonlinear.Limiter

### Limit the range of a signal



### Information

The Limiter block passes its input signal as output signal as long as the input is within the specified upper and lower limits. If this is not the case, the corresponding limits are passed as output.

### Parameters

Name	Default	Description
uMax	1	Upper limits of input signals
uMin	-uMax	Lower limits of input signals
limitsAtInit	true	= false, if limits are ignored during initialization (i.e., y=u)

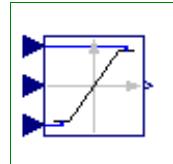
### Connectors

Name	Description

u	Connector of Real input signal
y	Connector of Real output signal

## Modelica.Blocks.Nonlinear.VariableLimiter

Limit the range of a signal with variable limits



### Information

The Limiter block passes its input signal as output signal as long as the input is within the upper and lower limits specified by the two additional inputs limit1 and limit2. If this is not the case, the corresponding limit is passed as output.

### Parameters

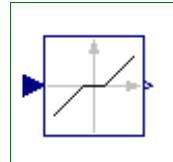
Name	Default	Description
limitsAtInit	true	= false, if limits are ignored during initialization (i.e., y=u)

### Connectors

Name	Description
u	Connector of Real input signal
y	Connector of Real output signal
limit1	Connector of Real input signal used as maximum of input u
limit2	Connector of Real input signal used as minimum of input u

## Modelica.Blocks.Nonlinear.DeadZone

Provide a region of zero output



### Information

The DeadZone block defines a region of zero output.

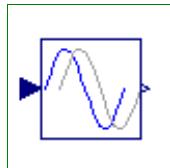
If the input is within  $u_{\text{Min}} \dots u_{\text{Max}}$ , the output is zero. Outside of this zone, the output is a linear function of the input with a slope of 1.

### Parameters

Name	Default	Description
uMax	1	Upper limits of dead zones
uMin	-uMax	Lower limits of dead zones
deadZoneAtInit	true	= false, if dead zone is ignored during initialization (i.e., y=u)

### Connectors

Name	Description
u	Connector of Real input signal
y	Connector of Real output signal

**Modelica.Blocks.Nonlinear.FixedDelay****Delay block with fixed DelayTime****Information**

The Input signal is delayed by a given time instant, or more precisely:

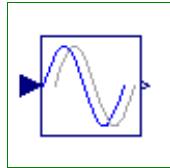
$$\begin{aligned} y &= u(\text{time} - \text{delayTime}) \quad \text{for } \text{time} > \text{time.start} + \text{delayTime} \\ &= u(\text{time.start}) \quad \text{for } \text{time} \leq \text{time.start} + \text{delayTime} \end{aligned}$$

**Parameters**

Name	Default	Description
delayTime	1	Delay time of output with respect to input signal [s]

**Connectors**

Name	Description
u	Connector of Real input signal
y	Connector of Real output signal

**Modelica.Blocks.Nonlinear.PadeDelay****Pade approximation of delay block with fixed DelayTime****Information**

The Input signal is delayed by a given time instant, or more precisely:

$$\begin{aligned} y &= u(\text{time} - \text{delayTime}) \quad \text{for } \text{time} > \text{time.start} + \text{delayTime} \\ &= u(\text{time.start}) \quad \text{for } \text{time} \leq \text{time.start} + \text{delayTime} \end{aligned}$$

The delay is approximated by a Pade approximation, i.e., by a transfer function

$$y(s) = \frac{b[1]*s^m + b[2]*s^{m-1} + \dots + b[m+1]}{a[1]*s^n + a[2]*s^{n-1} + \dots + a[n+1]} * u(s)$$

where the coefficients  $b[:]$  and  $a[:]$  are calculated such that the coefficients of the Taylor expansion of the delay  $\exp(-T*s)$  around  $s=0$  are identical upto order  $n+m$ .

The main advantage of this approach is that the delay is approximated by a linear differential equation system, which is continuous and continuously differentiable. For example, it is uncritical to linearize a system containing a Pade-approximated delay.

The standard text book version uses order "m=n", which is also the default setting of this block. The setting "m=n-1" may yield a better approximation in certain cases.

Literature:

Otto Foellinger: Regelungstechnik, 8. Auflage, chapter 11.9, page 412-414, Huethig Verlag Heidelberg, 1994

**Parameters**

Name	Default	Description
delayTime	1	Delay time of output with respect to input signal [s]

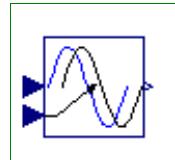
n	1	Order of pade approximation
m	n	Order of numerator

## Connectors

Name	Description
u	Connector of Real input signal
y	Connector of Real output signal

## Modelica.Blocks.Nonlinear.VariableDelay

Delay block with variable DelayTime



## Information

The Input signal is delayed by a given time instant, or more precisely:

$$\begin{aligned} y &= u(\text{time} - \text{delayTime}) \quad \text{for } \text{time} > \text{time.start} + \text{delayTime} \\ &= u(\text{time.start}) \quad \text{for } \text{time} \leq \text{time.start} + \text{delayTime} \end{aligned}$$

where delayTime is an additional input signal which must follow the following relationship:

$$0 \leq \text{delayTime} \leq \text{delayMax}$$

## Parameters

Name	Default	Description
delayMax	1	maximum delay time

## Connectors

Name	Description
u	Connector of Real input signal
y	Connector of Real output signal
delayTime	

## Modelica.Blocks.Routing

Library of blocks to combine and extract signals

## Information

This package contains blocks to combine and extract signals.

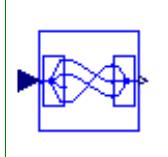
## Package Content

Name	Description
ExtractSignal	Extract signals from an input signal vector
Extractor	Extract scalar signal out of signal vector dependent on IntegerRealInput index
Multiplex2	Multiplexer block for two input connectors

 Multiplex3	Multiplexer block for three input connectors
 Multiplex4	Multiplexer block for four input connectors
 Multiplex5	Multiplexer block for five input connectors
 Multiplex6	Multiplexer block for six input connectors
 DeMultiplex2	DeMultiplexer block for two output connectors
 DeMultiplex3	DeMultiplexer block for three output connectors
 DeMultiplex4	DeMultiplexer block for four output connectors
 DeMultiplex5	DeMultiplexer block for five output connectors
 DeMultiplex6	DeMultiplexer block for six output connectors
 RealPassThrough	Pass a Real signal through without modification
 IntegerPassThrough	Pass a Integer signal through without modification
 BooleanPassThrough	Pass a Boolean signal through without modification

## Modelica.Blocks.Routing.ExtractSignal

Extract signals from an input signal vector



### Information

Extract signals from the input connector and transfer them to the output connector.

The extracting scheme is given by the integer vector 'extract'. This vector specifies, which input signals are taken and in which order they are transferred to the output vector. Note, that the dimension of 'extract' has to match the number of outputs. Additionally, the dimensions of the input connector signals and the output connector signals have to be explicitly defined via the parameters 'nin' and 'nout'.

Example:

```
nin = 7 "Number of inputs";
nout = 4 "Number of outputs";
extract[nout] = {6,3,3,2} "Extracting vector";
```

extracts four output signals (nout=4) from the seven elements of the input vector (nin=7):

```
output no. 1 is set equal to input no. 6
output no. 2 is set equal to input no. 3
output no. 3 is set equal to input no. 3
output no. 4 is set equal to input no. 2
```

### Parameters

Name	Default	Description
nin	1	Number of inputs
nout	1	Number of outputs
extract[nout]	1:nout	Extracting vector

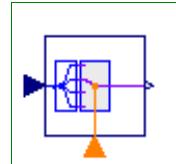
### Connectors

Name	Description

u[nin]	Connector of Real input signals
y[nout]	Connector of Real output signals

## Modelica.Blocks.Routing.Extractor

Extract scalar signal out of signal vector dependent on IntegerRealInput index



### Information

This block extracts a scalar output signal out the vector of input signals dependent on the Integer value of the additional u index:

```
y = u [ index ] ;
```

where index is an additional Integer input signal.

### Parameters

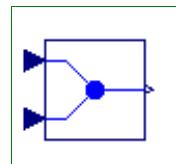
Name	Default	Description
nin	1	Number of inputs
allowOutOfRange	false	Index may be out of range
outOfRangeValue	1e10	Output signal if index is out of range

### Connectors

Name	Description
u[nin]	Connector of Real input signals
y	Connector of Real output signal
index	

## Modelica.Blocks.Routing.Multiplex2

Multiplexer block for two input connectors



### Information

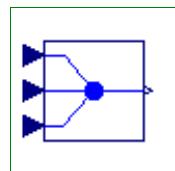
The output connector is the **concatenation** of the two input connectors. Note, that the dimensions of the input connector signals have to be explicitly defined via parameters n1 and n2.

### Parameters

Name	Default	Description
n1	1	dimension of input signal connector 1
n2	1	dimension of input signal connector 2

### Connectors

Name	Description
u1[n1]	Connector of Real input signals 1
u2[n2]	Connector of Real input signals 2
y[n1 + n2]	Connector of Real output signals

**Modelica.Blocks.Routing.Multiplex3****Multiplexer block for three input connectors****Information**

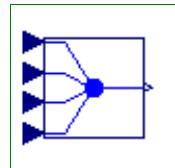
The output connector is the **concatenation** of the three input connectors. Note, that the dimensions of the input connector signals have to be explicitly defined via parameters n1, n2 and n3.

**Parameters**

Name	Default	Description
n1	1	dimension of input signal connector 1
n2	1	dimension of input signal connector 2
n3	1	dimension of input signal connector 3

**Connectors**

Name	Description
u1[n1]	Connector of Real input signals 1
u2[n2]	Connector of Real input signals 2
u3[n3]	Connector of Real input signals 3
y[n1 + n2 + n3]	Connector of Real output signals

**Modelica.Blocks.Routing.Multiplex4****Multiplexer block for four input connectors****Information**

The output connector is the **concatenation** of the four input connectors. Note, that the dimensions of the input connector signals have to be explicitly defined via parameters n1, n2, n3 and n4.

**Parameters**

Name	Default	Description
n1	1	dimension of input signal connector 1
n2	1	dimension of input signal connector 2
n3	1	dimension of input signal connector 3
n4	1	dimension of input signal connector 4

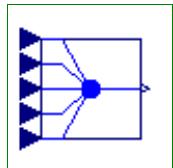
**Connectors**

Name	Description
u1[n1]	Connector of Real input signals 1
u2[n2]	Connector of Real input signals 2
u3[n3]	Connector of Real input signals 3
u4[n4]	Connector of Real input signals 4
y[n1 + n2 + n3 + n4]	Connector of Real output signals

## Modelica.Blocks.Routing.Multiplex5

Multiplexer block for five input connectors

### Information



The output connector is the **concatenation** of the five input connectors. Note, that the dimensions of the input connector signals have to be explicitly defined via parameters n1, n2, n3, n4 and n5.

### Parameters

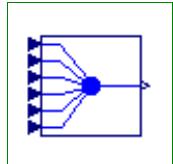
Name	Default	Description
n1	1	dimension of input signal connector 1
n2	1	dimension of input signal connector 2
n3	1	dimension of input signal connector 3
n4	1	dimension of input signal connector 4
n5	1	dimension of input signal connector 5

### Connectors

Name	Description
u1[n1]	Connector of Real input signals 1
u2[n2]	Connector of Real input signals 2
u3[n3]	Connector of Real input signals 3
u4[n4]	Connector of Real input signals 4
u5[n5]	Connector of Real input signals 5
y[n1 + n2 + n3 + n4 + n5]	Connector of Real output signals

## Modelica.Blocks.Routing.Multiplex6

Multiplexer block for six input connectors



### Information

The output connector is the **concatenation** of the six input connectors. Note, that the dimensions of the input connector signals have to be explicitly defined via parameters n1, n2, n3, n4, n5 and n6.

### Parameters

Name	Default	Description
n1	1	dimension of input signal connector 1
n2	1	dimension of input signal connector 2
n3	1	dimension of input signal connector 3
n4	1	dimension of input signal connector 4
n5	1	dimension of input signal connector 5
n6	1	dimension of input signal connector 6

### Connectors

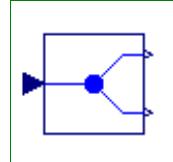
Name	Description
u1[n1]	Connector of Real input signals 1

## 164 Modelica.Blocks.Routing.Multiplex6

u2[n2]	Connector of Real input signals 2
u3[n3]	Connector of Real input signals 3
u4[n4]	Connector of Real input signals 4
u5[n5]	Connector of Real input signals 5
u6[n6]	Connector of Real input signals 6
y[n1 + n2 + n3 + n4 + n5 + n6]	Connector of Real output signals

### Modelica.Blocks.Routing.DeMultiplex2

DeMultiplexer block for two output connectors



#### Information

The input connector is **splitted** up into two output connectors. Note, that the dimensions of the output connector signals have to be explicitly defined via parameters n1 and n2.

#### Parameters

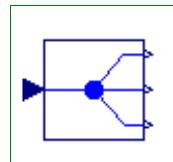
Name	Default	Description
n1	1	dimension of output signal connector 1
n2	1	dimension of output signal connector 2

#### Connectors

Name	Description
u[n1 + n2]	Connector of Real input signals
y1[n1]	Connector of Real output signals 1
y2[n2]	Connector of Real output signals 2

### Modelica.Blocks.Routing.DeMultiplex3

DeMultiplexer block for three output connectors



#### Information

The input connector is **splitted** up into three output connectors. Note, that the dimensions of the output connector signals have to be explicitly defined via parameters n1, n2 and n3.

#### Parameters

Name	Default	Description
n1	1	dimension of output signal connector 1
n2	1	dimension of output signal connector 2
n3	1	dimension of output signal connector 3

#### Connectors

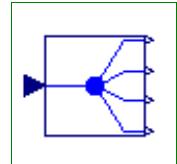
Name	Description
u[n1 + n2 + n3]	Connector of Real input signals
y1[n1]	Connector of Real output signals 1

---

y2[n2]	Connector of Real output signals 2
y3[n3]	Connector of Real output signals 3

## Modelica.Blocks.Routing.DeMultiplex4

DeMultiplexer block for four output connectors



### Information

The input connector is **splitted** up into four output connectors. Note, that the dimensions of the output connector signals have to be explicitly defined via parameters n1, n2, n3 and n4.

### Parameters

Name	Default	Description
n1	1	dimension of output signal connector 1
n2	1	dimension of output signal connector 2
n3	1	dimension of output signal connector 3
n4	1	dimension of output signal connector 4

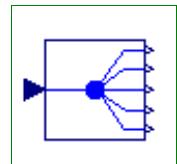
### Connectors

Name	Description
u[n1 + n2 + n3 + n4]	Connector of Real input signals
y1[n1]	Connector of Real output signals 1
y2[n2]	Connector of Real output signals 2
y3[n3]	Connector of Real output signals 3
y4[n4]	Connector of Real output signals 4

---

## Modelica.Blocks.Routing.DeMultiplex5

DeMultiplexer block for five output connectors



### Information

The input connector is **splitted** up into five output connectors. Note, that the dimensions of the output connector signals have to be explicitly defined via parameters n1, n2, n3, n4 and n5.

### Parameters

Name	Default	Description
n1	1	dimension of output signal connector 1
n2	1	dimension of output signal connector 2
n3	1	dimension of output signal connector 3
n4	1	dimension of output signal connector 4
n5	1	dimension of output signal connector 5

### Connectors

Name	Description

## 166 Modelica.Blocks.Routing.DeMultiplex5

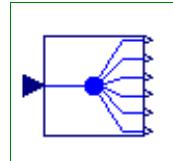
---

u[n1 + n2 + n3 + n4 + n5]	Connector of Real input signals
y1[n1]	Connector of Real output signals 1
y2[n2]	Connector of Real output signals 2
y3[n3]	Connector of Real output signals 3
y4[n4]	Connector of Real output signals 4
y5[n5]	Connector of Real output signals 5

---

## Modelica.Blocks.Routing.DeMultiplex6

DeMultiplexer block for six output connectors



### Information

The input connector is **splitted** up into six output connectors. Note, that the dimensions of the output connector signals have to be explicitly defined via parameters n1, n2, n3, n4, n5 and n6.

### Parameters

Name	Default	Description
n1	1	dimension of output signal connector 1
n2	1	dimension of output signal connector 2
n3	1	dimension of output signal connector 3
n4	1	dimension of output signal connector 4
n5	1	dimension of output signal connector 5
n6	1	dimension of output signal connector 6

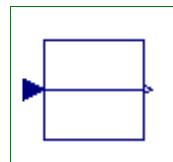
### Connectors

Name	Description
u[n1 + n2 + n3 + n4 + n5 + n6]	Connector of Real input signals
y1[n1]	Connector of Real output signals 1
y2[n2]	Connector of Real output signals 2
y3[n3]	Connector of Real output signals 3
y4[n4]	Connector of Real output signals 4
y5[n5]	Connector of Real output signals 5
y6[n6]	Connector of Real output signals 6

---

## Modelica.Blocks.Routing.RealPassThrough

Pass a Real signal through without modification



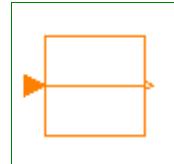
### Information

Passes a Real signal through without modification. Enables signals to be read out of one bus, have their name changed and be sent back to a bus.

### Connectors

Name	Description

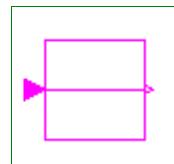
u	Input signal
y	Output signal

**Modelica.Blocks.Routing.IntegerPassThrough****Pass a Integer signal through without modification****Information**

Passes a Integer signal through without modification. Enables signals to be read out of one bus, have their name changed and be sent back to a bus.

**Connectors**

Name	Description
u	Input signal
y	Output signal

**Modelica.Blocks.Routing.BooleanPassThrough****Pass a Boolean signal through without modification****Information**

Passes a Boolean signal through without modification. Enables signals to be read out of one bus, have their name changed and be sent back to a bus.

**Connectors**

Name	Description
u	Input signal
y	Output signal

**Modelica.Blocks.Sources****Library of signal source blocks generating Real and Boolean signals****Information**

This package contains **source** components, i.e., blocks which have only output signals. These blocks are used as signal generators for Real, Integer and Boolean signals.

All Real source signals (with the exception of the Constant source) have at least the following two parameters:

<b>offset</b>	Value which is added to the signal
<b>startTime</b>	Start time of signal. For time < startTime, the output y is set to offset.

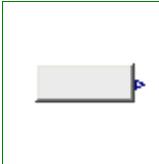
The **offset** parameter is especially useful in order to shift the corresponding source, such that at initial time the system is stationary. To determine the corresponding value of offset, usually requires a trimming calculation.

## Package Content

Name	Description
 RealExpression	Set output signal to a time varying Real expression
 IntegerExpression	Set output signal to a time varying Integer expression
 BooleanExpression	Set output signal to a time varying Boolean expression
 Clock	Generate actual time signal
 Constant	Generate constant signal of type Real
 Step	Generate step signal of type Real
 Ramp	Generate ramp signal
 Sine	Generate sine signal
 ExpSine	Generate exponentially damped sine signal
 Exponentials	Generate a rising and falling exponential signal
 Pulse	Generate pulse signal of type Real
 SawTooth	Generate saw tooth signal
 Trapezoid	Generate trapezoidal signal of type Real
 KinematicPTP	Move as fast as possible along a distance within given kinematic constraints
 KinematicPTP2	Move as fast as possible from start to end position within given kinematic constraints with output signals q, qd=der(q), qdd=der(qd)
 TimeTable	Generate a (possibly discontinuous) signal by linear interpolation in a table
 CombiTimeTable	Table look-up with respect to time and linear/periodic extrapolation methods (data from matrix/file)
 BooleanConstant	Generate constant signal of type Boolean
 BooleanStep	Generate step signal of type Boolean
 BooleanPulse	Generate pulse signal of type Boolean
 SampleTrigger	Generate sample trigger signal
 BooleanTable	Generate a Boolean output signal based on a vector of time instants
 IntegerConstant	Generate constant signal of type Integer
 IntegerStep	Generate step signal of type Integer

### Modelica.Blocks.Sources.RealExpression

Set output signal to a time varying Real expression



#### Information

The (time varying) Real output signal of this block can be defined in its parameter menu via variable **y**. The purpose is to support the easy definition of Real expressions in a block diagram. For example, in the y-menu the definition "if time < 1 then 0 else 1" can be given in order to define that the output signal is one, if time  $\geq 1$  and otherwise it is zero. Note, that "time" is a built-in variable that is always accessible and represents the "model time" and that Variable **y** is both a variable and a connector.

## Parameters

Name	Default	Description
Time varying output signal		
y	0.0	Value of Real output

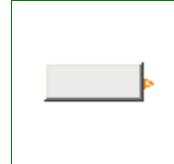
## Connectors

Name	Description
Time varying output signal	
y	Value of Real output

---

## Modelica.Blocks.Sources.IntegerExpression

Set output signal to a time varying Integer expression



## Information

The (time varying) Integer output signal of this block can be defined in its parameter menu via variable **y**. The purpose is to support the easy definition of Integer expressions in a block diagram. For example, in the y-menu the definition "if time < 1 then 0 else 1" can be given in order to define that the output signal is one, if time  $\geq 1$  and otherwise it is zero. Note, that "time" is a built-in variable that is always accessible and represents the "model time" and that Variable **y** is both a variable and a connector.

## Parameters

Name	Default	Description
Time varying output signal		
y	0	Value of Integer output

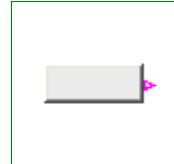
## Connectors

Name	Description
Time varying output signal	
y	Value of Integer output

---

## Modelica.Blocks.Sources.BooleanExpression

Set output signal to a time varying Boolean expression



## Information

The (time varying) Boolean output signal of this block can be defined in its parameter menu via variable **y**. The purpose is to support the easy definition of Boolean expressions in a block diagram. For example, in the y-menu the definition "time  $\geq 1$  and time  $\leq 2$ " can be given in order to define that the output signal is **true** in the time interval  $1 \leq \text{time} \leq 2$  and otherwise it is **false**. Note, that "time" is a built-in variable that is always accessible and represents the "model time" and that Variable **y** is both a variable and a connector.

## Parameters

Name	Default	Description
Time varying output signal		

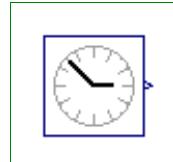
y	false	Value of Boolean output
---	-------	-------------------------

### Connectors

Name	Description
Time varying output signal	
y	Value of Boolean output

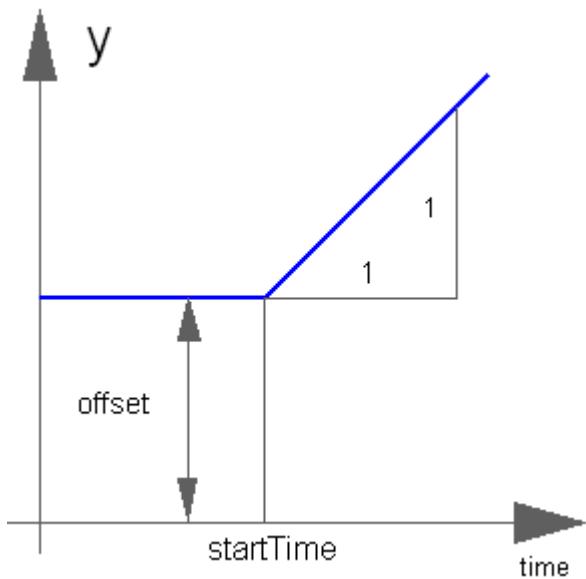
### Modelica.Blocks.Sources.Clock

Generate actual time signal



### Information

The Real output y is a clock signal:



### Parameters

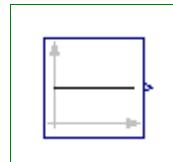
Name	Default	Description
offset	0	Offset of output signal
startTime	0	Output = offset for time < startTime [s]

### Connectors

Name	Description
y	Connector of Real output signal

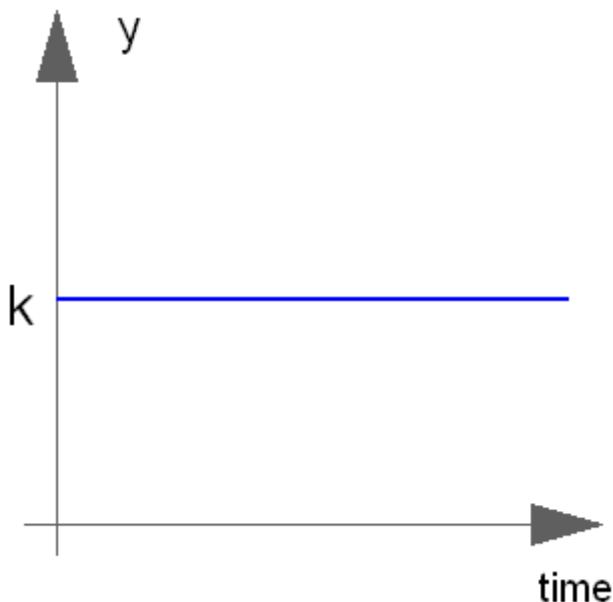
### Modelica.Blocks.Sources.Constant

Generate constant signal of type Real



### Information

The Real output y is a constant signal:



### Parameters

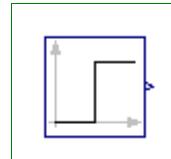
Name	Default	Description
$k$	1	Constant output value

### Connectors

Name	Description
$y$	Connector of Real output signal

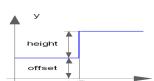
### Modelica.Blocks.Sources.Step

Generate step signal of type Real



### Information

The Real output  $y$  is a step signal:



### Parameters

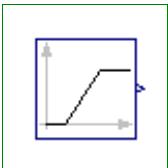
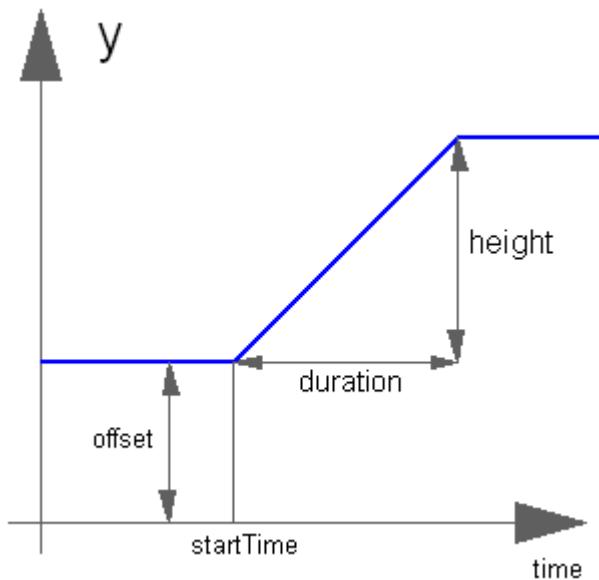
Name	Default	Description
height	1	Height of step
offset	0	Offset of output signal $y$
startTime	0	Output $y = \text{offset}$ for time < startTime [s]

### Connectors

Name	Description
$y$	Connector of Real output signal

**Modelica.Blocks.Sources.Ramp**

Generate ramp signal

**Information**The Real output  $y$  is a ramp signal:**Parameters**

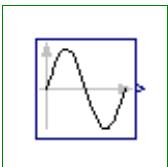
Name	Default	Description
height	1	Height of ramps
duration	2	Durations of ramp
offset	0	Offset of output signal
startTime	0	Output = offset for time < startTime [s]

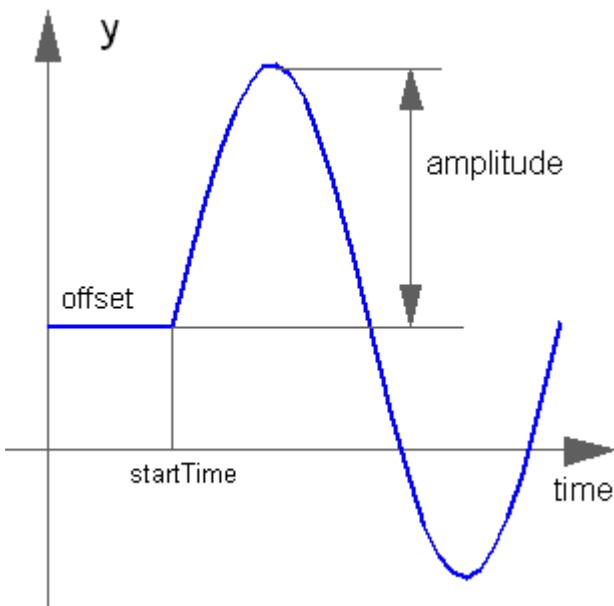
**Connectors**

Name	Description
y	Connector of Real output signal

**Modelica.Blocks.Sources.Sine**

Generate sine signal

**Information**The Real output  $y$  is a sine signal:



## Parameters

Name	Default	Description
amplitude	1	Amplitude of sine wave
freqHz	1	Frequency of sine wave [Hz]
phase	0	Phase of sine wave [rad]
offset	0	Offset of output signal
startTime	0	Output = offset for time < startTime [s]

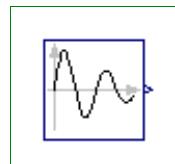
## Connectors

Name	Description
y	Connector of Real output signal

---

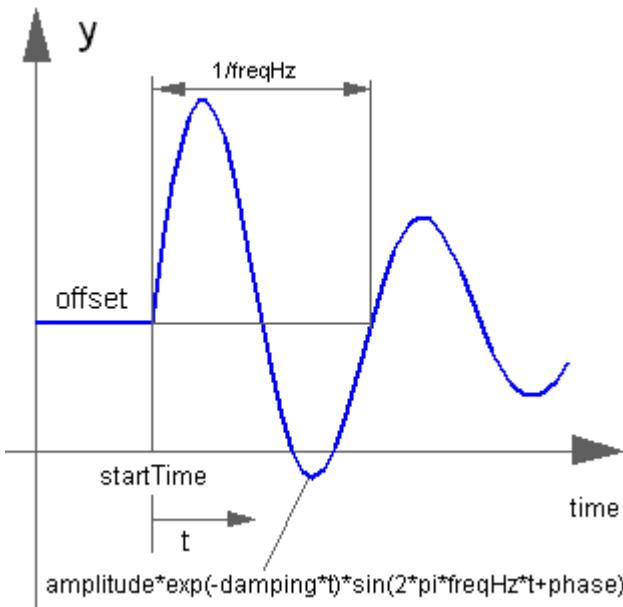
## Modelica.Blocks.Sources.ExpSine

Generate exponentially damped sine signal



## Information

The Real output  $y$  is a sine signal with exponentially changing amplitude:



## Parameters

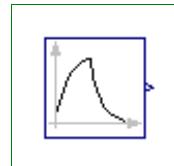
Name	Default	Description
amplitude	1	Amplitude of sine wave
freqHz	2	Frequency of sine wave [Hz]
phase	0	Phase of sine wave [rad]
damping	1	Damping coefficient of sine wave [s-1]
offset	0	Offset of output signal
startTime	0	Output = offset for time < startTime [s]

## Connectors

Name	Description
y	Connector of Real output signal

## Modelica.Blocks.Sources.Exponentials

Generate a rising and falling exponential signal



## Information

The Real output  $y$  is a rising exponential followed by a falling exponential signal:



## Parameters

Name	Default	Description
outMax	1	Height of output for infinite riseTime
riseTime	0.5	Rise time [s]
riseTimeConst	0.1	Rise time constant [s]

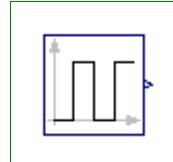
fallTimeConst	riseTimeConst	Fall time constant [s]
offset	0	Offset of output signal
startTime	0	Output = offset for time < startTime [s]

## Connectors

Name	Description
y	Connector of Real output signal

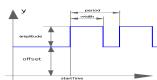
## Modelica.Blocks.Sources.Pulse

Generate pulse signal of type Real



## Information

The Real output y is a pulse signal:



## Parameters

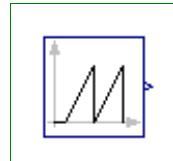
Name	Default	Description
amplitude	1	Amplitude of pulse
width	50	Width of pulse in % of periods
period	1	Time for one period [s]
offset	0	Offset of output signals
startTime	0	Output = offset for time < startTime [s]

## Connectors

Name	Description
y	Connector of Real output signal

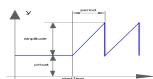
## Modelica.Blocks.Sources.SawTooth

Generate saw tooth signal



## Information

The Real output y is a saw tooth signal:



## Parameters

Name	Default	Description
amplitude	1	Amplitude of saw tooth
period	1	Time for one period [s]
offset	0	Offset of output signals

## 176 Modelica.Blocks.Sources.SawTooth

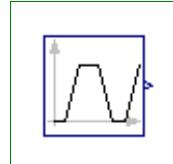
startTime   0	Output = offset for time < startTime [s]
---------------	--

### Connectors

Name	Description
y	Connector of Real output signal

## Modelica.Blocks.Sources.Trapezoid

Generate trapezoidal signal of type Real



### Information

The Real output y is a trapezoid signal:



### Parameters

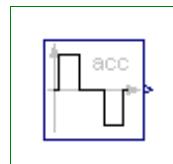
Name	Default	Description
amplitude	1	Amplitude of trapezoid
rising	0	Rising duration of trapezoid [s]
width	0.5	Width duration of trapezoid [s]
falling	0	Falling duration of trapezoid [s]
period	1	Time for one period [s]
nperiod	-1	Number of periods (< 0 means infinite number of periods)
offset	0	Offset of output signal
startTime	0	Output = offset for time < startTime [s]

### Connectors

Name	Description
y	Connector of Real output signal

## Modelica.Blocks.Sources.KinematicPTP

Move as fast as possible along a distance within given kinematic constraints



### Information

The goal is to move as **fast** as possible along a distance **deltaq** under given **kinematical constraints**. The distance can be a positional or angular range. In robotics such a movement is called **PTP** (Point-To-Point). This source block generates the **acceleration** qdd of this signal as output:



After integrating the output two times, the position q is obtained. The signal is constructed in such a way that it is not possible to move faster, given the **maximally allowed velocity** qd\_max and the **maximally allowed acceleration** qdd\_max.

If several distances are given (vector deltaq has more than 1 element), an acceleration output vector is

constructed such that all signals are in the same periods in the acceleration, constant velocity and deceleration phase. This means that only one of the signals is at its limits whereas the others are synchronized in such a way that the end point is reached at the same time instant.

This element is useful to generate a reference signal for a controller which controls a drive train or in combination with model Modelica.Mechanics.Rotational.**Accelerate** to drive a flange according to a given acceleration.

## Parameters

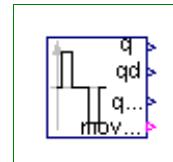
Name	Default	Description
deltaq[:]	{1}	Distance to move
qd_max[:]	{1}	Maximum velocities der(q)
qdd_max[:]	{1}	Maximum accelerations der(qd)
startTime	0	Time instant at which movement starts [s]

## Connectors

Name	Description
y[nout]	Connector of Real output signals

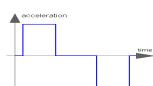
## Modelica.Blocks.Sources.KinematicPTP2

Move as fast as possible from start to end position within given kinematic constraints with output signals q, qd=der(q), qdd=der(qd)



## Information

The goal is to move as **fast** as possible from start position **q\_begin** to end position **q\_end** under given **kinematical constraints**. The positions can be translational or rotational definitions (i.e.,  $q_{begin}/q_{end}$  is given). In robotics such a movement is called **PTP** (Point-To-Point). This source block generates the **position**  $q(t)$ , the **speed**  $qd(t) = \text{der}(q)$ , and the **acceleration**  $qdd(t) = \text{der}(qd)$  as output. The signals are constructed in such a way that it is not possible to move faster, given the **maximally allowed velocity**  $qd_{max}$  and the **maximally allowed acceleration**  $qdd_{max}$ :



If vectors  $q_{begin}/q_{end}$  have more than 1 element, the output vectors are constructed such that all signals are in the same periods in the acceleration, constant velocity and deceleration phase. This means that only one of the signals is at its limits whereas the others are synchronized in such a way that the end point is reached at the same time instant.

This element is useful to generate a reference signal for a controller which controls, e.g., a drive train, or to drive a flange according to a given acceleration.

## Parameters

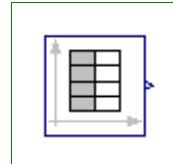
Name	Default	Description
q_begin[:]	{0}	Start position
q_end[:]	{1}	End position
qd_max[:]	{1}	Maximum velocities der(q)
qdd_max[:]	{1}	Maximum accelerations der(qd)
startTime	0	Time instant at which movement starts [s]

## Connectors

Name	Description
q[nout]	Reference position of path planning
qd[nout]	Reference speed of path planning
qdd[nout]	Reference acceleration of path planning
moving[nout]	= true, if end position not yet reached; = false, if end position reached or axis is completely at rest

## Modelica.Blocks.Sources.TimeTable

Generate a (possibly discontinuous) signal by linear interpolation in a table



## Information

This block generates an output signal by **linear interpolation** in a table. The time points and function values are stored in a matrix **table[i,j]**, where the first column **table[:,1]** contains the time points and the second column contains the data to be interpolated. The table interpolation has the following properties:

- The time points need to be **monotonically increasing**.
- Discontinuities** are allowed, by providing the same time point twice in the table.
- Values **outside** of the table range, are computed by **extrapolation** through the last or first two points of the table.
- If the table has only **one row**, no interpolation is performed and the function value is just returned independantly of the actual time instant.
- Via parameters **startTime** and **offset** the curve defined by the table can be shifted both in time and in the ordinate value.
- The table is implemented in a numerically sound way by generating **time events** at interval boundaries, in order to not integrate over a discontinuous or not differentiable points.

Example:

```
table = [0  0
         1  0
         1  1
         2  4
         3  9
         4 16]
```

If, e.g., time = 1.0, the output y = 0.0 (before event), 1.0 (after event)  
e.g., time = 1.5, the output y = 2.5,  
e.g., time = 2.0, the output y = 4.0,  
e.g., time = 5.0, the output y = 23.0 (i.e. extrapolation).



## Parameters

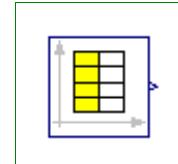
Name	Default	Description
table[:, 2]	[0, 0; 1, 1; 2, 4]	Table matrix (time = first column)
offset	0	Offset of output signal
startTime	0	Output = offset for time < startTime [s]

## Connectors

Name	Description
y	Connector of Real output signal

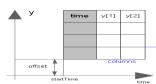
## Modelica.Blocks.Sources.CombiTimeTable

Table look-up with respect to time and linear/periodic extrapolation methods (data from matrix/file)



## Information

This block generates an output signal  $y[:]$  by **linear interpolation** in a table. The time points and function values are stored in a matrix  $\text{table}[i,j]$ , where the first column  $\text{table}[:,1]$  contains the time points and the other columns contain the data to be interpolated.



Via parameter **columns** it can be defined which columns of the table are interpolated. If, e.g.,  $\text{columns}=\{2,4\}$ , it is assumed that 2 output signals are present and that the first output is computed by interpolation of column 2 and the second output is computed by interpolation of column 4 of the table matrix. The table interpolation has the following properties:

- The time points need to be **monotonically increasing**.
- **Discontinuities** are allowed, by providing the same time point twice in the table.
- Values **outside** of the table range, are computed by extrapolation according to the setting of parameter **extrapolation**:
 

```
extrapolation = 0: hold the first or last value of the table,
                     if outside of the range.
                     = 1: extrapolate through the last or first two
                           points of the table.
                     = 2: periodically repeat the table data
                           (periodical function).
```
- Via parameter **smoothness** it is defined how the data is interpolated:
 

```
smoothness = 0: linear interpolation
                     = 1: smooth interpolation with Akima Splines such
                           that  $\text{der}(y)$  is continuous.
```
- If the table has only **one row**, no interpolation is performed and the table values of this row are just returned.
- Via parameters **startTime** and **offset** the curve defined by the table can be shifted both in time and in the ordinate value. The time instants stored in the table are therefore **relative** to **startTime**. If  $\text{time} < \text{startTime}$ , no interpolation is performed and the offset is used as ordinate value for all outputs.
- The table is implemented in a numerically sound way by generating **time events** at interval boundaries, in order to not integrate over a discontinuous or not differentiable points.
- For special applications it is sometimes needed to know the minimum and maximum time instant defined in the table as a parameter. For this reason parameters **t\_min** and **t\_max** are provided and can be access from the outside of the table object.

Example:

```
table = [0  0
         1  0
         1  1
         2  4
```

```

3   9
4 16]; extrapolation = 1 (default)
If, e.g., time = 1.0, the output y = 0.0 (before event), 1.0 (after event)
e.g., time = 1.5, the output y = 2.5,
e.g., time = 2.0, the output y = 4.0,
e.g., time = 5.0, the output y = 23.0 (i.e. extrapolation via last 2
points).

```

The table matrix can be defined in the following ways:

1. Explicitly supplied as **parameter matrix** "table", and the other parameters have the following values:

```

tableName is "NoName" or has only blanks,
fileName is "NoName" or has only blanks.

```

2. **Read from a file** "fileName" where the matrix is stored as "tableName". Both ASCII and binary file format is possible. (the ASCII format is described below). It is most convenient to generate the binary file from Matlab (Matlab 4 storage format), e.g., by command

```
save tables.mat tab1 tab2 tab3 -v4
```

when the three tables tab1, tab2, tab3 should be used from the model.

3. Statically stored in function "usertab" in file "usertab.c". The matrix is identified by "tableName". Parameter fileName = "NoName" or has only blanks.

Table definition methods (1) and (3) do **not** allocate dynamic memory, and do not access files, whereas method (2) does. Therefore (1) and (3) are suited for hardware-in-the-loop simulation (e.g. with dSpace hardware). When the constant "NO\_FILE" is defined in "usertab.c", all parts of the source code of method (2) are removed by the C-preprocessor, such that no dynamic memory allocation and no access to files takes place.

If tables are read from an ASCII-file, the file need to have the following structure ("----" is not part of the file content):

```
-----
#1
double tab1(6,2)    # comment line
0   0
1   0
1   1
2   4
3   9
4  16
double tab2(6,2)    # another comment line
0   0
2   0
2   2
4   8
6  18
8  32
-----
```

Note, that the first two characters in the file need to be "#1". Afterwards, the corresponding matrix has to be declared with type, name and actual dimensions. Finally, in successive rows of the file, the elements of the matrix have to be given. Several matrices may be defined one after another.

## Parameters

Name	Default	Description
table data definition		

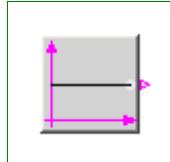
tableOnFile	false	= true, if table is defined on file or in function usertab
table[:, :]	fill(0.0, 0, 2)	Table matrix (time = first column)
tableName	"NoName"	Table name on file or in function usertab (see docu)
fileName	"NoName"	File where matrix is stored
<b>table data interpretation</b>		
columns[:]	2:size(table, 2)	Columns of table to be interpolated
smoothness	Blocks.Types.Smoothness.Line...	Smoothness of table interpolation
extrapolation	Blocks.Types.Extrapolation.L...	Extrapolation of data outside the definition range
offset[:]	{0}	Offsets of output signals
startTime	0	Output = offset for time < startTime [s]

## Connectors

Name	Description
y[nout]	Connector of Real output signals

## Modelica.Blocks.Sources.BooleanConstant

Generate constant signal of type Boolean



## Information

The Boolean output y is a constant signal:



## Parameters

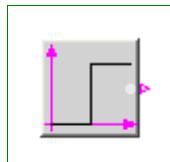
Name	Default	Description
k	true	Constant output value

## Connectors

Name	Description
y	Connector of Boolean output signal

## Modelica.Blocks.Sources.BooleanStep

Generate step signal of type Boolean



## Information

The Boolean output y is a step signal:



## Parameters

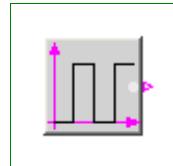
Name	Default	Description
startTime	0	Time instant of step start [s]
startValue	false	Output before startTime

## Connectors

Name	Description
y	Connector of Boolean output signal

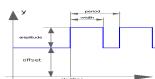
## Modelica.Blocks.Sources.BooleanPulse

Generate pulse signal of type Boolean



## Information

The Boolean output y is a pulse signal:



## Parameters

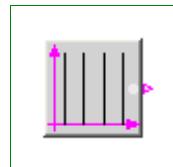
Name	Default	Description
width	50	Width of pulse in % of period
period	1	Time for one period [s]
startTime	0	Time instant of first pulse [s]

## Connectors

Name	Description
y	Connector of Boolean output signal

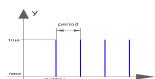
## Modelica.Blocks.Sources.SampleTrigger

Generate sample trigger signal



## Information

The Boolean output y is a trigger signal where the output y is only **true** at sample times (defined by parameter **period**) and is otherwise **false**.



## Parameters

Name	Default	Description
period	0.01	Sample period [s]
startTime	0	Time instant of first sample trigger [s]

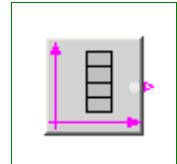
## Connectors

Name	Description
y	Connector of Boolean output signal

---

## Modelica.Blocks.Sources.BooleanTable

Generate a Boolean output signal based on a vector of time instants



## Information

The Boolean output y is a signal defined by parameter vector **table**. In the vector time points are stored. At every time point, the output y changes its value to the negated value of the previous one.



## Parameters

Name	Default	Description
startValue	false	Start value of y. At time = table[1], y changes to 'not startValue'
table[:]		Vector of time points. At every time point, the output y gets its opposite value [s]

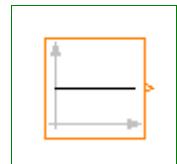
## Connectors

Name	Description
y	Connector of Boolean output signal

---

## Modelica.Blocks.Sources.IntegerConstant

Generate constant signal of type Integer



## Information

The Integer output y is a constant signal:



## Parameters

Name	Default	Description
k	1	Constant output value

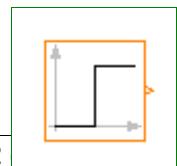
## Connectors

Name	Description
y	Connector of Integer output signal

---

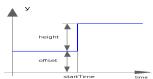
## Modelica.Blocks.Sources.IntegerStep

Generate step signal of type Integer



## Information

The Integer output y is a step signal:



## Parameters

Name	Default	Description
height	1	Height of step
offset	0	Offset of output signal y
startTime	0	Output y = offset for time < startTime [s]

## Connectors

Name	Description
y	Connector of Integer output signal

---

## Modelica.Blocks.Tables

### Library of blocks to interpolate in one and two-dimensional tables

## Information

This package contains blocks for one- and two-dimensional interpolation in tables.

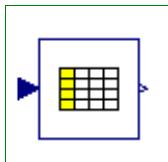
## Package Content

Name	Description
CombiTable1D	Table look-up in one dimension (matrix/file) with n inputs and n outputs
CombiTable1Ds	Table look-up in one dimension (matrix/file) with one input and n outputs
CombiTable2D	Table look-up in two dimensions (matrix/file)

---

## Modelica.Blocks.Tables.CombiTable1D

### Table look-up in one dimension (matrix/file) with n inputs and n outputs



## Information

**Linear interpolation in one dimension of a table.** Via parameter **columns** it can be defined how many columns of the table are interpolated. If, e.g., `columns={2,4}`, it is assumed that 2 input and 2 output signals are present and that the first output interpolates the first input via column 2 and the second output interpolates the second input via column 4 of the table matrix.

The grid points and function values are stored in a matrix "table[i,j]", where the first column "table[:,1]" contains the grid points and the other columns contain the data to be interpolated. Example:

```
table = [0, 0;
         1, 1;
         2, 4;
         4, 16]
If, e.g., the input u = 1.0, the output y = 1.0,
```

e.g., the input  $u = 1.5$ , the output  $y = 2.5$ ,  
e.g., the input  $u = 2.0$ , the output  $y = 4.0$ ,  
e.g., the input  $u = -1.0$ , the output  $y = -1.0$  (i.e. extrapolation).

- The interpolation is **efficient**, because a search for a new interpolation starts at the interval used in the last call.
- If the table has only **one row**, the table value is returned, independent of the value of the input signal.
- If the input signal  **$u[i]$**  is **outside** of the defined **interval**, i.e.,  $u[i] > \text{table}[\text{size}(\text{table}, 1), i+1]$  or  $u[i] < \text{table}[1, 1]$ , the corresponding value is also determined by linear interpolation through the last or first two points of the table.
- The grid values (first column) have to be **strict** monotonically increasing.

The table matrix can be defined in the following ways:

1. Explicitly supplied as **parameter matrix** "table", and the other parameters have the following values:

```
tableName is "NoName" or has only blanks,  
fileName is "NoName" or has only blanks.
```

2. **Read** from a **file** "fileName" where the matrix is stored as "tableName". Both ASCII and binary file format is possible. (the ASCII format is described below). It is most convenient to generate the binary file from Matlab (Matlab 4 storage format), e.g., by command

```
save tables.mat tab1 tab2 tab3 -v4
```

when the three tables tab1, tab2, tab3 should be used from the model.

3. Statically stored in function "usertab" in file "usertab.c". The matrix is identified by "tableName". Parameter fileName = "NoName" or has only blanks.

Table definition methods (1) and (3) do **not** allocate dynamic memory, and do not access files, whereas method (2) does. Therefore (1) and (3) are suited for hardware-in-the-loop simulation (e.g. with dSpace hardware). When the constant "NO\_FILE" is defined in "usertab.c", all parts of the source code of method (2) are removed by the C-preprocessor, such that no dynamic memory allocation and no access to files takes place.

If tables are read from an ASCII-file, the file need to have the following structure ("----" is not part of the file content):

```
-----  
#1  
double tab1(5,2)    # comment line  
0 0  
1 1  
2 4  
3 9  
4 16  
double tab2(5,2)    # another comment line  
0 0  
2 2  
4 8  
6 18  
8 32  
-----
```

Note, that the first two characters in the file need to be "#1". Afterwards, the corresponding matrix has to be declared with type, name and actual dimensions. Finally, in successive rows of the file, the elements of the matrix have to be given. Several matrices may be defined one after another.

## Parameters

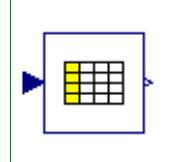
Name	Default	Description
table data definition		
tableOnFile	false	true, if table is defined on file or in function usertab
table[:, :]	fill(0.0, 0, 2)	table matrix (grid = first column)
tableName	"NoName"	table name on file or in function usertab (see docu)
fileName	"NoName"	file where matrix is stored
table data interpretation		
columns[:]	2:size(table, 2)	columns of table to be interpolated
smoothness	Types.Smoothness.LinearSegme...	smoothness of table interpolation

## Connectors

Name	Description
u[n]	Connector of Real input signals
y[n]	Connector of Real output signals

## Modelica.Blocks.Tables.CombiTable1Ds

Table look-up in one dimension (matrix/file) with one input and n outputs



## Information

**Linear interpolation** in **one** dimension of a **table**. Via parameter **columns** it can be defined how many columns of the table are interpolated. If, e.g., `icol={2,4}`, it is assumed that one input and 2 output signals are present and that the first output interpolates via column 2 and the second output interpolates via column 4 of the table matrix.

The grid points and function values are stored in a matrix "table[i,j]", where the first column "table[:,1]" contains the grid points and the other columns contain the data to be interpolated. Example:

```
table = [0, 0;
         1, 1;
         2, 4;
         4, 16]
If, e.g., the input u = 1.0, the output y = 1.0,
e.g., the input u = 1.5, the output y = 2.5,
e.g., the input u = 2.0, the output y = 4.0,
e.g., the input u = -1.0, the output y = -1.0 (i.e. extrapolation).
```

- The interpolation is **efficient**, because a search for a new interpolation starts at the interval used in the last call.
- If the table has only **one row**, the table value is returned, independent of the value of the input signal.
- If the input signal **u** is **outside** of the defined **interval**, i.e.,  $u > \text{table}[\text{size}(\text{table}, 1), 1]$  or  $u < \text{table}[1, 1]$ , the corresponding value is also determined by linear interpolation through the last or first two points of the table.
- The grid values (first column) have to be **strict** monotonically increasing.

The table matrix can be defined in the following ways:

1. Explicitly supplied as **parameter matrix** "table", and the other parameters have the following values:

```
tableName is "NoName" or has only blanks,
fileName is "NoName" or has only blanks.
```

2. **Read** from a **file** "fileName" where the matrix is stored as "tableName". Both ASCII and binary file format is possible. (the ASCII format is described below). It is most convenient to generate the binary file from Matlab (Matlab 4 storage format), e.g., by command

```
save tables.mat tab1 tab2 tab3 -v4
```

when the three tables tab1, tab2, tab3 should be used from the model.

3. Statically stored in function "usertab" in file "usertab.c". The matrix is identified by "tableName". Parameter fileName = "NoName" or has only blanks.

Table definition methods (1) and (3) do **not** allocate dynamic memory, and do not access files, whereas method (2) does. Therefore (1) and (3) are suited for hardware-in-the-loop simulation (e.g. with dSpace hardware). When the constant "NO\_FILE" is defined, all parts of the source code of method (2) are removed by the C-preprocessor, such that no dynamic memory allocation and no access to files takes place.

If tables are read from an ASCII-file, the file need to have the following structure ("----" is not part of the file content):

```
-----
#1
double tab1(5,2)    # comment line
 0   0
 1   1
 2   4
 3   9
 4  16
double tab2(5,2)    # another comment line
 0   0
 2   2
 4   8
 6  18
 8  32
-----
```

Note, that the first two characters in the file need to be "#1". Afterwards, the corresponding matrix has to be declared with type, name and actual dimensions. Finally, in successive rows of the file, the elements of the matrix have to be given. Several matrices may be defined one after another.

## Parameters

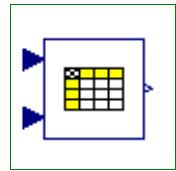
Name	Default	Description
<b>table data definition</b>		
tableOnFile	false	true, if table is defined on file or in function usertab
table[:, :]	fill(0.0, 0, 2)	table matrix (grid = first column)
tableName	"NoName"	table name on file or in function usertab (see docu)
fileName	"NoName"	file where matrix is stored
<b>table data interpretation</b>		
columns[]	2:size(table, 2)	columns of table to be interpolated
smoothness	Types.Smoothness.LinearSegme...	smoothness of table interpolation

## Connectors

Name	Description
u	Connector of Real input signal
y[nout]	Connector of Real output signals

## Modelica.Blocks.Tables.CombiTable2D

### Table look-up in two dimensions (matrix/file)



### Information

**Linear interpolation in two dimensions of a table.** The grid points and function values are stored in a matrix "table[i,j]", where:

- the first column "table[2:,1]" contains the u[1] grid points,
- the first row "table[1,2:]" contains the u[2] grid points,
- the other rows and columns contain the data to be interpolated.

Example:

```

    |   1.0   |   2.0   |   3.0   | // u2
----*-----*-----*-----*
 1.0 |   1.0   |   3.0   |   5.0   |
----*-----*-----*-----*
 2.0 |   2.0   |   4.0   |   6.0   |
----*-----*-----*-----*
// u1
is defined as
table = [0.0,    1.0,    2.0,    3.0;
         1.0,    1.0,    3.0,    5.0;
         2.0,    2.0,    4.0,    6.0]
If, e.g. the input u is [1.0;1.0], the output y is 1.0,
e.g. the input u is [2.0;1.5], the output y is 3.0.

```

- The interpolation is **efficient**, because a search for a new interpolation starts at the interval used in the last call.
- If the table has only **one element**, the table value is returned, independent of the value of the input signal.
- If the input signal **u1** or **u2** is **outside** of the defined **interval**, the corresponding value is also determined by linear interpolation through the last or first two points of the table.
- The grid values (first column and first row) have to be **strict** monotonically increasing.

The table matrix can be defined in the following ways:

1. Explicitly supplied as **parameter matrix** "table", and the other parameters have the following values:

```

tableName is "NoName" or has only blanks,
fileName is "NoName" or has only blanks.

```

2. **Read from a file** "fileName" where the matrix is stored as "tableName". Both ASCII and binary file format is possible. (the ASCII format is described below). It is most convenient to generate the binary file from Matlab (Matlab 4 storage format), e.g., by command

```
save tables.mat tab1 tab2 tab3 -v4
```

when the three tables tab1, tab2, tab3 should be used from the model.

3. Statically stored in function "usertab" in file "usertab.c". The matrix is identified by "tableName". Parameter fileName = "NoName" or has only blanks.

Table definition methods (1) and (3) do **not** allocate dynamic memory, and do not access files, whereas method (2) does. Therefore (1) and (3) are suited for hardware-in-the-loop simulation (e.g. with dSpace hardware). When the constant "NO\_FILE" is defined, all parts of the source code of method (2) are removed by the C-preprocessor, such that no dynamic memory allocation and no access to files takes place.

If tables are read from an ASCII-file, the file need to have the following structure ("----" is not part of the file content):

```
-----
#1
double tab1(5,2)    # comment line
0   0
1   1
2   4
3   9
4   16
double tab2(5,2)    # another comment line
0   0
2   2
4   8
6   18
8   32
-----
```

Note, that the first two characters in the file need to be "#1". Afterwards, the corresponding matrix has to be declared with type, name and actual dimensions. Finally, in successive rows of the file, the elements of the matrix have to be given. Several matrices may be defined one after another.

## Parameters

Name	Default	Description
table data definition		
tableOnFile	false	true, if table is defined on file or in function usertab
table[:, :]	fill(0.0, 0, 2)	table matrix (grid u1 = first column, grid u2 = first row)
tableName	"NoName"	table name on file or in function usertab (see docu)
fileName	"NoName"	file where matrix is stored
table data interpretation		
smoothness	Types.Smoothness.LinearSegme...	smoothness of table interpolation

## Connectors

Name	Description
u1	Connector of Real input signal 1
u2	Connector of Real input signal 2
y	Connector of Real output signal

---

## Modelica.Blocks.Types

Library of constants and types with choices, especially to build menus

## Information

In this package **types** and **constants** are defined that are used in library Modelica.Blocks. The types have additional annotation choices definitions that define the menus to be built up in the graphical user interface when the type is used as parameter in a declaration.

## Package Content

Name	Description
(e) Extrapolation	Type, constants and menu choices to define the extrapolation of time table interpolation

 <b>Init</b>	Type, constants and menu choices to define initialization of blocks
 <b>InitPID</b>	Type, constants and menu choices to define initialization of PID and LimPID blocks
 <b>SimpleController</b>	Type, constants and menu choices to define a simple controller type
 <b>Smoothness</b>	Type, constants and menu choices to define the smoothness of table interpolation
 <b>StateSelection</b>	Type, constants and menu choices to define state selection of variables

## Modelica.Blocks.Types.Extrapolation

Type, constants and menu choices to define the extrapolation of time table interpolation

### Information

#### Package Content

Name	Description
HoldLastPoint=0	Hold the last table point outside of the table scope
LastTwoPoints=1	Extrapolate linearly through the last two table points outside of the table scope
Periodic=2	Repeat the table scope periodically
 <b>Temp</b>	Temporary type of Extrapolation with choices for menus (until enumerations are available)

### Types and constants

```

constant Integer HoldLastPoint=0
"Hold the last table point outside of the table scope";

constant Integer LastTwoPoints=1
"Extrapolate linearly through the last two table points outside of the table
scope";

constant Integer Periodic=2 "Repeat the table scope periodically";

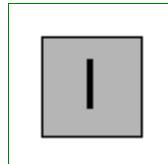
type Temp
"Temporary type of Extrapolation with choices for menus (until enumerations
are available)"

extends Modelica.Icons.TypeInteger;
end Temp;

```

## Modelica.Blocks.Types.Extrapolation.Temp

Temporary type of Extrapolation with choices for menus (until enumerations are available)



### Information

Type of extrapolation in a table.

choice	Meaning of choice
HoldLastPoint	Hold the last table point outside of the table scope

---

LastTwoPoints	Extrapolate linearly through the last two table points outside of the table scope
Periodic	Repeat the table scope periodically

## Modelica.Blocks.Types.Init

Type, constants and menu choices to define initialization of blocks

### Information

#### Package Content

Name	Description
NoInit=1	no initialization (start values are used as guess values with fixed=false)
SteadyState=2	steady state initialization (derivatives of states are zero)
InitialState=3	initialization with initial states
InitialOutput=4	initialization with initial outputs (and steady state of the states if possible)
 Temp	Temporary type of initialization with choices for menus (until enumerations are available)

#### Types and constants

```

constant Integer NoInit=1
"no initialization (start values are used as guess values with fixed=false)";

constant Integer SteadyState=2
"steady state initialization (derivatives of states are zero)";

constant Integer InitialState=3 "initialization with initial states";

constant Integer InitialOutput=4
"initialization with initial outputs (and steady state of the states if
possible)";

type Temp
"Temporary type of initialization with choices for menus (until enumerations
are available)"
extends Modelica.Icons.TypeInteger(min=1,max=4);

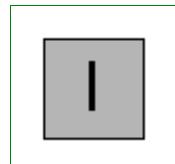
end Temp;

```

---

## Modelica.Blocks.Types.Init.Temp

Temporary type of initialization with choices for menus (until enumerations are available)



#### Parameters

Name	Default	Description
min	1	
max	4	

## Modelica.Blocks.Types.InitPID

Type, constants and menu choices to define initialization of PID and LimPID blocks

### Information

This initialization type is identical to Types.Init and has just one additional option **DoNotUse\_InitialIntegratorState**. This option is only introduced in order that the default initialization for the Continuous.PID and Continuous.LimPID blocks are backward compatible. In Modelica 2.2, the integrators have been initialized with their given states whereas the D-part has not been initialized. The option "DoNotUse\_InitialIntegratorState" leads to this initialization definition.

### Package Content

Name	Description
NoInit=1	no initialization (start values are used as guess values with fixed=false)
SteadyState=2	steady state initialization (derivatives of states are zero)
InitialState=3	initialization with initial states
InitialOutput=4	initialization with initial outputs (and steady state of the states if possible)
DoNotUse_InitialIntegratorState=5	don't use, only for backward compatibility (initialize only integrator state)
 Temp	Temporary type of initialization with choices for menus (until enumerations are available)

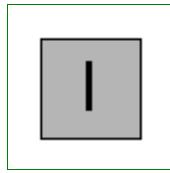
### Types and constants

```
constant Integer NoInit=1  
"no initialization (start values are used as guess values with fixed=false);  
  
constant Integer SteadyState=2  
"steady state initialization (derivatives of states are zero);  
  
constant Integer InitialState=3 "initialization with initial states";  
  
constant Integer InitialOutput=4  
"initialization with initial outputs (and steady state of the states if possible);  
  
constant Integer DoNotUse_InitialIntegratorState=5  
"don't use, only for backward compatibility (initialize only integrator state);  
  
type Temp  
"Temporary type of initialization with choices for menus (until enumerations are available)"  
extends Modelica.Icons.TypeInteger(min=1,max=5);  
  
end Temp;
```

---

**Modelica.Blocks.Types.InitPID.Temp**

**Temporary type of initialization with choices for menus (until enumerations are available)**

**Parameters**

Name	Default	Description
min	1	
max	5	

**Modelica.Blocks.Types.SimpleController**

**Type, constants and menu choices to define a simple controller type**

**Information****Package Content**

Name	Description
P=1	P controller
PI=2	PI controller
PD=3	PD controller
PID=4	PID controller
Temp	Temporary type of simple controller type with choices for menus (until enumerations are available)

**Types and constants**

```

constant Integer P=1 "P controller";

constant Integer PI=2 "PI controller";

constant Integer PD=3 "PD controller";

constant Integer PID=4 "PID controller";

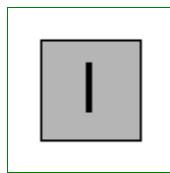
type Temp
  "Temporary type of simple controller type with choices for menus (until
  enumerations are available)"
  extends Modelica.Icons.TypeInteger(min=1,max=4);

end Temp;

```

**Modelica.Blocks.Types.SimpleController.Temp**

**Temporary type of simple controller type with choices for menus (until enumerations are available)**



## Parameters

Name	Default	Description
min	1	
max	4	

---

## Modelica.Blocks.Types.Smoothness

Type, constants and menu choices to define the smoothness of table interpolation

## Information

Interpolation type of a table

## Package Content

Name	Description
LinearSegments=0	Table points are linearly interpolated
ContinuousDerivative=1	Table points are interpolated such that the first derivative is continuous
 Temp	Temporary type of Smoothness with choices for menus (until enumerations are available)

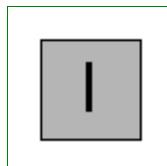
## Types and constants

```
constant Integer LinearSegments=0 "Table points are linearly interpolated";  
  
constant Integer ContinuousDerivative=1  
"Table points are interpolated such that the first derivative is continuous";  
  
type Temp  
"Temporary type of Smoothness with choices for menus (until enumerations are available)"  
  
extends Modelica.Icons.TypeInteger;  
end Temp;
```

---

## Modelica.Blocks.Types.Smoothness.Temp

Temporary type of Smoothness with choices for menus (until enumerations are available)



## Information

Smoothness of interpolation in a table.

choice	Meaning of choice
LinearSegments	Table points are linearly interpolated
ContinuousDerivative	Table points are interpolated such that the first derivative is continuous

---

## Modelica.Blocks.Types.StateSelection

Type, constants and menu choices to define state selection of variables

### Information

Type to define the stateSelection attribute.

### Package Content

Name	Description
Never=1	Never (never use as state)
Avoid=2	Avoid (avoid to use as state)
Default=3	Default (default behaviour)
Prefer=4	Prefer (use as state if possible)
Always=5	Always (always use as state)
 Temp	Temporary type of state selection with choices for menus (until enumerations are available)

### Types and constants

```
constant Integer Never=1 "Never (never use as state)";

constant Integer Avoid=2 "Avoid (avoid to use as state)";

constant Integer Default=3 "Default (default behaviour)";

constant Integer Prefer=4 "Prefer (use as state if possible)";

constant Integer Always=5 "Always (always use as state)";

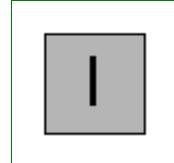
type Temp
  "Temporary type of state selection with choices for menus (until enumerations
  are available)"
  extends Modelica.Icons.TypeInteger(min=1,max=5);

end Temp;
```

---

## Modelica.Blocks.Types.StateSelection.Temp

Temporary type of state selection with choices for menus (until enumerations are available)



### Parameters

Name	Default	Description
min	1	
max	5	

---

## Modelica.Constants

Library of mathematical constants and constants of nature (e.g., pi, eps, R, sigma)

## Information

This package provides often needed constants from mathematics, machine dependent constants and constants from nature. The latter constants (name, value, description) are from the following source:

Peter J. Mohr and Barry N. Taylor (1999):

**CODATA Recommended Values of the Fundamental Physical Constants: 1998.** Journal of Physical and Chemical Reference Data, Vol. 28, No. 6, 1999 and Reviews of Modern Physics, Vol. 72, No. 2, 2000. See also <http://physics.nist.gov/cuu/Constants/>

CODATA is the Committee on Data for Science and Technology.

### Main Author:

[Martin Otter](#)

Deutsches Zentrum für Luft und Raumfahrt e. V. (DLR)  
Oberpfaffenhofen  
Postfach 11 16  
D-82230 Weßling  
email: [Martin.Otter@dlr.de](mailto:Martin.Otter@dlr.de)

Copyright © 1998-2007, Modelica Association and DLR.

*This Modelica package is free software; it can be redistributed and/or modified under the terms of the Modelica license, see the license conditions and the accompanying disclaimer [here](#).*

## Package Content

Name	Description
e=Modelica.Math.exp(1.0)	
pi=2*Modelica.Math.asin(1.0)	
D2R=pi/180	Degree to Radian
R2D=180/pi	Radian to Degree
eps=1.e-15	Biggest number such that 1.0 + eps = 1.0
small=1.e-60	Smallest number such that small and -small are representable on the machine
inf=1.e+60	Biggest Real number such that inf and -inf are representable on the machine
Integer_inf=2147483647	Biggest Integer number such that Integer_inf and -Integer_inf are representable on the machine
c=299792458	Speed of light in vacuum
g_n=9.80665	Standard acceleration of gravity on earth
G=6.6742e-11	Newtonian constant of gravitation
h=6.6260693e-34	Planck constant
k=1.3806505e-23	Boltzmann constant
R=8.314472	Molar gas constant
sigma=5.670400e-8	Stefan-Boltzmann constant
N_A=6.0221415e23	Avogadro constant
mue_0=4*pi*1.e-7	Magnetic constant
epsilon_0=1/(mue_0*c*c)	Electric constant
T_zero=-273.15	Absolute zero temperature

## Types and constants

```
constant Real e=Modelica.Math.exp(1.0);
```

```

constant Real pi=2*Modelica.Math.asin(1.0);

constant Real D2R=pi/180 "Degree to Radian";

constant Real R2D=180/pi "Radian to Degree";

constant Real eps=1.e-15 "Biggest number such that 1.0 + eps = 1.0";

constant Real small=1.e-60
"Smallest number such that small and -small are representable on the machine";

constant Real inf=1.e+60
"Biggest Real number such that inf and -inf are representable on the machine";

constant Integer Integer_inf=2147483647
"Biggest Integer number such that Integer_inf and -Integer_inf are
representable on the machine";

constant SI.Velocity c=299792458 "Speed of light in vacuum";

constant SI.Acceleration g_n=9.80665
"Standard acceleration of gravity on earth";

constant Real G(final unit="m3/(kg.s2)") = 6.6742e-11
"Newtonian constant of gravitation";

constant Real h(final unit="J.s") = 6.6260693e-34 "Planck constant";

constant Real k(final unit="J/K") = 1.3806505e-23 "Boltzmann constant";

constant Real R(final unit="J/(mol.K)") = 8.314472 "Molar gas constant";

constant Real sigma(final unit="W/(m2.K4)") = 5.670400e-8
"Stefan-Boltzmann constant";

constant Real N_A(final unit="1/mol") = 6.0221415e23 "Avogadro constant";

constant Real mue_0(final unit="N/A2") = 4*pi*1.e-7 "Magnetic constant";

constant Real epsilon_0(final unit="F/m") = 1/(mue_0*c*c) "Electric constant";

constant NonSI.Temperature_degC T_zero=-273.15 "Absolute zero temperature";

```

## Modelica.Electrical

**Library of electrical models (analog, digital, machines, multi-phase)**

### Information

This library contains electrical components to build up analog and digital circuits, as well as machines to model electrical motors and generators, especially three phase induction machines such as an asynchronous motor.

## Package Content

Name	Description
 Analog	Library for analog electrical models
 Digital	Library for digital electrical components based on the VHDL standard with 9-valued logic and conversion to 2-,3-,4-valued logic
 Machines	Library for electric machines
 MultiPhase	Library for electrical components with 2, 3 or more phases

---

## Modelica.Electrical.Analog

### Library for analog electrical models

#### Information

This package contains packages for analog electrical components:

- Basic: basic components (resistor, capacitor, conductor, inductor, transformer, gyrator)
- Semiconductors: semiconductor devices (diode, bipolar and MOS transistors)
- Lines: transmission lines (lossy and lossless)
- Ideal: ideal elements (switches, diode, transformer, idle, short, ...)
- Sources: time-dependend and controlled voltage and current sources
- Sensors: sensors to measure potential, voltage, and current

#### Main Authors:

Christoph Clauß <clauss@eas.iis.fhg.de>  
André Schneider <schneider@eas.iis.fhg.de>  
Fraunhofer Institute for Integrated Circuits  
Design Automation Department  
Zeunerstraße 38  
D-01069 Dresden

Copyright © 1998-2007, Modelica Association and Fraunhofer-Gesellschaft.

*This Modelica package is free software; it can be redistributed and/or modified under the terms of the Modelica license, see the license conditions and the accompanying disclaimer here.*

## Package Content

Name	Description
 Examples	Examples that demonstrate the usage of the Analog electrical components
 Basic	Basic electrical components such as resistor, capacitor, transformer
 Ideal	Ideal electrical elements such as switches, diode, transformer, operational amplifier
 Interfaces	Connectors and partial models for Analog electrical components
 Lines	Lossy and lossless segmented transmission lines, and LC distributed line models
 Semiconductors	Semiconductor devices such as diode, MOS and bipolar transistor
 Sensors	Potential, voltage, current, and power sensors
 Sources	Time-dependend and controlled voltage and current sources

---

## Modelica.Electrical.Analog.Examples

Examples that demonstrate the usage of the Analog electrical components

### Information

This package contains examples that demonstrate the usage of the components of the Electrical.Analog library.

### Package Content

Name	Description
<a href="#">CauerLowPassAnalog</a>	Cauer low pass filter with analog components
<a href="#">CauerLowPassOPV</a>	Cauer low pass filter with operational amplifiers
<a href="#">CauerLowPassSC</a>	Cauer low-pass filter with operational amplifiers and switched capacitors
<a href="#">CharacteristicIdealDiodes</a>	Characteristic of ideal diodes
<a href="#">CharacteristicThyristors</a>	Characteristic of ideal thyristors
<a href="#">ChuaCircuit</a>	Chua's circuit, ns, V, A
<a href="#">DifferenceAmplifier</a>	Simple NPN transistor amplifier circuit
<a href="#">HeatingMOSInverter</a>	Heating MOS Inverter
<a href="#">HeatingNPN_OrGate</a>	Heating NPN Or Gate
<a href="#">HeatingRectifier</a>	Heating rectifier
<a href="#">NandGate</a>	CMOS NAND Gate (see Tietze/Schenk, page 157)
<a href="#">Rectifier</a>	B6 diode bridge
<a href="#">ShowSaturatingInductor</a>	Simple demo to show behaviour of SaturatingInductor component
<a href="#">ShowVariableResistor</a>	Simple demo of a VariableResistor model
<a href="#">Utilities</a>	Utility components used by package Examples

## Modelica.Electrical.Analog.Examples.CauerLowPassAnalog

Cauer low pass filter with analog components



### Information

The example Cauer Filter is a low-pass-filter of the fifth order. It is realized using an analog network. The voltage source V is the input voltage (step), and the R2.p.v is the filter output voltage. The pulse response is calculated.

The simulation end time should be 60. Please plot both V.p.v (input voltage) and R2.p.v (output voltage).

### Parameters

Name	Default	Description
I1	1.304	[H]
I2	0.8586	[H]
c1	1.072	[F]
c2	$1/(1.704992^2 * I1)$	[F]

	)	
c3	1.682	[F]
c4	$1/(1.179945^2 \cdot I2)$ )	[F]
c5	0.7262	[F]

**Modelica.Electrical.Analog.Examples.CauerLowPassOPV****Cauer low pass filter with operational amplifiers****Information**

The example Cauer Filter is a low-pass-filter of the fifth order. It is realized using an analog network with operational amplifiers. The voltage source V is the input voltage (step), and the OP5.out.v is the filter output voltage. The pulse response is calculated.

This model is identical to the CauerLowPassAnalog example, but inverting. To get the same response as that of the CauerLowPassAnalog example, a negative voltage step is used as input.

The simulation end time should be 60. Please plot both V.v (which is the inverted input voltage) and OP5.p.v (output voltage). Compare this result with the CauerLowPassAnalog result.

During translation some warnings are issued concerning resistor values (Value=-1 not in range[0,1.e+100]). Do not worry about it. The negative values are o.k.

**Parameters**

Name	Default	Description
I1	1.304	[F]
I2	0.8586	[F]
c1	1.072	[F]
c2	$1/(1.704992^2 \cdot I1)$ )	[F]
c3	1.682	[F]
c4	$1/(1.179945^2 \cdot I2)$ )	[F]
c5	0.7262	[F]

**Modelica.Electrical.Analog.Examples.CauerLowPassSC****Cauer low-pass filter with operational amplifiers and switched capacitors****Information**

The example CauerLowPassSC is a low-pass-filter of the fifth order. It is realized using an switched-capacitor network with operational amplifiers. The voltage source V is the input voltage (step), and the OP5.out.v is the filter output voltage. The pulse response is calculated.

This model is identical to the CauerLowPassAnalog example, but inverting. To get the same response as that of the CauerLowPassAnalog example, a negative voltage step is used as input.

This model is identical to the CauerLowPassOPV example. But the resistors are realized by switched capacitors. There are two such resistors Rp (of value +1), and Rn (of value -1). In this models the switching clock source is included. In a typical switched capacitor circuit there would be a central clock source.

The simulation end time should be 60. Please plot both V.v (which is the inverted input voltage) and OP5.p.v (output voltage). Compare this result with the CauerLowPassAnalog result.

Due to the recharging of the capacitances after switching the performance of simulation is not as good as in the CauerLowPassOPV example.

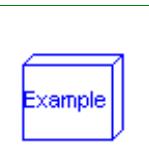
## Parameters

Name	Default	Description
I1	1.304	
I2	0.8586	
c1	1.072	
c2	$1/(1.704992^2 * I1)$	
c3	1.682	
c4	$1/(1.179945^2 * I2)$	
c5	0.7262	

---

## Modelica.Electrical.Analog.Examples.CharacteristicIdealDiodes

### Characteristic of ideal diodes



#### Information

Three examples of ideal diodes are shown:

the **totally ideal diode** (ideal) with all parameters to be zero

the **nearly ideal diode** with  $Ron=0.1$  and  $Goff=0.1$

the nearly ideal but **displaced diode** with  $Vknee=5$  and  $Ron=0.1$  and  $Goff=0.1$

The resistance and conductance are chosen untypically high since the slopes should be seen in the graphics.

Simulate until  $T=1$  s.

Plot in separate windows:

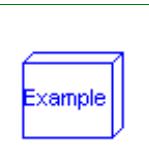
Ideal.i versus Ideal.v

With\_Ron\_Goff.i versus With\_Ron\_Goff.v

With\_Ron\_Goff\_Vknee.i versus With\_Ron\_Goff\_Vknee.v

---

## Modelica.Electrical.Analog.Examples.CharacteristicThyristors



### Characteristic of ideal thyristors

#### Information

Two examples of thyristors are shown:

the **ideal thyristor**

and the **ideal GTO thyristor** with  $Vknee=5$

Simulate until  $T=2$  s.

## 202 Modelica.Electrical.Analog.Examples.CharacteristicThyristors

---

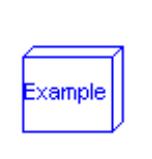
Plot in separate windows:

IdealThyristor1.i and IdealGTOThyristor1.i  
IdealThyristor1.v and IdealGTOThyristor1.v

---

## Modelica.Electrical.Analog.Examples.ChuaCircuit

Chua's circuit, ns, V, A



### Information

Chua's circuit is the most simple nonlinear circuit which shows chaotic behaviour. The circuit consists of linear basic elements (capacitors, resistor, conductor, inductor), and one nonlinear element, which is called Chua's diode. The chaotic behaviour is simulated.

The simulation end time should be set to 5e4. To get the chaotic behaviour please plot C1.v. Choose C2.v as the independent variable.

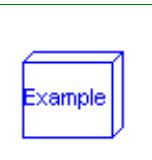
### Reference:

Kennedy, M.P.: Three Steps to Chaos - Part I: Evolution. IEEE Transactions on CAS I 40 (1993)10, 640-656

---

## Modelica.Electrical.Analog.Examples.DifferenceAmplifier

Simple NPN transistor amplifier circuit



### Information

It is a simple NPN transistor amplifier circuit. The voltage difference between R1.p and R3.n is amplified. The output signal is the voltage between R2.n and R4.n. In this example the voltage at V1 is amplified because R3.n is grounded.

The simulation end time should be set to 1e- 8. Please plot the input voltage V1.v, and the output voltages R2.n.v, and R4.n.v.

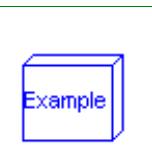
### Reference:

Tietze, U.; Schenk, Ch.: Halbleiter-Schaltungstechnik. Springer-Verlag Berlin Heidelberg NewYork 1980, p. 59

---

## Modelica.Electrical.Analog.Examples.HeatingMOSInverter

Heating MOS Inverter



### Information

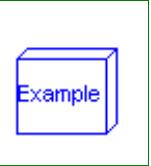
The heating MOS inverter shows a heat flow always if a transistor is leading.

Simulate until T=5 s.

Plot in separate windows:

Sin.p.v and Capacitor1.p.v  
HeatCapacitor1.port.T and H\_PMOS.heatPort.T and H\_NMOS.heatPort.T  
H\_PMOS.heatPort.Q\_flow and H\_NMOS.heatPort.Q\_flow

---

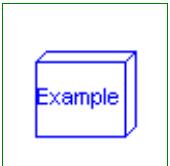
**Modelica.Electrical.Analog.Examples.HeatingNPN\_OrGate****Heating NPN Or Gate****Information**

The heating NPN nand gate shows a heat flow always if a transistor is leading.

Simulate until T=200 s.

Plot in separate windows:

V1.v and V2.v and C2.v  
HeatCapacitor1.port.T and T1.heatPort.T and T2.heatPort.T  
T1.heatPort.Q\_flow and T2.heatPort.Q\_flow

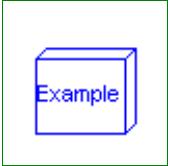
**Modelica.Electrical.Analog.Examples.HeatingRectifier****Heating rectifier****Information**

The heating rectifier shows a heat flow always if the electrical capacitor is loaded.

Simulate until T=5 s.

Plot in separate windows:

SineVoltage1.v and Capacitor1.p.v  
HeatCapacitor1.port.T and HeatingDiode1.heatPort.T  
HeatingDiode1.heatPort.Q\_flow

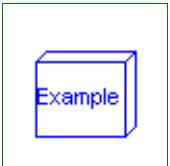
**Modelica.Electrical.Analog.Examples.NandGate****CMOS NAND Gate (see Tietze/Schenk, page 157)****Information**

The nand gate is a basic CMOS building block. It consists of four CMOS transistors. The output voltage Nand.y.v is low if and only if the two input voltages at Nand.x1.v and Nand.x2.v are both high. In this way the nand functionality is realized.

The simulation end time should be set to 1e-7. Please plot the input voltages Nand.x1.v, d Nand.x2.v, and the output voltage Nand.y.v.

**Reference:**

Tietze, U.; Schenk, Ch.: Halbleiter-Schaltungstechnik. Springer-Verlag Berlin Heidelberg NewYork 1980, p. 157

**Modelica.Electrical.Analog.Examples.Rectifier****B6 diode bridge**

## Information

The rectifier example shows a B6 diode bridge fed by a three phase sinusoidal voltage, loaded by a DC current.

DC capacitors start at ideal no-load voltage, thus making easier initial transient.

Simulate until T=0.1 s.

Plot in separate windows:

uDC ... DC-voltage

iAC ... AC-currents 1..3

uAC ... AC-voltages 1..3 (distorted)

Try different load currents iDC = 0..approximately 500 A.

You may watch Losses (of the whole diode bridge) trying different diode parameters.

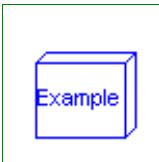
## Parameters

Name	Default	Description
VAC	400	RMS line-to-line [V]
f	50	line frequency [Hz]
LAC	60E-6	line inductor [H]
Ron	1E-3	diode forward resistance [Ohm]
Goff	1E-3	diode backward conductance [S]
Vknee	2	diode threshold voltage [V]
CDC	15E-3	DC capacitance [F]
IDC	500	load current [A]

---

## Modelica.Electrical.Analog.Examples.ShowSaturatingInductor

Simple demo to show behaviour of SaturatingInductor component



## Information

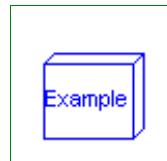
## Parameters

Name	Default	Description
Lzer	2	[H]
Lnom	1	[H]
Inom	1	[A]
Linf	0.5	[H]
U	1.25	[V]
f	1/(2*Modelica.Constants.pi)	[Hz]
phase	Modelica.Constants.pi/2	[rad]

---

## Modelica.Electrical.Analog.Examples>ShowVariableResistor

Simple demo of a VariableResistor model



## Information

It is a simple test circuit for the VariableResistor. The VariableResistor could be compared with R2. Simulate until T=1 s.

## Modelica.Electrical.Analog.Examples.Utilities

### Utility components used by package Examples

## Information

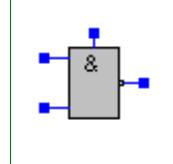
This package contains utility components used by package Examples.

## Package Content

Name	Description
 Nand	CMOS NAND Gate (see Tietze/Schenk, page 157)
 NonlinearResistor	Chua's resistor
 RealSwitch	
 Transistor	

## Modelica.Electrical.Analog.Examples.Utilities.Nand

### CMOS NAND Gate (see Tietze/Schenk, page 157)



## Information

The nand gate is a basic CMOS building block. It consists of four CMOS transistors.

## Reference:

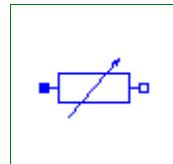
Tietze, U.; Schenk, Ch.: Halbleiter-Schaltungstechnik. Springer-Verlag Berlin Heidelberg NewYork 1980, p. 157

## Connectors

Name	Description
x1	
x2	
Vdd	
y	

## Modelica.Electrical.Analog.Examples.Utilities.NonlinearResistor

### Chua's resistor



## Information

## Parameters

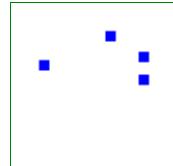
Name	Default	Description
Ga		[S]
Gb		[S]
Ve		[V]

## Connectors

Name	Description
p	Positive pin (potential p.v > n.v for positive voltage drop v)
n	Negative pin

---

## Modelica.Electrical.Analog.Examples.Utilities.RealSwitch



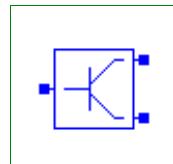
## Information

## Connectors

Name	Description
p	
n1	
n2	
control	

---

## Modelica.Electrical.Analog.Examples.Utilities.Transistor



## Information

## Connectors

Name	Description
c	
b	
e	

---

## Modelica.Electrical.Analog.Basic

Basic electrical components such as resistor, capacitor, transformer

## Information

This package contains basic analog electrical components.

## Package Content

Name	Description
 Ground	Ground node
 Resistor	Ideal linear electrical resistor
 HeatingResistor	Temperature dependent electrical resistor
 Conductor	Ideal linear electrical conductor
 Capacitor	Ideal linear electrical capacitor
 Inductor	Ideal linear electrical inductor
 SaturatingInductor	Simple model of an inductor with saturation
 Transformer	Transformer with two ports
 Gyrator	Gyrator
 EMF	Electromotoric force (electric/mechanic transformer)
 VCV	Linear voltage-controlled voltage source
 VCC	Linear voltage-controlled current source
 CCV	Linear current-controlled voltage source
 CCC	Linear current-controlled current source
 OpAmp	Simple nonideal model of an OpAmp with limitation
 VariableResistor	Ideal linear electrical resistor with variable resistance
 VariableConductor	Ideal linear electrical conductor with variable conductance
 VariableCapacitor	Ideal linear electrical capacitor with variable capacitance
 VariableInductor	Ideal linear electrical inductor with variable inductance

## Modelica.Electrical.Analog.Basic.Ground

### Ground node



### Information

Ground of an electrical circuit. The potential at the ground node is zero. Every electrical circuit has to contain at least one ground object.

### Connectors

Name	Description
p	

## Modelica.Electrical.Analog.Basic.Resistor

### Ideal linear electrical resistor



### Information

The linear resistor connects the branch voltage  $v$  with the branch current  $i$  by  $i \cdot R = v$ . The Resistance  $R$  is

## 208 Modelica.Electrical.Analog.Basic.Resistor

---

allowed to be positive, zero, or negative.

### Parameters

Name	Default	Description
R	1	Resistance [Ohm]

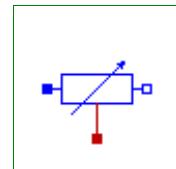
### Connectors

Name	Description
p	Positive pin (potential p.v > n.v for positive voltage drop v)
n	Negative pin

---

## Modelica.Electrical.Analog.Basic.HeatingResistor

Temperature dependent electrical resistor



### Information

This is a model for an electrical resistor where the generated heat is dissipated to the environment via connector **heatPort** and where the resistance R is temperature dependent according to the following equation:

$$R = R_{\text{ref}} * (1 + \alpha * (\text{heatPort.T} - T_{\text{ref}}))$$

**alpha** is the **temperature coefficient of resistance**, which is often abbreviated as **TCR**. In resistor catalogues, it is usually defined as **X [ppm/K]** (parts per million, similarly to per centage) meaning **X\*1.e-6 [1/K]**. Resistors are available for 1 .. 7000 ppm/K, i.e., alpha = 1e-6 .. 7e-3 1/K;

When connector **heatPort** is **not** connected, the temperature dependent behaviour is switched off by setting **heatPort.T = T\_ref**. Additionally, the equation **heatPort.Q\_flow = 0** is implicitly present due to a special rule in Modelica that flow variables of not connected connectors are set to zero.

### Parameters

Name	Default	Description
R_ref	1	Resistance at temperature T_ref [Ohm]
T_ref	300	Reference temperature [K]
alpha	0	Temperature coefficient of resistance [1/K]

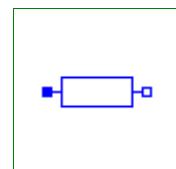
### Connectors

Name	Description
p	Positive pin (potential p.v > n.v for positive voltage drop v)
n	Negative pin
heatPort	

---

## Modelica.Electrical.Analog.Basic.Conductor

Ideal linear electrical conductor



## Information

The linear conductor connects the branch voltage  $v$  with the branch current  $i$  by  $i = v * G$ . The Conductance  $G$  is allowed to be positive, zero, or negative.

## Parameters

Name	Default	Description
G	1	Conductance [S]

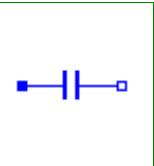
## Connectors

Name	Description
p	Positive pin (potential p.v > n.v for positive voltage drop v)
n	Negative pin

---

## Modelica.Electrical.Analog.Basic.Capacitor

Ideal linear electrical capacitor



## Information

The linear capacitor connects the branch voltage  $v$  with the branch current  $i$  by  $i = C * dv/dt$ . The Capacitance  $C$  is allowed to be positive, zero, or negative.

## Parameters

Name	Default	Description
C	1	Capacitance [F]

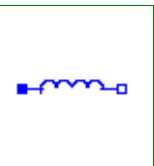
## Connectors

Name	Description
p	Positive pin (potential p.v > n.v for positive voltage drop v)
n	Negative pin

---

## Modelica.Electrical.Analog.Basic.Inductor

Ideal linear electrical inductor



## Information

The linear inductor connects the branch voltage  $v$  with the branch current  $i$  by  $v = L * di/dt$ . The Inductance  $L$  is allowed to be positive, zero, or negative.

## Parameters

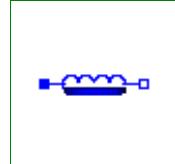
Name	Default	Description
L	1	Inductance [H]

## Connectors

Name	Description
p	Positive pin (potential p.v > n.v for positive voltage drop v)
n	Negative pin

## Modelica.Electrical.Analog.Basic.SaturatingInductor

Simple model of an inductor with saturation



## Information

This model approximates the behaviour of an inductor with the influence of saturation, i.e. the value of the inductance depends on the current flowing through the inductor. The inductance decreases as current increases.

The parameters are:

- $I_{nom}$ ...nominal current
- $L_{nom}$ ...nominal inductance at nominal current
- $L_{zer}$ ...inductance near current = 0;  $L_{zer}$  has to be greater than  $L_{nom}$
- $L_{inf}$ ...inductance at large currents;  $L_{inf}$  has to be less than  $L_{nom}$

## Parameters

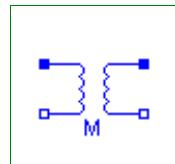
Name	Default	Description
$I_{nom}$	1	Nominal current [A]
$L_{nom}$	1	Nominal inductance at Nominal current [H]
$L_{zer}$	$2*L_{nom}$	Inductance near current=0 [H]
$L_{inf}$	$L_{nom}/2$	Inductance at large currents [H]

## Connectors

Name	Description
p	Positive pin (potential p.v > n.v for positive voltage drop v)
n	Negative pin

## Modelica.Electrical.Analog.Basic.Transformer

Transformer with two ports



## Information

The transformer is a two port. The left port voltage  $v1$ , left port current  $i1$ , right port voltage  $v2$  and right port current  $i2$  are connected by the following relation:

$$\begin{array}{l|l} | \quad v1 \quad | & | \quad L1 \quad M \quad | \quad | \quad i1' \quad | \\ | \quad | \quad = \quad | \quad | \quad | \quad | \quad | \\ | \quad v2 \quad | & | \quad M \quad L2 \quad | \quad | \quad i2' \quad | \end{array}$$

$L1$ ,  $L2$ , and  $M$  are the primary, secondary, and coupling inductances respectively.

## Parameters

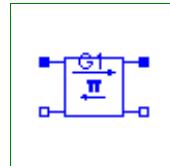
Name	Default	Description
L1	1	Primary inductance [H]
L2	1	Secondary inductance [H]
M	1	Coupling inductance [H]

## Connectors

Name	Description
p1	Positive pin of the left port (potential p1.v > n1.v for positive voltage drop v1)
n1	Negative pin of the left port
p2	Positive pin of the right port (potential p2.v > n2.v for positive voltage drop v2)
n2	Negative pin of the right port

## Modelica.Electrical.Analog.Basic.Gyrator

### Gyrator



## Information

A gyrator is a two-port element defined by the following equations:

$$\begin{aligned} i_1 &= G_2 \cdot v_2 \\ i_2 &= -G_1 \cdot v_1 \end{aligned}$$

where the constants  $G_1$ ,  $G_2$  are called the gyration conductance.

## Parameters

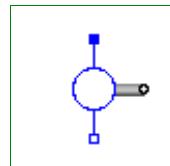
Name	Default	Description
G1	1	Gyration conductance [S]
G2	1	Gyration conductance [S]

## Connectors

Name	Description
p1	Positive pin of the left port (potential p1.v > n1.v for positive voltage drop v1)
n1	Negative pin of the left port
p2	Positive pin of the right port (potential p2.v > n2.v for positive voltage drop v2)
n2	Negative pin of the right port

## Modelica.Electrical.Analog.Basic.EMF

### Electromotoric force (electric/mechanic transformer)



## Information

EMF transforms electrical energy into rotational mechanical energy. It is used as basic building block of an electrical motor. The mechanical connector flange\_b can be connected to elements of the

## 212 Modelica.Electrical.Analog.Basic.EMF

---

Modelica.Mechanics.Rotational library. flange\_b.tau is the cut-torque, flange\_b.phi is the angle at the rotational connection.

### Parameters

Name	Default	Description
k	1	Transformation coefficient [N.m/A]

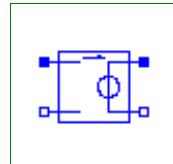
### Connectors

Name	Description
p	
n	
flange_b	

---

## Modelica.Electrical.Analog.Basic.VCV

### Linear voltage-controlled voltage source



### Information

The linear voltage-controlled voltage source is a TwoPort. The right port voltage v2 is controlled by the left port voltage v1 via

$$v2 = v1 * \text{gain}.$$

The left port current is zero. Any voltage gain can be chosen.

### Parameters

Name	Default	Description
gain	1	Voltage gain

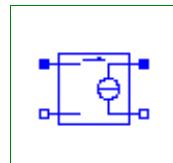
### Connectors

Name	Description
p1	Positive pin of the left port (potential p1.v > n1.v for positive voltage drop v1)
n1	Negative pin of the left port
p2	Positive pin of the right port (potential p2.v > n2.v for positive voltage drop v2)
n2	Negative pin of the right port

---

## Modelica.Electrical.Analog.Basic.VCC

### Linear voltage-controlled current source



### Information

The linear voltage-controlled current source is a TwoPort. The right port current i2 is controlled by the left port voltage v1 via

$$i2 = v1 * \text{transConductance}.$$

The left port current is zero. Any transConductance can be chosen.

## Parameters

Name	Default	Description
transConductance	1	Transconductance [S]

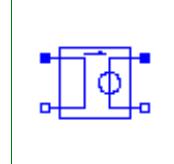
## Connectors

Name	Description
p1	Positive pin of the left port (potential p1.v > n1.v for positive voltage drop v1)
n1	Negative pin of the left port
p2	Positive pin of the right port (potential p2.v > n2.v for positive voltage drop v2)
n2	Negative pin of the right port

---

## Modelica.Electrical.Analog.Basic.CCV

### Linear current-controlled voltage source



## Information

The linear current-controlled voltage source is a TwoPort. The right port voltage v2 is controlled by the left port current i1 via

$$v2 = i1 * \text{transResistance}.$$

The left port voltage is zero. Any transResistance can be chosen.

## Parameters

Name	Default	Description
transResistance	1	Transresistance [Ohm]

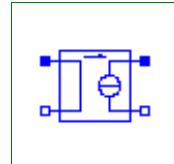
## Connectors

Name	Description
p1	Positive pin of the left port (potential p1.v > n1.v for positive voltage drop v1)
n1	Negative pin of the left port
p2	Positive pin of the right port (potential p2.v > n2.v for positive voltage drop v2)
n2	Negative pin of the right port

---

## Modelica.Electrical.Analog.Basic.CCC

### Linear current-controlled current source



## Information

The linear current-controlled current source is a TwoPort. The right port current i2 is controlled by the left port current i1 via

$$i2 = i1 * \text{gain}.$$

## 214 Modelica.Electrical.Analog.Basic.CCC

---

The left port voltage is zero. Any current gain can be chosen.

### Parameters

Name	Default	Description
gain	1	Current gain

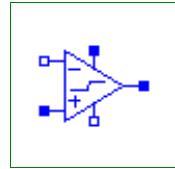
### Connectors

Name	Description
p1	Positive pin of the left port (potential p1.v > n1.v for positive voltage drop v1)
n1	Negative pin of the left port
p2	Positive pin of the right port (potential p2.v > n2.v for positive voltage drop v2)
n2	Negative pin of the right port

---

## Modelica.Electrical.Analog.Basic.OpAmp

Simple nonideal model of an OpAmp with limitation



### Information

The OpAmp is a simle nonideal model with a smooth  $\text{out.v} = f(\text{vin})$  characteristic, where " $\text{vin} = \text{in\_p.v} - \text{in\_n.v}$ ". The characteristic is limited by  $\text{VMax.v}$  and  $\text{VMMin.v}$ . Its slope at  $\text{vin}=0$  is the parameter **Slope**, which must be positive. (Therefore, the absolute value of Slope is taken into calculation.)

### Parameters

Name	Default	Description
Slope	1	Slope of the $\text{out.v}/\text{vin}$ characteristic at $\text{vin}=0$

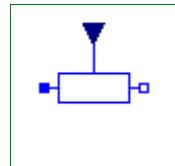
### Connectors

Name	Description
in_p	Positive pin of the input port
in_n	Negative pin of the input port
out	Output pin
VMax	Positive output voltage limitation
VMMin	Negative output voltage limitation

---

## Modelica.Electrical.Analog.Basic.VariableResistor

Ideal linear electrical resistor with variable resistance



### Information

The linear resistor connects the branch voltage  $v$  with the branch current  $i$  by

$$i \cdot R = v$$

The Resistance  $R$  is given as input signal.

### Attention!!!

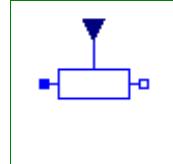
It is recommended that the R signal should not cross the zero value. Otherwise depending on the surrounding circuit the probability of singularities is high.

## Connectors

Name	Description
p	Positive pin (potential p.v > n.v for positive voltage drop v)
n	Negative pin
R	

## Modelica.Electrical.Analog.Basic.VariableConductor

Ideal linear electrical conductor with variable conductance



## Information

The linear conductor connects the branch voltage  $v$  with the branch current  $i$  by

$$i = G * v$$

The Conductance  $G$  is given as input signal.

## Attention!!!

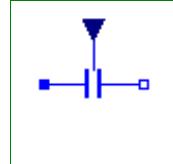
It is recommended that the G signal should not cross the zero value. Otherwise depending on the surrounding circuit the probability of singularities is high.

## Connectors

Name	Description
p	Positive pin (potential p.v > n.v for positive voltage drop v)
n	Negative pin
G	

## Modelica.Electrical.Analog.Basic.VariableCapacitor

Ideal linear electrical capacitor with variable capacitance



## Information

The linear capacitor connects the branch voltage  $v$  with the branch current  $i$  by

$$i = dQ/dt \text{ with } Q = C * v .$$

The capacitance  $C$  is given as input signal.

It is required that  $C \geq 0$ , otherwise an assertion is raised. To avoid a variable index system,  $C = C_{\min}$ , if  $0 \leq C < C_{\min}$ , where  $C_{\min}$  is a parameter with default value Modelica.Constants.eps.

## Parameters

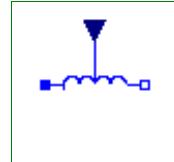
Name	Default	Description
Cmin	Modelica.Constants.eps	[F]

## Connectors

Name	Description
p	Positive pin (potential p.v > n.v for positive voltage drop v)
n	Negative pin
C	

## Modelica.Electrical.Analog.Basic.VariableInductor

Ideal linear electrical inductor with variable inductance



## Information

The linear inductor connects the branch voltage  $v$  with the branch current  $i$  by

$$v = d \Psi i / dt \text{ with } \Psi i = L * i.$$

The inductance  $L$  is as input signal.

It is required that  $L \geq 0$ , otherwise an assertion is raised. To avoid a variable index system,  $L = L_{\min}$ , if  $0 \leq L < L_{\min}$ , where  $L_{\min}$  is a parameter with default value Modelica.Constants.eps.

## Parameters

Name	Default	Description
$L_{\min}$	Modelica.Constants.eps	[H]

## Connectors

Name	Description
p	Positive pin (potential p.v > n.v for positive voltage drop v)
n	Negative pin
L	

## Modelica.Electrical.Analog.Ideal

Ideal electrical elements such as switches, diode, transformer, operational amplifier

## Information

This package contains electrical components with idealized behaviour:

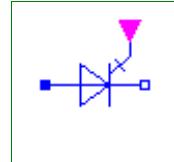
## Package Content

Name	Description
IdealThyristor	Ideal thyristor
IdealGTOThyristor	Ideal GTO thyristor
IdealCommutingSwitch	Ideal commuting switch
IdealIntermediateSwitch	Ideal intermediate switch

 ControlledIdealCommutingSwitch	Controlled ideal commuting switch
 ControlledIdealIntermediateSwitch	Controlled ideal intermediate switch
 IdealOpAmp	Ideal operational amplifier (norator-nullator pair)
 IdealOpAmp3Pin	Ideal operational amplifier (norator-nullator pair), but 3 pins
 IdealOpAmpLimited	Ideal operational amplifier with limitation
 IdealDiode	Ideal diode
 IdealTransformer	Ideal electrical transformer
 IdealGyrator	Ideal gyrator
 Idle	Idle branch
 Short	Short cut branch
 IdealOpeningSwitch	Ideal electrical opener
 IdealClosingSwitch	Ideal electrical closer
 ControlledIdealOpeningSwitch	Controlled ideal electrical opener
 ControlledIdealClosingSwitch	Controlled ideal electrical closer

## Modelica.Electrical.Analog.Ideal.IdealThyristor

### Ideal thyristor



### Information

This is an ideal thyristor model which is

**open** (off), if the voltage drop is less than 0 or fire is false

**closed** (on), if the voltage drop is greater or equal 0 and fire is true.

This is the behaviour if all parameters are exactly zero.

Note, there are circuits, where this ideal description with zero resistance and zero cinductance is not possible. In order to prevent singularities during switching, the opened thyristor has a small conductance *Goff* and the closed thyristor has a low resistance *Ron* which is default.

The parameter *Vknee* which is the forward threshold voltage, allows to displace the knee point along the *Goff*-characteristic until  $v = V_{knee}$ .

### Parameters

Name	Default	Description
Ron	1.E-5	Closed thyristor resistance [Ohm]
Goff	1.E-5	Opened thyristor conductance [S]
Vknee	0	Forward threshold voltage [V]

### Connectors

Name	Description
p	Positive pin (potential p.v > n.v for positive voltage drop v)
n	Negative pin

fire

**Modelica.Electrical.Analog.Ideal.IdealGTOThyristor****Ideal GTO thyristor****Information**

This is an ideal GTO thyristor model which is

**open** (off), if the voltage drop is less than 0 or fire is false

**closed** (on), if the voltage drop is greater or equal 0 and fire is true.

This is the behaviour if all parameters are exactly zero.

Note, there are circuits, where this ideal description with zero resistance and zero conductance is not possible. In order to prevent singularities during switching, the opened thyristor has a small conductance *Goff* and the closed thyristor has a low resistance *Ron* which is default.

The parameter *Vknee* which is the forward threshold voltage, allows to displace the knee point along the *Goff*-characteristic until  $v = V_{knee}$ .

**Parameters**

Name	Default	Description
Ron	1.E-5	Closed thyristor resistance [Ohm]
Goff	1.E-5	Opened thyristor conductance [S]
Vknee	0	Forward threshold voltage [V]

**Connectors**

Name	Description
p	Positive pin (potential p.v > n.v for positive voltage drop v)
n	Negative pin
fire	

**Modelica.Electrical.Analog.Ideal.IdealCommutingSwitch****Ideal commuting switch****Information**

The commuting switch has a positive pin p and two negative pins n1 and n2. The switching behaviour is controlled by the input signal control. If control is true, the pin p is connected with the negative pin n2. Otherwise, the pin p is connected to the negative pin n1.

In order to prevent singularities during switching, the opened switch has a (very low) conductance *Goff* and the closed switch has a (very low) resistance *Ron*. The limiting case is also allowed, i.e., the resistance *Ron* of the closed switch could be exactly zero and the conductance *Goff* of the open switch could be also exactly zero. Note, there are circuits, where a description with zero *Ron* or zero *Goff* is not possible.

## Parameters

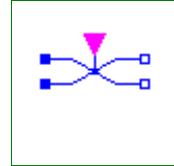
Name	Default	Description
Ron	1.E-5	Closed switch resistance [Ohm]
Goff	1.E-5	Opened switch conductance [S]

## Connectors

Name	Description
p	
n2	
n1	
control	true => p--n2 connected, false => p--n1 connected

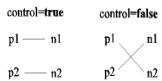
## Modelica.Electrical.Analog.Ideal.IdealIntermediateSwitch

Ideal intermediate switch

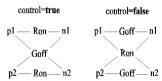


## Information

The intermediate switch has four switching contact pins p1, p2, n1, and n2. The switching behaviour is controlled by the input signal control. If control is true, the pin p1 is connected to pin n2, and the pin p2 is connected to the pin n2. Otherwise, the pin p1 is connected to n1, and p2 is connected to n2.



In order to prevent singularities during switching, the opened switch has a (very low) conductance Goff and the closed switch has a (very low) resistance Ron.



The limiting case is also allowed, i.e., the resistance Ron of the closed switch could be exactly zero and the conductance Goff of the open switch could be also exactly zero. Note, there are circuits, where a description with zero Ron or zero Goff is not possible.

## Parameters

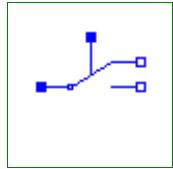
Name	Default	Description
Ron	1.E-5	Closed switch resistance [Ohm]
Goff	1.E-5	Opened switch conductance [S]

## Connectors

Name	Description
p1	
p2	
n1	
n2	
control	true => p1--n2, p2--n1 connected, otherwise p1--n1, p2--n2 connected

## Modelica.Electrical.Analog.Ideal.ControlledIdealCommutingSwitch

### Controlled ideal commuting switch



### Information

The commuting switch has a positive pin p and two negative pins n1 and n2. The switching behaviour is controlled by the control pin. If its voltage exceeds the value of the parameter level, the pin p is connected with the negative pin n2. Otherwise, the pin p is connected the negative pin n1.

In order to prevent singularities during switching, the opened switch has a (very low) conductance Goff and the closed switch has a (very low) resistance Ron. The limiting case is also allowed, i.e., the resistance Ron of the closed switch could be exactly zero and the conductance Goff of the open switch could be also exactly zero. Note, there are circuits, where a description with zero Ron or zero Goff is not possible.

### Parameters

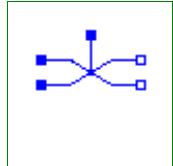
Name	Default	Description
level	0.5	Switch level [V]
Ron	1.E-5	Closed switch resistance [Ohm]
Goff	1.E-5	Opened switch conductance [S]

### Connectors

Name	Description
p	
n2	
n1	
control	Control pin: if control.v > level p--n2 connected, otherwise p--n1 connected

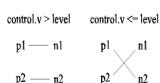
## Modelica.Electrical.Analog.Ideal.ControlledIdealIntermediateSwitch

### Controlled ideal intermediate switch

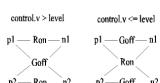


### Information

The intermediate switch has four switching contact pins p1, p2, n1, and n2. The switching behaviour is controlled by the control pin. If its voltage exceeds the value of the parameter level, the pin p1 is connected to pin n2, and the pin p2 is connected to the pin n2. Otherwise, the pin p1 is connected to n1, and p2 is connected to n2.



In order to prevent singularities during switching, the opened switch has a (very low) conductance Goff and the closed switch has a (very low) resistance Ron.



The limiting case is also allowed, i.e., the resistance Ron of the closed switch could be exactly zero and the conductance Goff of the open switch could be also exactly zero. Note, there are circuits, where a description with zero Ron or zero Goff is not possible.

## Parameters

Name	Default	Description
level	0.5	Switch level [V]
Ron	1.E-5	Closed switch resistance [Ohm]
Goff	1.E-5	Opened switch conductance [S]

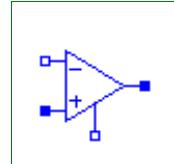
## Connectors

Name	Description
p1	
p2	
n1	
n2	
control	Control pin: if control.v > level p1--n2, p2--n1 connected, otherwise p1--n1, p2--n2 connected

---

## Modelica.Electrical.Analog.Ideal.IdealOpAmp

Ideal operational amplifier (norator-nullator pair)



## Information

The ideal OpAmp is a two-port. The left port is fixed to  $v1=0$  and  $i1=0$  (nullator). At the right port both any voltage  $v2$  and any current  $i2$  are possible (norator).

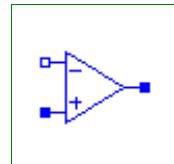
## Connectors

Name	Description
p1	Positive pin of the left port
n1	Negative pin of the left port
p2	Positive pin of the right port
n2	Negative pin of the right port

---

## Modelica.Electrical.Analog.Ideal.IdealOpAmp3Pin

Ideal operational amplifier (norator-nullator pair), but 3 pins



## Information

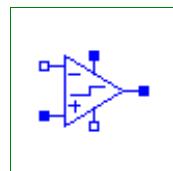
The ideal OpAmp with three pins is of exactly the same behaviour as the ideal OpAmp with four pins. Only the negative output pin is left out. Both the input voltage and current are fixed to zero (nullator). At the output pin both any voltage  $v2$  and any current  $i2$  are possible.

## Connectors

Name	Description
in_p	Positive pin of the input port
in_n	Negative pin of the input port
out	Output pin

## Modelica.Electrical.Analog.Ideal.IdealOpAmpLimited

Ideal operational amplifier with limitation



### Information

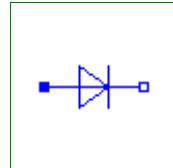
The ideal OpAmp with limitation behaves like an ideal OpAmp without limitation, if the output voltage is within the limits ( $VMin < out.v < VMax$ ). In this case the input voltage  $vin = in\_p.v - in\_n.v$  is zero. If the input voltage is  $vin < 0$ , the output voltage is  $out.v = VMin$ . If the input voltage is  $vin > 0$ , the output voltage is  $out.v = VMax$ .

### Connectors

Name	Description
in_p	Positive pin of the input port
in_n	Negative pin of the input port
out	Output pin
VMax	Positive output voltage limitation
VMin	Negative output voltage limitation

## Modelica.Electrical.Analog.Ideal.IdealDiode

Ideal diode



### Information

This is an ideal switch which is

**open** (off), if it is reversed biased (voltage drop less than 0)  
**closed** (on), if it is conducting (current > 0).

This is the behaviour if all parameters are exactly zero.

Note, there are circuits, where this ideal description with zero resistance and zero conductance is not possible. In order to prevent singularities during switching, the opened diode has a small conductance  $Gon$  and the closed diode has a low resistance  $Roff$  which is default.

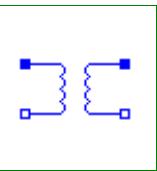
The parameter  $Vknee$  which is the forward threshold voltage, allows to displace the knee point along the  $Gon$ -characteristic until  $v = Vknee$ .

### Parameters

Name	Default	Description
Ron	1.E-5	Forward state-on differential resistance (closed diode resistance) [Ohm]
Goff	1.E-5	Backward state-off conductance (opened diode conductance) [S]
Vknee	0	Forward threshold voltage [V]

### Connectors

Name	Description
p	Positive pin (potential p.v > n.v for positive voltage drop v)
n	Negative pin

**Modelica.Electrical.Analog.Ideal.IdealTransformer****Ideal electrical transformer****Information**

The ideal transformer is an ideal two-port resistive circuit element which is characterized by the following two equations:

$$\begin{aligned} v1 &= n * v2 \\ i2 &= -n * i1 \end{aligned}$$

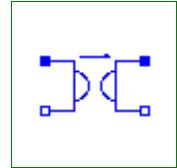
where  $n$  is a real number called the turns ratio.

**Parameters**

Name	Default	Description
n	1	Turns ratio

**Connectors**

Name	Description
p1	Positive pin of the left port (potential p1.v > n1.v for positive voltage drop v1)
n1	Negative pin of the left port
p2	Positive pin of the right port (potential p2.v > n2.v for positive voltage drop v2)
n2	Negative pin of the right port

**Modelica.Electrical.Analog.Ideal.IdealGyrator****Ideal gyrator****Information**

A gyrator is an ideal two-port element defined by the following equations:

$$\begin{aligned} i1 &= G * v2 \\ i2 &= -G * v1 \end{aligned}$$

where the constant  $G$  is called the gyration conductance.

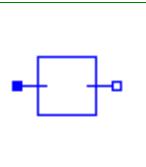
**Parameters**

Name	Default	Description
G	1	Gyration conductance [S]

**Connectors**

Name	Description
p1	Positive pin of the left port (potential p1.v > n1.v for positive voltage drop v1)
n1	Negative pin of the left port
p2	Positive pin of the right port (potential p2.v > n2.v for positive voltage drop v2)

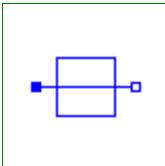
n2	Negative pin of the right port
----	--------------------------------

**Modelica.Electrical.Analog.Ideal.Idle****Idle branch****Information**

The model Idle is a simple idle running branch.

**Connectors**

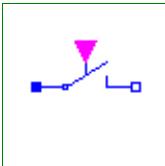
Name	Description
p	Positive pin (potential p.v > n.v for positive voltage drop v)
n	Negative pin

**Modelica.Electrical.Analog.Ideal.Short****Short cut branch****Information**

The model Short is a simple short cut branch.

**Connectors**

Name	Description
p	Positive pin (potential p.v > n.v for positive voltage drop v)
n	Negative pin

**Modelica.Electrical.Analog.Ideal.IdealOpeningSwitch****Ideal electrical opener****Information**

The ideal opening switch has a positive pin p and a negative pin n. The switching behaviour is controlled by the input signal control. If control is true, pin p is not connected with negative pin n. Otherwise, pin p is connected with negative pin n.

In order to prevent singularities during switching, the opened switch has a (very low) conductance Goff and the closed switch has a (very low) resistance Ron. The limiting case is also allowed, i.e., the resistance Ron of the closed switch could be exactly zero and the conductance Goff of the open switch could be also exactly zero. Note, there are circuits, where a description with zero Ron or zero Goff is not possible.

**Parameters**

Name	Default	Description
Ron	1.E-5	Closed switch resistance [Ohm]

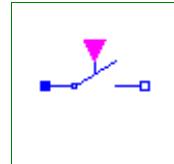
Goff	1.E-5	Opened switch conductance [S]
------	-------	-------------------------------

## Connectors

Name	Description
p	Positive pin (potential p.v > n.v for positive voltage drop v)
n	Negative pin
control	true => switch open, false => p--n connected

## Modelica.Electrical.Analog.Ideal.IdealClosingSwitch

Ideal electrical closer



## Information

The ideal closing switch has a positive pin p and a negative pin n. The switching behaviour is controlled by input signal control. If control is true, pin p is connected with negative pin n. Otherwise, pin p is not connected with negative pin n.

In order to prevent singularities during switching, the opened switch has a (very low) conductance Goff and the closed switch has a (very low) resistance Ron. The limiting case is also allowed, i.e., the resistance Ron of the closed switch could be exactly zero and the conductance Goff of the open switch could be also exactly zero. Note, there are circuits, where a description with zero Ron or zero Goff is not possible.

## Parameters

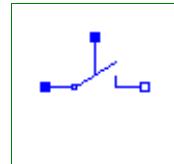
Name	Default	Description
Ron	1.E-5	Closed switch resistance [Ohm]
Goff	1.E-5	Opened switch conductance [S]

## Connectors

Name	Description
p	Positive pin (potential p.v > n.v for positive voltage drop v)
n	Negative pin
control	true => p--n connected, false => switch open

## Modelica.Electrical.Analog.Ideal.ControlledIdealOpeningSwitch

Controlled ideal electrical opener



## Information

The ideal switch has a positive pin p and a negative pin n. The switching behaviour is controlled by the control pin. If its voltage exceeds the voltage of the parameter level, pin p is not connected with negative pin n. Otherwise, pin p is connected with negative pin n.

In order to prevent singularities during switching, the opened switch has a (very low) conductance Goff and the closed switch has a (very low) resistance Ron. The limiting case is also allowed, i.e., the resistance Ron of the closed switch could be exactly zero and the conductance Goff of the open switch could be also exactly zero. Note, there are circuits, where a description with zero Ron or zero Goff is not possible.

## Parameters

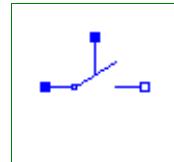
Name	Default	Description
level	0.5	Switch level [V]
Ron	1.E-5	Closed switch resistance [Ohm]
Goff	1.E-5	Opened switch conductance [S]

## Connectors

Name	Description
p	
n	
control	Control pin: control.v > level switch open, otherwise p-n connected

## Modelica.Electrical.Analog.Ideal.ControlledIdealClosingSwitch

Controlled ideal electrical closer



## Information

The closing ideal switch has a positive pin p and a negative pin n. The switching behaviour is controlled by the control pin. If its voltage exceeds the voltage of the parameter level, pin p is connected with negative pin n. Otherwise, pin p is not connected with negative pin n.

In order to prevent singularities during switching, the opened switch has a (very low) conductance Goff and the closed switch has a (very low) resistance Ron. The limiting case is also allowed, i.e., the resistance Ron of the closed switch could be exactly zero and the conductance Goff of the open switch could be also exactly zero. Note, there are circuits, where a description with zero Ron or zero Goff is not possible.

## Parameters

Name	Default	Description
level	0.5	Switch level [V]
Ron	1.E-5	Closed switch resistance [Ohm]
Goff	1.E-5	Opened switch conductance [S]

## Connectors

Name	Description
p	
n	
control	Control pin: control.v > level switch closed, otherwise switch open

## Modelica.Electrical.Analog.Interfaces

Connectors and partial models for Analog electrical components

## Information

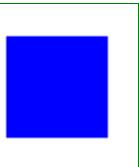
This package contains connectors and interfaces (partial models) for analog electrical components.

## Package Content

Name	Description
 Pin	Pin of an electrical component
 PositivePin	Positive pin of an electric component
 NegativePin	Negative pin of an electric component
 TwoPin	Component with one electrical port
 OnePort	Component with two electrical pins p and n and current i from p to n
 TwoPort	Component with two electrical ports, including current
 AbsoluteSensor	Base class to measure the absolute value of a pin variable
 RelativeSensor	Base class to measure a relative variable between two pins
 VoltageSource	Interface for voltage sources
 CurrentSource	Interface for current sources

## Modelica.Electrical.Analog.Interfaces.Pin

Pin of an electrical component

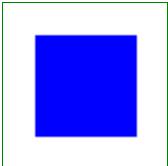


### Contents

Name	Description
v	Potential at the pin [V]
i	Current flowing into the pin [A]

## Modelica.Electrical.Analog.Interfaces.PositivePin

Positive pin of an electric component



### Information

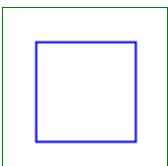
Connectors PositivePin and NegativePin are nearly identical. The only difference is that the icons are different in order to identify more easily the pins of a component. Usually, connector PositivePin is used for the positive and connector NegativePin for the negative pin of an electrical component.

### Contents

Name	Description
v	Potential at the pin [V]
i	Current flowing into the pin [A]

## Modelica.Electrical.Analog.Interfaces.NegativePin

Negative pin of an electric component



### Information

Connectors PositivePin and NegativePin are nearly identical. The only difference is that the icons are

## 228 Modelica.Electrical.Analog.Interfaces.NegativePin

---

different in order to identify more easily the pins of a component. Usually, connector PositivePin is used for the positive and connector NegativePin for the negative pin of an electrical component.

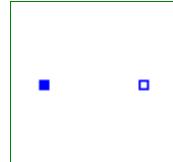
### Contents

Name	Description
v	Potential at the pin [V]
i	Current flowing into the pin [A]

---

## Modelica.Electrical.Analog.Interfaces.TwoPin

Component with one electrical port



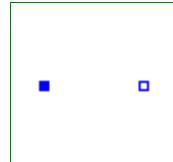
### Connectors

Name	Description
p	Positive pin Positive pin (potential p.v > n.v for positive voltage drop v)
n	Negative pin

---

## Modelica.Electrical.Analog.Interfaces.OnePort

Component with two electrical pins p and n and current i from p to n



### Information

Superclass of elements which have **two** electrical pins: the positive pin connector *p*, and the negative pin connector *n*. It is assumed that the current flowing into pin *p* is identical to the current flowing out of pin *n*. This current is provided explicitly as current *i*.

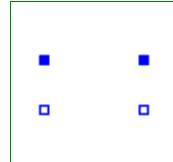
### Connectors

Name	Description
p	Positive pin (potential p.v > n.v for positive voltage drop v)
n	Negative pin

---

## Modelica.Electrical.Analog.Interfaces.TwoPort

Component with two electrical ports, including current



### Information

### Connectors

Name	Description
p1	Positive pin of the left port (potential p1.v > n1.v for positive voltage drop v1)
n1	Negative pin of the left port
p2	Positive pin of the right port (potential p2.v > n2.v for positive voltage drop v2)

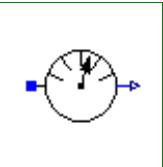
---

---

n2	Negative pin of the right port
----	--------------------------------

**Modelica.Electrical.Analog.Interfaces.AbsoluteSensor**

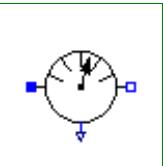
Base class to measure the absolute value of a pin variable

**Connectors**

Name	Description
p	Pin to be measured
y	Measured quantity as Real output signal

**Modelica.Electrical.Analog.Interfaces.RelativeSensor**

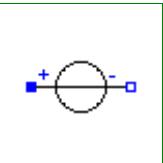
Base class to measure a relative variable between two pins

**Connectors**

Name	Description
p	Positive pin
n	Negative pin
y	Measured quantity as Real output signal

**Modelica.Electrical.Analog.Interfaces.VoltageSource**

Interface for voltage sources

**Parameters**

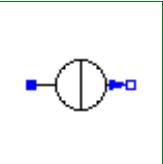
Name	Default	Description
offset	0	Voltage offset [V]
startTime	0	Time offset [s]
signalSource	redeclare Modelica.Blocks.In...	

**Connectors**

Name	Description
p	Positive pin (potential p.v > n.v for positive voltage drop v)
n	Negative pin

**Modelica.Electrical.Analog.Interfaces.CurrentSource**

Interface for current sources

**Parameters**

Name	Default	Description
offset	0	Current offset [A]

## 230 Modelica.Electrical.Analog.Interfaces.CurrentSource

---

startTime	0	Time offset [s]
signalSource	redeclare Modelica.Blocks.In...	

### Connectors

Name	Description
p	Positive pin (potential p.v > n.v for positive voltage drop v)
n	Negative pin

---

## Modelica.Electrical.Analog.Lines

### Lossy and lossless segmented transmission lines, and LC distributed line models

#### Information

This package contains lossy and lossless segmented transmission lines, and LC distributed line models.

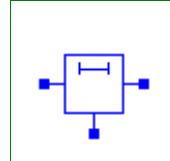
#### Package Content

Name	Description
 OLine	Lossy Transmission Line
 ULine	Lossy RC Line
 TLine1	Lossless transmission line with characteristic impedance Z0 and transmission delay TD
 TLine2	Lossless transmission line with characteristic impedance Z0, frequency F and normalized length NL
 TLine3	Lossless transmission line with characteristic impedance Z0 and frequency F

---

## Modelica.Electrical.Analog.Lines.OLine

### Lossy Transmission Line



#### Information

Lossy Transmission Line. The lossy transmission line OLine consists of segments of lumped resistances and inductances in series and conductances and capacitances that are connected with the reference pin p3. The precision of the model depends on the number N of lumped segments.

#### References:

Johnson, B.; Quarles, T.; Newton, A. R.; Pederson, D. O.; Sangiovanni-Vincentelli, A.: SPICE3 Version 3e User's Manual (April 1, 1991). Department of Electrical Engineering and Computer Sciences, University of California, Berkley p. 12, p. 106 - 107

#### Parameters

Name	Default	Description
r	1	Resistance per meter [Ohm/m]
l	1	Inductance per meter [H/m]
g	1	Conductance per meter [S/m]
c	1	Capacitance per meter [F/m]

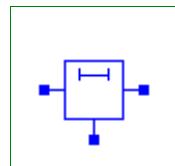
length	1	Length of line [m]
N	1	Number of lumped segments

## Connectors

Name	Description
p1	
p2	
p3	

## Modelica.Electrical.Analog.Lines.ULine

Lossy RC Line



## Information

The lossy RC line ULine consists of segments of lumped series resistances and capacitances that are connected with the reference pin p3. The precision of the model depends on the number N of lumped segments.

## References

Johnson, B.; Quarles, T.; Newton, A. R.; Pederson, D. O.; Sangiovanni-Vincentelli, A.  
SPICE3 Version 3e User's Manual (April 1, 1991). Department of Electrical Engineering and Computer Sciences, University of California, Berkley p. 22, p. 124

## Parameters

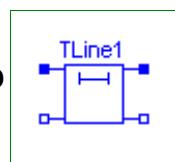
Name	Default	Description
r	1	Resistance per meter [Ohm/m]
c	1	Capacitance per meter [F/m]
length	1	Length of line [m]
N	1	Number of lumped segments

## Connectors

Name	Description
p1	
p2	
p3	

## Modelica.Electrical.Analog.Lines.TLine1

Lossless transmission line with characteristic impedance Z0 and transmission delay TD



## Information

Lossless transmission line with characteristic impedance Z0 and transmission delay TD The lossless transmission line TLine1 is a two Port. Both port branches consist of a resistor with characteristic impedance Z0 and a controlled voltage source that takes into consideration the transmission delay TD. For further details see Branin's article below. The model parameters can be derived from inductance and capacitance per length (L' resp. C'), i. e.  $Z_0 = \sqrt{L'/C'}$  and  $TD = \sqrt{L'*C'} * \text{length\_of\_line}$ . Resistance R' and conductance

## 232 Modelica.Electrical.Analog.Lines.TLine1

---

C' per meter are assumed to be zero.

### References:

Branin Jr., F. H.

Transient Analysis of Lossless Transmission Lines. Proceedings of the IEEE 55(1967), 2012 - 2013

Hoefer, E. E. E.; Nielinger, H.

SPICE : Analyseprogramm fuer elektronische Schaltungen. Springer-Verlag, Berlin, Heidelberg, New York, Tokyo, 1985.

### Parameters

Name	Default	Description
Z0	1	Characteristic impedance [Ohm]
TD	1	Transmission delay [s]

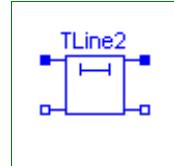
### Connectors

Name	Description
p1	Positive pin of the left port (potential p1.v > n1.v for positive voltage drop v1)
n1	Negative pin of the left port
p2	Positive pin of the right port (potential p2.v > n2.v for positive voltage drop v2)
n2	Negative pin of the right port

---

## Modelica.Electrical.Analog.Lines.TLine2

Lossless transmission line with characteristic impedance Z0, frequency F and normalized length NL



### Information

Lossless transmission line with characteristic impedance Z0, frequency F and normalized length NL. The lossless transmission line TLine2 is a two Port. Both port branches consist of a resistor with the value of the characteristic impedance Z0 and a controlled voltage source that takes into consideration the transmission delay. For further details see Branin's article below. Resistance R' and conductance C' per meter are assumed to be zero. The characteristic impedance Z0 can be derived from inductance and capacitance per length (L' resp. C'), i. e.  $Z_0 = \sqrt{L'/C'}$ . The normalized length NL is equal to the length of the line divided by the wavelength corresponding to the frequency F, i. e. the transmission delay TD is the quotient of NL and F.

### References:

Branin Jr., F. H.

Transient Analysis of Lossless Transmission Lines. Proceedings of the IEEE 55(1967), 2012 - 2013

Hoefer, E. E. E.; Nielinger, H.

SPICE : Analyseprogramm fuer elektronische Schaltungen. Springer-Verlag, Berlin, Heidelberg, New York, Tokyo, 1985.

### Parameters

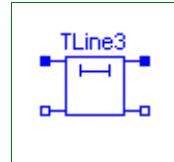
Name	Default	Description
Z0	1	Characteristic impedance [Ohm]
F	1	Frequency [Hz]
NL	1	Normalized length

## Connectors

Name	Description
p1	Positive pin of the left port (potential p1.v > n1.v for positive voltage drop v1)
n1	Negative pin of the left port
p2	Positive pin of the right port (potential p2.v > n2.v for positive voltage drop v2)
n2	Negative pin of the right port

## Modelica.Electrical.Analog.Lines.TLine3

Lossless transmission line with characteristic impedance Z0 and frequency F



## Information

Lossless transmission line with characteristic impedance Z0 and frequency F. The lossless transmission line TLine3 is a two Port. Both port branches consist of a resistor with value of the characteristic impedance Z0 and a controled voltage source that takes into consideration the transmission delay. For further details see Branin's article below. Resistance R' and conductance C' per meter are assumed to be zero. The characteristic impedance Z0 can be derived from inductance and capacitance per length ( $L'$  resp.  $C'$ ), i. e.  $Z_0 = \sqrt{L'/C'}$ . The length of the line is equal to a quarter of the wavelength corresponding to the frequency F, i. e. the transmission delay is the quotient of 4 and F. In this case, the characteristic impedance is called natural impedance.

## References:

Branin Jr., F. H.

Transient Analysis of Lossless Transmission Lines. Proceedings of the IEEE 55(1967), 2012 - 2013

Hoefer, E. E. E.; Nielinger, H.

SPICE : Analyseprogramm fuer elektronische Schaltungen. Springer-Verlag, Berlin, Heidelberg, New York, Tokyo, 1985.

## Parameters

Name	Default	Description
Z0	1	Natural impedance [Ohm]
F	1	Frequency [Hz]

## Connectors

Name	Description
p1	Positive pin of the left port (potential p1.v > n1.v for positive voltage drop v1)
n1	Negative pin of the left port
p2	Positive pin of the right port (potential p2.v > n2.v for positive voltage drop v2)
n2	Negative pin of the right port

## Modelica.Electrical.Analog.Semiconductors

Semiconductor devices such as diode, MOS and bipolar transistor

## Information

This package contains semiconductor devices:

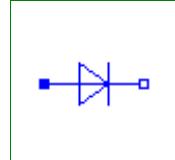
- diode
- MOS transistors
- bipolar transistors
- diode, MOS and bipolar transistors with temperature dependent characteristic and a heatPort for connection to the thermal domain

## Package Content

Name	Description
Diode	Simple diode
PMOS	Simple MOS Transistor
NMOS	Simple MOS Transistor
NPN	Simple BJT according to Ebers-Moll
PNP	Simple BJT according to Ebers-Moll
HeatingDiode	Simple diode with heating port
HeatingNMOS	Simple MOS Transistor with heating port
HeatingPMOS	Simple PMOS Transistor with heating port
HeatingNPN	Simple NPN BJT according to Ebers-Moll with heating port
HeatingPNP	Simple PNP BJT according to Ebers-Moll with heating port

## Modelica.Electrical.Analog.Semiconductors.Diode

### Simple diode



#### Information

The simple diode is a one port. It consists of the diode itself and an parallel ohmic resistance  $R$ . The diode formula is:

$$i = i_{ds} \left( e^{\frac{v}{Vt}} - 1 \right).$$

If the exponent  $v/Vt$  reaches the limit *maxex*, the diode characteristic is linearly continued to avoid overflow.

## Parameters

Name	Default	Description
Ids	1.e-6	Saturation current [A]
Vt	0.04	Voltage equivalent of temperature ( $kT/qn$ ) [V]
Maxexp	15	Max. exponent for linear continuation
R	1.e8	Parallel ohmic resistance [Ohm]

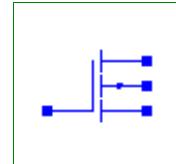
## Connectors

Name	Description
p	Positive pin (potential p.v > n.v for positive voltage drop)

	v)
n	Negative pin

## Modelica.Electrical.Analog.Semiconductors.PMOS

### Simple MOS Transistor



#### Information

The PMOS model is a simple model of a p-channel metal-oxide semiconductor FET. It differs slightly from the device used in the SPICE simulator. For more details please care for H. Spiro.

The model does not consider capacitances. A high drain-source resistance RDS is included to avoid numerical difficulties.

#### References:

Spiro, H.: Simulation integrierter Schaltungen. R. Oldenbourg Verlag Muenchen Wien 1990.

Some typical parameter sets are:

W m	L m	Beta A/V^2	Vt V	K2 -	K5 -	DW m	DL m
50.e-6	8.e-6	.0085e-3	-.15	.41	.839	-3.8e-6	-4.0e-6
20.e-6	6.e-6	.0105e-3	-1.0	.41	.839	-2.5e-6	-2.1e-6
30.e-6	5.e-6	.0059e-3	-.3	.98	1.01	0	-3.9e-6
30.e-6	5.e-6	.0152e-3	-.69	.104	1.1	-.8e-6	-.4e-6
30.e-6	5.e-6	.0163e-3	-.69	.104	1.1	-.8e-6	-.4e-6
30.e-6	5.e-6	.0182e-3	-.69	.086	1.06	-.1e-6	-.6e-6
20.e-6	6.e-6	.0074e-3	-1.	.4	.59	0	0

#### Parameters

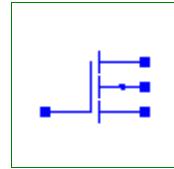
Name	Default	Description
W	20.0e-6	Width [m]
L	6.0e-6	Length [m]
Beta	0.0105e-3	Transconductance parameter [A/V2]
Vt	-1.0	Zero bias threshold voltage [V]
K2	0.41	Bulk threshold parameter
K5	0.839	Reduction of pinch-off region
dW	-2.5e-6	Narrowing of channel [m]
DL	-2.1e-6	Shortening of channel [m]
RDS	1.e+7	Drain-Source-Resistance [Ohm]

#### Connectors

Name	Description
D	Drain
G	Gate
S	Source
B	Bulk

## Modelica.Electrical.Analog.Semiconductors.NMOS

### Simple MOS Transistor



### Information

The NMos model is a simple model of a n-channel metal-oxide semiconductor FET. It differs slightly from the device used in the SPICE simulator. For more details please care for H. Spiro.

The model does not consider capacitances. A high drain-source resistance RDS is included to avoid numerical difficulties.

W m	L m	Beta A/V^2	Vt V	K2	K5	DW m	DL m
12.e-6	4.e-6	.062e-3	-4.5	.24	.61	-1.2e-6	-.9e-6
depletion							
60.e-6	3.e-6	.048e-3	.1	.08	.68	-1.2e-6	-.9e-6
enhancement							
12.e-6	4.e-6	.0625e-3	-.8	.21	.78	-1.2e-6	-.9e-6
zero							
50.e-6	8.e-6	.0299e-3	.24	1.144	.7311	-5.4e-6	-4.e-6
20.e-6	6.e-6	.041e-3	.8	1.144	.7311	-2.5e-6	-1.5e-6
30.e-6	9.e-6	.025e-3	-4.	.861	.878	-3.4e-6	-1.74e-6
30.e-6	5.e-6	.031e-3	.6	1.5	.72	0	-3.9e-6
50.e-6	6.e-6	.0414e-3	-3.8	.34	.8	-1.6e-6	-2.e-6
depletion							
50.e-6	5.e-6	.03e-3	.37	.23	.86	-1.6e-6	-2.e-6
enhancement							
50.e-6	6.e-6	.038e-3	-.9	.23	.707	-1.6e-6	-2.e-6
zero							
20.e-6	4.e-6	.06776e-3	.5409	.065	.71	-.8e-6	-.2e-6
20.e-6	4.e-6	.06505e-3	.6209	.065	.71	-.8e-6	-.2e-6
20.e-6	4.e-6	.05365e-3	.6909	.03	.8	-.3e-6	-.2e-6
20.e-6	4.e-6	.05365e-3	.4909	.03	.8	-.3e-6	-.2e-6
12.e-6	4.e-6	.023e-3	-4.5	.29	.6	0	0
depletion							
60.e-6	3.e-6	.022e-3	.1	.11	.65	0	0
enhancement							
12.e-6	4.e-6	.038e-3	-.8	.33	.6	0	0
zero							
20.e-6	6.e-6	.022e-3	.8	1	.66	0	0

### References:

Spiro, H.: Simulation integrierter Schaltungen. R. Oldenbourg Verlag Muenchen Wien 1990.

### Parameters

Name	Default	Description
W	20.e-6	Width [m]
L	6.e-6	Length [m]
Beta	0.041e-3	Transconductance parameter [A/V2]
Vt	0.8	Zero bias threshold voltage [V]
K2	1.144	Bulk threshold parameter
K5	0.7311	Reduction of pinch-off region
dW	-2.5e-6	narrowing of channel [m]
DL	-1.5e-6	shortening of channel [m]

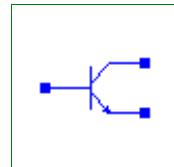
RDS	1.e+7	Drain-Source-Resistance [Ohm]
-----	-------	-------------------------------

## Connectors

Name	Description
D	Drain
G	Gate
S	Source
B	Bulk

## Modelica.Electrical.Analog.Semiconductors.NPN

### Simple BJT according to Ebers-Moll



## Information

This model is a simple model of a bipolar npn junction transistor according to Ebers-Moll.

A typical parameter set is:

Bf	Br	Is	Vak	Tauf	Taur	Ccs	Cje	Cjc	Phie	Me	PHic	Mc
Gbc	Gbe		Vt									
-	-	A	V	s	s	F	F	F	V	-	V	-
mS	mS		V									
50	0.1	1e-16	0.02	0.12e-9	5e-9	1e-12	0.4e-12	0.5e-12	0.8	0.4	0.8	
0.333		1e-15	1e-15			0.02585						

## References:

Vlach, J.; Singal, K.: Computer methods for circuit analysis and design. Van Nostrand Reinhold, New York 1983 on page 317 ff.

## Parameters

Name	Default	Description
Bf	50	Forward beta
Br	0.1	Reverse beta
Is	1.e-16	Transport saturation current [A]
Vak	0.02	Early voltage (inverse), 1/Volt [1/V]
Tauf	0.12e-9	Ideal forward transit time [s]
Taur	5e-9	Ideal reverse transit time [s]
Ccs	1e-12	Collector-substrat(ground) cap. [F]
Cje	0.4e-12	Base-emitter zero bias depletion cap. [F]
Cjc	0.5e-12	Base-coll. zero bias depletion cap. [F]
Phie	0.8	Base-emitter diffusion voltage [V]
Me	0.4	Base-emitter gradation exponent
Phic	0.8	Base-collector diffusion voltage [V]
Mc	0.333	Base-collector gradation exponent
Gbc	1e-15	Base-collector conductance [S]
Gbe	1e-15	Base-emitter conductance [S]
Vt	0.02585	Voltage equivalent of temperature [V]
EMin	-100	if x < EMin, the exp(x) function is linearized

EMax	40	if $x > EMax$ , the $\exp(x)$ function is linearized
------	----	--

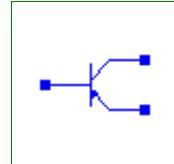
## Connectors

Name	Description
C	Collector
B	Base
E	Emitter

---

## Modelica.Electrical.Analog.Semiconductors.PNP

Simple BJT according to Ebers-Moll



## Information

This model is a simple model of a bipolar pnp junction transistor according to Ebers-Moll.

A typical parameter set is:

Bf	Br	Is	Vak	Tauf	Taur	Ccs	Cje	Cjc	Phie	Me	PHic	Mc
Gbc	Gbe		Vt									
-	-	A	V	s	s	F	F	F	V	-	V	-
mS	mS		V									
50	0.1	1e-16	0.02	0.12e-9	5e-9	1e-12	0.4e-12	0.5e-12	0.8	0.4	0.8	
0.333		1e-15	1e-15		0.02585							

## References:

Vlach, J.; Singal, K.: Computer methods for circuit analysis and design. Van Nostrand Reinhold, New York 1983 on page 317 ff.

## Parameters

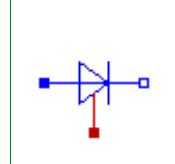
Name	Default	Description
Bf	50	Forward beta
Br	0.1	Reverse beta
Is	1.e-16	Transport saturation current [A]
Vak	0.02	Early voltage (inverse), 1/Volt [1/V]
Tauf	0.12e-9	Ideal forward transit time [s]
Taur	5e-9	Ideal reverse transit time [s]
Ccs	1e-12	Collector-substrat(ground) cap. [F]
Cje	0.4e-12	Base-emitter zero bias depletion cap. [F]
Cjc	0.5e-12	Base-col. zero bias depletion cap. [F]
Phie	0.8	Base-emitter diffusion voltage [V]
Me	0.4	Base-emitter gradation exponent
Phic	0.8	Base-collector diffusion voltage [V]
Mc	0.333	Base-collector gradation exponent
Gbc	1e-15	Base-collector conductance [S]
Gbe	1e-15	Base-emitter conductance [S]
Vt	0.02585	Voltage equivalent of temperature [V]
EMin	-100	if $x < EMin$ , the $\exp(x)$ function is linearized
EMax	40	if $x > EMax$ , the $\exp(x)$ function is linearized

## Connectors

Name	Description
C	Collector
B	Base
E	Emitter

## Modelica.Electrical.Analog.Semiconductors.HeatingDiode

Simple diode with heating port



## Information

The simple diode is an electrical one port, where a heat port is added, which is defined in the Modelica.Thermal library. It consists of the diode itself and an parallel ohmic resistance  $R$ . The diode formula is:

$$i = i_{ds} \left( e^{\frac{v}{vt_t}} - 1 \right).$$

where  $vt_t$  depends on the temperature of the heat port:

$$vt_t = k * \text{temp} / q$$

If the exponent  $v/vt_t$  reaches the limit  $\text{maxex}$ , the diode characteristic is linearly continued to avoid overflow. The thermal power is calculated by  $i*v$ .

## Parameters

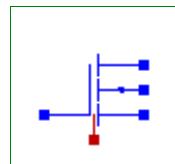
Name	Default	Description
Ids	1.e-6	Saturation current [A]
Maxexp	15	Max. exponent for linear continuation
R	1.e8	Parallel ohmic resistance [Ohm]
EG	1.11	activation energy
N	1	Emission coefficient
TNOM	300.15	Parameter measurement temperature [K]
XTI	3	Temperature exponent of saturation current

## Connectors

Name	Description
p	Positive pin (potential p.v > n.v for positive voltage drop v)
n	Negative pin
heatPort	

## Modelica.Electrical.Analog.Semiconductors.HeatingNMOS

Simple MOS Transistor with heating port



## Information

The NMos model is a simple model of a n-channel metal-oxide semiconductor FET. It differs slightly from the device used in the SPICE simulator. For more details please care for H. Spiro.

A heating port is added for thermal electric simulation. The heating port is defined in the Modelica.Thermal library.

The model does not consider capacitances. A high drain-source resistance RDS is included to avoid numerical difficulties.

W m	L m	Beta A/V^2	Vt V	K2 -	K5 -	DW m	DL m
12.e-6	4.e-6	.062e-3	-4.5	.24	.61	-1.2e-6	-.9e-6
depletion							
60.e-6	3.e-6	.048e-3	.1	.08	.68	-1.2e-6	-.9e-6
enhancement							
12.e-6	4.e-6	.0625e-3	-.8	.21	.78	-1.2e-6	-.9e-6
zero							
50.e-6	8.e-6	.0299e-3	.24	1.144	.7311	-5.4e-6	-.4e-6
20.e-6	6.e-6	.041e-3	.8	1.144	.7311	-2.5e-6	-1.5e-6
30.e-6	9.e-6	.025e-3	-4.	.861	.878	-3.4e-6	-1.74e-6
30.e-6	5.e-6	.031e-3	.6	1.5	.72	0	-3.9e-6
50.e-6	6.e-6	.0414e-3	-3.8	.34	.8	-1.6e-6	-2.e-6
depletion							
50.e-6	5.e-6	.03e-3	.37	.23	.86	-1.6e-6	-2.e-6
enhancement							
50.e-6	6.e-6	.038e-3	-.9	.23	.707	-1.6e-6	-2.e-6
zero							
20.e-6	4.e-6	.06776e-3	.5409	.065	.71	-.8e-6	-.2e-6
20.e-6	4.e-6	.06505e-3	.6209	.065	.71	-.8e-6	-.2e-6
20.e-6	4.e-6	.05365e-3	.6909	.03	.8	-.3e-6	-.2e-6
20.e-6	4.e-6	.05365e-3	.4909	.03	.8	-.3e-6	-.2e-6
12.e-6	4.e-6	.023e-3	-4.5	.29	.6	0	0
depletion							
60.e-6	3.e-6	.022e-3	.1	.11	.65	0	0
enhancement							
12.e-6	4.e-6	.038e-3	-.8	.33	.6	0	0
zero							
20.e-6	6.e-6	.022e-3	.8	1	.66	0	0

## References:

Spiro, H.: Simulation integrierter Schaltungen. R. Oldenbourg Verlag Muenchen Wien 1990.

## Parameters

Name	Default	Description
W	20.e-6	Width [m]
L	6.e-6	Length [m]
Beta	0.041e-3	Transconductance parameter [A/V2]
Vt	0.8	Zero bias threshold voltage [V]
K2	1.144	Bulk threshold parameter
K5	0.7311	Reduction of pinch-off region
dW	-2.5e-6	narrowing of channel [m]
dL	-1.5e-6	shortening of channel [m]
RDS	1.e+7	Drain-Source-Resistance [Ohm]

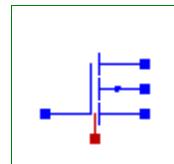
Tnom	300.15	Parameter measurement temperature [K]
kvt	-6.96e-3	fitting parameter for Vt
kk2	6.0e-4	fitting parameter for K22

## Connectors

Name	Description
D	Drain
G	Gate
S	Source
B	Bulk
heatPort	

## Modelica.Electrical.Analog.Semiconductors.HeatingPMOS

Simple PMOS Transistor with heating port



## Information

The PMOS model is a simple model of a p-channel metal-oxide semiconductor FET. It differs slightly from the device used in the SPICE simulator. For more details please care for H. Spiro.

A heating port is added for thermal electric simulation. The heating port is defined in the Modelica.Thermal library.

The model does not consider capacitances. A high drain-source resistance RDS is included to avoid numerical difficulties.

## References:

Spiro, H.: Simulation integrierter Schaltungen. R. Oldenbourg Verlag Muenchen Wien 1990.

Some typical parameter sets are:

W	L	Beta	Vt	K2	K5	dW	DL
m	m	A/V^2	V	-	-	m	m
50.e-6	8.e-6	.0085e-3	-.15	.41	.839	-3.8e-6	-4.0e-6
20.e-6	6.e-6	.0105e-3	-1.0	.41	.839	-2.5e-6	-2.1e-6
30.e-6	5.e-6	.0059e-3	-.3	.98	1.01	0	-3.9e-6
30.e-6	5.e-6	.0152e-3	-.69	.104	1.1	-.8e-6	-.4e-6
30.e-6	5.e-6	.0163e-3	-.69	.104	1.1	-.8e-6	-.4e-6
30.e-6	5.e-6	.0182e-3	-.69	.086	1.06	-.1e-6	-.6e-6
20.e-6	6.e-6	.0074e-3	-1.	.4	.59	0	0

## Parameters

Name	Default	Description
W	20.0e-6	Width [m]
L	6.0e-6	Length [m]
Beta	0.0105e-3	Transconductance parameter [A/V2]
Vt	-1.0	Zero bias threshold voltage [V]
K2	0.41	Bulk threshold parameter
K5	0.839	Reduction of pinch-off region
dW	-2.5e-6	Narrowing of channel [m]

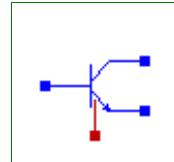
dL	-2.1e-6	Shortening of channel [m]
RDS	1.e+7	Drain-Source-Resistance [Ohm]
Tnom	300.15	Parameter measurement temperature [K]
kvt	-2.9e-3	fitting parameter for Vt
kk2	6.2e-4	fitting parameter for Kk2

## Connectors

Name	Description
D	Drain
G	Gate
S	Source
B	Bulk
heatPort	

## Modelica.Electrical.Analog.Semiconductors.HeatingNPN

Simple NPN BJT according to Ebers-Moll with heating port



## Information

This model is a simple model of a bipolar npn junction transistor according to Ebers-Moll.

A heating port is added for thermal electric simulation. The heating port is defined in the Modelica.Thermal library.

A typical parameter set is (the parameter Vt is no longer used):

Bf	Br	Is	Vak	Tauf	Taur	Ccs	Cje	Cjc	Phie	Me	PHic	Mc
Gbc	Gbe	-	A	V	s	s	F	F	V	-	V	-
mS	mS											
50	0.1	1e-16	0.02	0.12e-9	5e-9	1e-12	0.4e-12	0.5e-12	0.8	0.4	0.8	
0.333		1e-15	1e-15									

## References:

Vlach, J.; Singal, K.: Computer methods for circuit analysis and design. Van Nostrand Reinhold, New York 1983 on page 317 ff.

## Parameters

Name	Default	Description
Bf	50	Forward beta
Br	0.1	Reverse beta
Is	1.e-16	Transport saturation current [A]
Vak	0.02	Early voltage (inverse), 1/Volt [1/V]
Tauf	0.12e-9	Ideal forward transit time [s]
Taur	5e-9	Ideal reverse transit time [s]
Ccs	1e-12	Collector-substrat(ground) cap. [F]
Cje	0.4e-12	Base-emitter zero bias depletion cap. [F]
Cjc	0.5e-12	Base-coll. zero bias depletion cap. [F]
Phie	0.8	Base-emitter diffusion voltage [V]

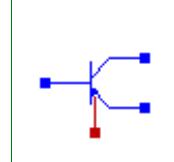
Me	0.4	Base-emitter gradation exponent
Phic	0.8	Base-collector diffusion voltage [V]
Mc	0.333	Base-collector gradation exponent
Gbc	1e-15	Base-collector conductance [S]
Gbe	1e-15	Base-emitter conductance [S]
EMin	-100	if $x < E_{\text{Min}}$ , the $\exp(x)$ function is linearized
EMax	40	if $x > E_{\text{Max}}$ , the $\exp(x)$ function is linearized
Tnom	300.15	Parameter measurement temperature [K]
XTI	3	Temperature exponent for effect on $I_s$
XTB	0	Forward and reverse beta temperature exponent
EG	1.11	Energy gap for temperature effect on $I_s$
NF	1.0	Forward current emission coefficient
NR	1.0	Reverse current emission coefficient
K	1.3806226e-23	Boltzmann's constant
q	1.6021918e-19	Elementary electronic charge

## Connectors

Name	Description
C	Collector
B	Base
E	Emitter
heatPort	

## Modelica.Electrical.Analog.Semiconductors.HeatingPnP

Simple PNP BJT according to Ebers-Moll with heating port



## Information

This model is a simple model of a bipolar pnp junction transistor according to Ebers-Moll.

A heating port is added for thermal electric simulation. The heating port is defined in the Modelica.Thermal library.

A typical parameter set is (the parameter  $V_t$  is no longer used):

Bf	Br	Is	Vak	Tauf	Taur	Ccs	Cje	Cjc	Phie	Me	PHic	Mc
Gbc	Gbe											
-	-	A	V	s	s	F	F	F	V	-	V	-
mS	mS											
50	0.1	1e-16	0.02	0.12e-9	5e-9	1e-12	0.4e-12	0.5e-12	0.8	0.4	0.8	
0.333	1e-15	1e-15										

## References:

Vlach, J.; Singal, K.: Computer methods for circuit analysis and design. Van Nostrand Reinhold, New York 1983 on page 317 ff.

## Parameters

Name	Default	Description
Bf	50	Forward beta

Br	0.1	Reverse beta
Is	1.e-16	Transport saturation current [A]
Vak	0.02	Early voltage (inverse), 1/Volt [1/V]
Tauf	0.12e-9	Ideal forward transit time [s]
Taur	5e-9	Ideal reverse transit time [s]
Ccs	1e-12	Collector-substrat(ground) cap. [F]
Cje	0.4e-12	Base-emitter zero bias depletion cap. [F]
Cjc	0.5e-12	Base-coll. zero bias depletion cap. [F]
Phie	0.8	Base-emitter diffusion voltage [V]
Me	0.4	Base-emitter gradation exponent
Phic	0.8	Base-collector diffusion voltage [V]
Mc	0.333	Base-collector gradation exponent
Gbc	1e-15	Base-collector conductance [S]
Gbe	1e-15	Base-emitter conductance [S]
EMin	-100	if $x < E_{Min}$ , the $\exp(x)$ function is linearized
EMax	40	if $x > E_{Max}$ , the $\exp(x)$ function is linearized
Tnom	300.15	Parameter measurement temperature [K]
XTI	3	Temperature exponent for effect on Is
XTB	0	Forward and reverse beta temperature exponent
EG	1.11	Energy gap for temperature effect on Is
NF	1.0	Forward current emission coefficient
NR	1.0	Reverse current emission coefficient
K	1.3806226e-23	Boltzmann's constant
q	1.6021918e-19	Elementary electronic charge

## Connectors

Name	Description
C	Collector
B	Base
E	Emitter
heatPort	

---

## Modelica.Electrical.Analog.Sensors

### Potential, voltage, current, and power sensors

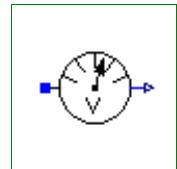
## Information

This package contains potential, voltage, and current sensors.

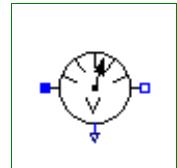
## Package Content

Name	Description
 PotentialSensor	Sensor to measure the potential
 VoltageSensor	Sensor to measure the voltage between two pins
 CurrentSensor	Sensor to measure the current in a branch

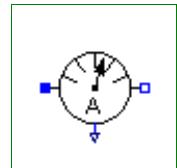
 PowerSensor	Sensor to measure the power
---	-----------------------------

**Modelica.Electrical.Analog.Sensors.PotentialSensor****Sensor to measure the potential****Connectors**

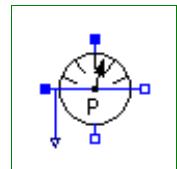
Name	Description
p	pin to be measured
phi	Absolute voltage potential as output signal

**Modelica.Electrical.Analog.Sensors.VoltageSensor****Sensor to measure the voltage between two pins****Connectors**

Name	Description
p	positive pin
n	negative pin
v	Voltage between pin p and n (= p.v - n.v) as output signal

**Modelica.Electrical.Analog.Sensors.CurrentSensor****Sensor to measure the current in a branch****Connectors**

Name	Description
p	positive pin
n	negative pin
i	current in the branch from p to n as output signal

**Modelica.Electrical.Analog.Sensors.PowerSensor****Sensor to measure the power****Information**

This power sensor measures instantaneous electrical power of a singlephase system and has a separated voltage and current path. The pins of the voltage path are  $p_v$  and  $n_v$ , the pins of the current path are  $p_c$  and  $n_c$ . The internal resistance of the current path is zero, the internal resistance of the voltage path is infinite.

**Connectors**

Name	Description
pc	Positive pin, current path

nc	Negative pin, current path
pv	Positive pin, voltage path
nv	Negative pin, voltage path
power	

---

## Modelica.Electrical.Analog.Sources

### Time-dependend and controlled voltage and current sources

#### Information

This package contains time-dependend and controlled voltage and current sources.

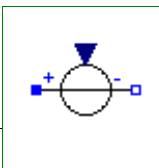
#### Package Content

Name	Description
 SignalVoltage	Generic voltage source using the input signal as source voltage
 ConstantVoltage	Source for constant voltage
 StepVoltage	Step voltage source
 RampVoltage	Ramp voltage source
 SineVoltage	Sine voltage source
 ExpSineVoltage	Exponentially damped sine voltage source
 ExponentialsVoltage	Rising and falling exponential voltage source
 PulseVoltage	Pulse voltage source
 SawToothVoltage	Saw tooth voltage source
 TrapezoidVoltage	Trapezoidal voltage source
 TableVoltage	Voltage source by linear interpolation in a table
 SignalCurrent	Generic current source using the input signal as source current
 ConstantCurrent	Source for constant current
 StepCurrent	Step current source
 RampCurrent	Ramp current source
 SineCurrent	Sine current source
 ExpSineCurrent	Exponentially damped sine current source
 ExponentialsCurrent	Rising and falling exponential current source
 PulseCurrent	Pulse current source
 SawToothCurrent	Saw tooth current source
 TrapezoidCurrent	Trapezoidal current source
 TableCurrent	Current source by linear interpolation in a table

---

## Modelica.Electrical.Analog.Sources.SignalVoltage

Generic voltage source using the input signal as source voltage

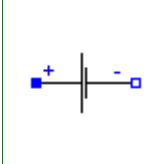


## Connectors

Name	Description
p	
n	
v	Voltage between pin p and n (= p.v - n.v) as input signal

## Modelica.Electrical.Analog.Sources.ConstantVoltage

Source for constant voltage



## Parameters

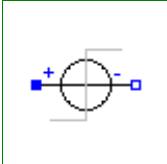
Name	Default	Description
V	1	Value of constant voltage [V]

## Connectors

Name	Description
p	Positive pin (potential p.v > n.v for positive voltage drop v)
n	Negative pin

## Modelica.Electrical.Analog.Sources.StepVoltage

Step voltage source



## Parameters

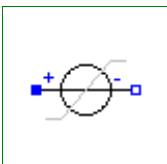
Name	Default	Description
V	1	Height of step [V]
offset	0	Voltage offset [V]
startTime	0	Time offset [s]

## Connectors

Name	Description
p	Positive pin (potential p.v > n.v for positive voltage drop v)
n	Negative pin

## Modelica.Electrical.Analog.Sources.RampVoltage

Ramp voltage source



## Parameters

Name	Default	Description
V	1	Height of ramp [V]
duration	2	Duration of ramp [s]

## 248 Modelica.Electrical.Analog.Sources.RampVoltage

---

offset	0	Voltage offset [V]
startTime	0	Time offset [s]

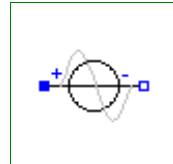
### Connectors

Name	Description
p	Positive pin (potential p.v > n.v for positive voltage drop v)
n	Negative pin

---

## Modelica.Electrical.Analog.Sources.SineVoltage

Sine voltage source



### Parameters

Name	Default	Description
V	1	Amplitude of sine wave [V]
phase	0	Phase of sine wave [rad]
freqHz	1	Frequency of sine wave [Hz]
offset	0	Voltage offset [V]
startTime	0	Time offset [s]

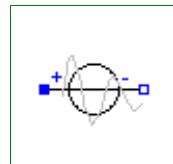
### Connectors

Name	Description
p	Positive pin (potential p.v > n.v for positive voltage drop v)
n	Negative pin

---

## Modelica.Electrical.Analog.Sources.ExpSineVoltage

Exponentially damped sine voltage source



### Parameters

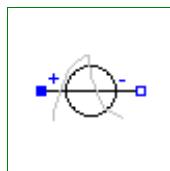
Name	Default	Description
V	1	Amplitude of sine wave [V]
freqHz	2	Frequency of sine wave [Hz]
phase	0	Phase of sine wave [rad]
damping	1	Damping coefficient of sine wave [s-1]
offset	0	Voltage offset [V]
startTime	0	Time offset [s]

### Connectors

Name	Description
p	Positive pin (potential p.v > n.v for positive voltage drop v)
n	Negative pin

**Modelica.Electrical.Analog.Sources.ExponentialsVoltage**

Rising and falling exponential voltage source

**Parameters**

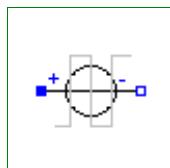
Name	Default	Description
vMax	1	Upper bound for rising edge
riseTime	0.5	Rise time [s]
riseTimeConst	0.1	Rise time constant [s]
fallTimeConst	riseTimeConst	Fall time constant [s]
offset	0	Voltage offset [V]
startTime	0	Time offset [s]

**Connectors**

Name	Description
p	Positive pin (potential p.v > n.v for positive voltage drop v)
n	Negative pin

**Modelica.Electrical.Analog.Sources.PulseVoltage**

Pulse voltage source

**Parameters**

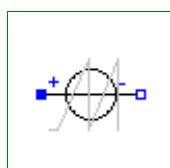
Name	Default	Description
V	1	Amplitude of pulse [V]
width	50	Width of pulse in % of period
period	1	Time for one period [s]
offset	0	Voltage offset [V]
startTime	0	Time offset [s]

**Connectors**

Name	Description
p	Positive pin (potential p.v > n.v for positive voltage drop v)
n	Negative pin

**Modelica.Electrical.Analog.Sources.SawToothVoltage**

Saw tooth voltage source

**Parameters**

Name	Default	Description
V	1	Amplitude of saw tooth [V]

## 250 Modelica.Electrical.Analog.Sources.SawToothVoltage

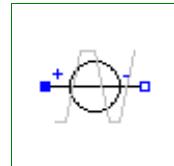
period	1	Time for one period [s]
offset	0	Voltage offset [V]
startTime	0	Time offset [s]

### Connectors

Name	Description
p	Positive pin (potential p.v > n.v for positive voltage drop v)
n	Negative pin

## Modelica.Electrical.Analog.Sources.TrapezoidVoltage

Trapezoidal voltage source



### Parameters

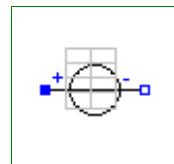
Name	Default	Description
V	1	Amplitude of trapezoid [V]
rising	0	Rising duration of trapezoid [s]
width	0.5	Width duration of trapezoid [s]
falling	0	Falling duration of trapezoid [s]
period	1	Time for one period [s]
nperiod	-1	Number of periods (< 0 means infinite number of periods)
offset	0	Voltage offset [V]
startTime	0	Time offset [s]

### Connectors

Name	Description
p	Positive pin (potential p.v > n.v for positive voltage drop v)
n	Negative pin

## Modelica.Electrical.Analog.Sources.TableVoltage

Voltage source by linear interpolation in a table



### Information

This block generates a voltage source by **linear interpolation** in a table. The time points and voltage values are stored in a matrix **table[i,j]**, where the first column **table[:,1]** contains the time points and the second column contains the voltage to be interpolated. The table interpolation has the following properties:

- The time points need to be **monotonically increasing**.
- **Discontinuities** are allowed, by providing the same time point twice in the table.
- Values **outside** of the table range, are computed by **extrapolation** through the last or first two points of the table.
- If the table has only **one row**, no interpolation is performed and the voltage value is just returned independantly of the actual time instant, i.e., this is a constant voltage source.
- Via parameters **startTime** and **offset** the curve defined by the table can be shifted both in time and in the voltage.

- The table is implemented in a numerically sound way by generating **time events** at interval boundaries, in order to not integrate over a discontinuous or not differentiable points.

Example:

```
table = [0 0
         1 0
         1 1
         2 4
         3 9
         4 16]
```

If, e.g., time = 1.0, the voltage v = 0.0 (before event), 1.0 (after event)  
e.g., time = 1.5, the voltage v = 2.5,  
e.g., time = 2.0, the voltage v = 4.0,  
e.g., time = 5.0, the voltage v = 23.0 (i.e. extrapolation).

## Parameters

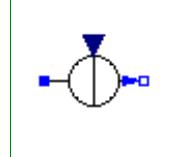
Name	Default	Description
table[:, :]	[0, 0; 1, 1; 2, 4]	Table matrix (time = first column, voltage = second column)
offset	0	Voltage offset [V]
startTime	0	Time offset [s]

## Connectors

Name	Description
p	Positive pin (potential p.v > n.v for positive voltage drop v)
n	Negative pin

## Modelica.Electrical.Analog.Sources.SignalCurrent

Generic current source using the input signal as source current

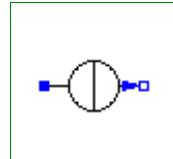


## Connectors

Name	Description
p	
n	
i	Current flowing from pin p to pin n as input signal

## Modelica.Electrical.Analog.Sources.ConstantCurrent

Source for constant current



## Parameters

Name	Default	Description
I	1	Value of constant current [A]

## Connectors

Name	Description

## 252 Modelica.Electrical.Analog.Sources.ConstantCurrent

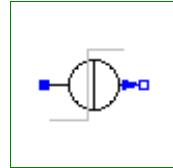
---

p	Positive pin (potential p.v > n.v for positive voltage drop v)
n	Negative pin

---

## Modelica.Electrical.Analog.Sources.StepCurrent

Step current source



### Parameters

Name	Default	Description
I	1	Height of step [A]
offset	0	Current offset [A]
startTime	0	Time offset [s]

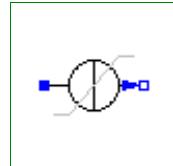
### Connectors

Name	Description
p	Positive pin (potential p.v > n.v for positive voltage drop v)
n	Negative pin

---

## Modelica.Electrical.Analog.Sources.RampCurrent

Ramp current source



### Parameters

Name	Default	Description
I	1	Height of ramp [A]
duration	2	Duration of ramp [s]
offset	0	Current offset [A]
startTime	0	Time offset [s]

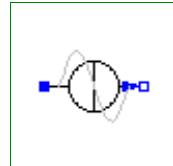
### Connectors

Name	Description
p	Positive pin (potential p.v > n.v for positive voltage drop v)
n	Negative pin

---

## Modelica.Electrical.Analog.Sources.SineCurrent

Sine current source



### Parameters

Name	Default	Description
I	1	Amplitude of sine wave [A]

phase	0	Phase of sine wave [rad]
freqHz	1	Frequency of sine wave [Hz]
offset	0	Current offset [A]
startTime	0	Time offset [s]

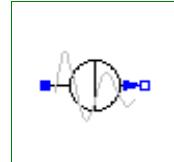
## Connectors

Name	Description
p	Positive pin (potential p.v > n.v for positive voltage drop v)
n	Negative pin

---

## Modelica.Electrical.Analog.Sources.ExpSineCurrent

Exponentially damped sine current source



## Parameters

Name	Default	Description
I	1	Amplitude of sine wave
freqHz	2	Frequency of sine wave [Hz]
phase	0	Phase of sine wave [rad]
damping	1	Damping coefficient of sine wave [s-1]
offset	0	Current offset [A]
startTime	0	Time offset [s]

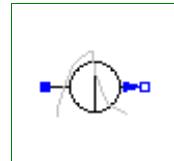
## Connectors

Name	Description
p	Positive pin (potential p.v > n.v for positive voltage drop v)
n	Negative pin

---

## Modelica.Electrical.Analog.Sources.ExponentialsCurrent

Rising and falling exponential current source



## Parameters

Name	Default	Description
iMax	1	Upper bound for rising edge
riseTime	0.5	Rise time [s]
riseTimeConst	0.1	Rise time constant [s]
fallTimeConst	riseTimeConst	Fall time constant [s]
offset	0	Current offset [A]
startTime	0	Time offset [s]

## Connectors

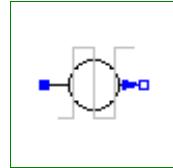
Name	Description

## 254 Modelica.Electrical.Analog.Sources.ExponentialsCurrent

p	Positive pin (potential p.v > n.v for positive voltage drop v)
n	Negative pin

## Modelica.Electrical.Analog.Sources.PulseCurrent

Pulse current source



### Parameters

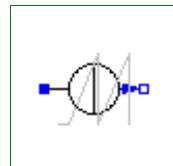
Name	Default	Description
I	1	Amplitude of pulse [A]
width	50	Width of pulse in % of period
period	1	Time for one period [s]
offset	0	Current offset [A]
startTime	0	Time offset [s]

### Connectors

Name	Description
p	Positive pin (potential p.v > n.v for positive voltage drop v)
n	Negative pin

## Modelica.Electrical.Analog.Sources.SawToothCurrent

Saw tooth current source



### Parameters

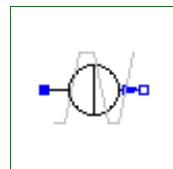
Name	Default	Description
I	1	Amplitude of saw tooth [A]
period	1	Time for one period [s]
offset	0	Current offset [A]
startTime	0	Time offset [s]

### Connectors

Name	Description
p	Positive pin (potential p.v > n.v for positive voltage drop v)
n	Negative pin

## Modelica.Electrical.Analog.Sources.TrapezoidCurrent

Trapezoidal current source



## Parameters

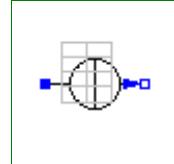
Name	Default	Description
I	1	Amplitude of trapezoid [A]
rising	0	Rising duration of trapezoid [s]
width	0.5	Width duration of trapezoid [s]
falling	0	Falling duration of trapezoid [s]
period	1	Time for one period [s]
nperiod	-1	Number of periods (< 0 means infinite number of periods)
offset	0	Current offset [A]
startTime	0	Time offset [s]

## Connectors

Name	Description
p	Positive pin (potential p.v > n.v for positive voltage drop v)
n	Negative pin

## Modelica.Electrical.Analog.Sources.TableCurrent

Current source by linear interpolation in a table



## Information

This block generates a current source by **linear interpolation** in a table. The time points and current values are stored in a matrix **table[i,j]**, where the first column **table[:,1]** contains the time points and the second column contains the current to be interpolated. The table interpolation has the following properties:

- The time points need to be **monotonically increasing**.
- **Discontinuities** are allowed, by providing the same time point twice in the table.
- Values **outside** of the table range, are computed by **extrapolation** through the last or first two points of the table.
- If the table has only **one row**, no interpolation is performed and the current value is just returned independantly of the actual time instant, i.e., this is a constant current source.
- Via parameters **startTime** and **offset** the curve defined by the table can be shifted both in time and in the current.
- The table is implemented in a numerically sound way by generating **time events** at interval boundaries, in order to not integrate over a discontinuous or not differentiable points.

Example:

```
table = [0  0
         1  0
         1  1
         2  4
         3  9
         4 16]
```

If, e.g., time = 1.0, the current i = 0.0 (before event), 1.0 (after event)  
e.g., time = 1.5, the current i = 2.5,  
e.g., time = 2.0, the current i = 4.0,  
e.g., time = 5.0, the current i = 23.0 (i.e. extrapolation).

## Parameters

Name	Default	Description
table[:, :]	[0, 0; 1, 1; 2, 4]	Table matrix (time = first column, current = second column)
offset	0	Current offset [A]
startTime	0	Time offset [s]

## Connectors

Name	Description
p	Positive pin (potential p.v > n.v for positive voltage drop v)
n	Negative pin

---

## Modelica.Electrical.Digital

Library for digital electrical components based on the VHDL standard with 9-valued logic and conversion to 2-,3-,4-valued logic

## Information

This library contains packages for digital electrical components. Both, type system and models are based on the VHDL standard (IEEE Std 1076-1987 VHDL, IEEE Std 1076-1993 VHDL, IEEE Std 1164 Multivalue Logic System):

- Interfaces: Definition of signals and interfaces
- Tables: All truth tables needed
- Delay: Transport and inertial delay
- Basic: Basic logic without delay
- Gates: Basic gates composed by basic components and inertial delay
- Tristate: (not yet available)
- FlipFlops: (not yet available)
- Latches: (not yet available)
- TransferGates: (not yet available)
- Multiplexers (not yet available)
- Memory: Ram, Rom, (not yet available)
- Sources: Time-dependend signal sources
- Converters
- Examples

The logic values are coded by integer values. The following code table is necessary for both setting of input and interpreting the output values.

### Code Table:

Logic value	Integer code	Meaning
'U'	1	Uninitialized
'X'	2	Forcing Unknown
'0'	3	Forcing 0
'1'	4	Forcing 1
'Z'	5	High Impedance
'W'	6	Weak Unknown
'L'	7	Weak 0
'H'	8	Weak 1
'.'	9	Don't care

The library will be developed in two main steps. The first step contains the basic components and the gates. In the next step the more complicated devices will be added. Currently the first step of the library is implemented and released for public use.

Copyright © 1998-2007, Modelica Association and Fraunhofer-Gesellschaft.

*This Modelica package is free software; it can be redistributed and/or modified under the terms of the Modelica license, see the license conditions and the accompanying disclaimer [here](#).*

## Package Content

Name	Description
 <a href="#">UsersGuide</a>	User's Guide
 Examples	Examples that demonstrate the usage of the Digital electrical components
 Interfaces	Connectors for Digital electrical components
 Tables	Truth tables for all components of package Digital
 Delay	Transport and inertial delay blocks
 Basic	Basic logic blocks without delays
 Gates	Logic gates including delays
 Sources	Time-dependend digital signal sources
 Converters	Converters between 2-,3-,4- and 9-valued logic

---

## Modelica.Electrical.Digital.UsersGuide



### User's Guide of package Electrical.Digital

Library **Electrical.Digital** is a **free** Modelica package providing components to model **digital** electronic systems based on combinational and sequential logic in a convenient way. This package contains the **user's Guide** for the library and has the following content:

1. [Overview of library](#) gives an overview of the library.
2. [A first example](#) demonstrates at hand of a first example how to use this library.
3. [An application example](#) demonstrates a generic n-bit adder. .
4. [Release Notes](#) summarizes the differences between different versions of this library.
5. [Literature](#) provides references that have been used to design and implement this library.
6. [Contact](#) provides information about the authors of the library as well as acknowledgments.

---

## Modelica.Electrical.Digital.Examples

### Examples that demonstrate the usage of the Digital electrical components

#### Information

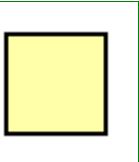
This package contains examples that demonstrate the usage of the components of the Electrical.Digital library.

## Package Content

Name	Description
 Multiplexer	4 to 1 Bit Multiplexer Example
 FlipFlop	Pulse Triggered Master Slave Flip-Flop
 HalfAdder	adding circuit for binary numbers without input carry bit
 FullAdder	Full 1 Bit Adder Example
 Adder4	4 Bit Adder Example
 Counter3	3 Bit Counter Example
 Counter	Generic N Bit Counter Example
 Utilities	Utility components used by package Examples

### Modelica.Electrical.Digital.Examples.Multiplexer

#### 4 to 1 Bit Multiplexer Example



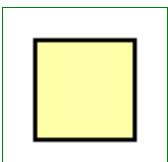
#### Information

##### 4 to 1 Bit Multiplexer

The multiplexer converts a parallel 4 bit signal in a sequential 1 bit stream.

### Modelica.Electrical.Digital.Examples.FlipFlop

#### Pulse Triggered Master Slave Flip-Flop



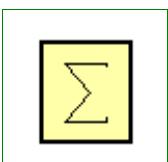
#### Information

##### FlipFlop

Pulse-triggered master-slave flip-flop.

### Modelica.Electrical.Digital.Examples.HalfAdder

#### adding circuit for binary numbers without input carry bit



#### Information

This example demonstrates an adding circuit for binary numbers, which internally realizes the interconnection to And and to Xor in the final sum.

$$1 + 0 = 1$$

$$0 + 1 = 1$$

$$1 + 1 = 10$$

$$0 + 0 = 0$$

$$\mathbf{a} + \mathbf{b} = \mathbf{s}$$

(The carry of this adding is **c**.)

and

$$\mathbf{a} * \mathbf{b} = \mathbf{s}$$

(It is an interconnection to And.)

$$\mathbf{a} * \mathbf{b} + \mathbf{a} * \mathbf{b} = \mathbf{a} \text{ Xor } \mathbf{b} = \mathbf{c}$$

(It is an interconnection to Xor.)

<b>a</b>	<b>b</b>	<b>c</b>	<b>s</b>	<b>t</b>
1	0	1	0	1
0	1	1	0	2
1	1	0	1	3
0	0	0	0	4

**t** is the pick-up instant of the next bit(s) in the simulation. The simulation stop time should be 5 seconds.

## Modelica.Electrical.Digital.Examples.FullAdder

### Full 1 Bit Adder Example



#### Information

It is an adding circuit for binary numbers with input carry bit, which consists of two HalfAdders.

**a.y**, **b.y** and **c.y** are the inputs of the FullAdder.

**cout** = **Or1.y** and **h.s** are the outputs of the Fulladder.

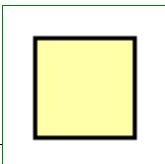
**t** is the pick-up instant of the next bit(s) in the simulation.

<b>a.y</b>	<b>b.y</b>	<b>c.y</b>	<b>cout</b>	<b>h.s</b>	<b>t</b>
1	0	0	0	1	1
0	1	0	0	1	2
0	0	1	0	1	3
1	1	0	1	0	4
0	1	1	1	0	5
1	0	1	1	0	6
1	1	1	1	1	7
0	0	0	0	0	8

The simulation stop time should be 10 seconds.

## Modelica.Electrical.Digital.Examples.Adder4

### 4 Bit Adder Example



## Information

Four Fulladders are combined to built a four bit adder unit.

In dependence on time five additions are carried out:

at t = 0	at t = 1
a        0 0 0 0	a        1 1 1 0
b        + 0 0 0 0	b        + 1 0 1 1
s        0 0 0 0 0	s        1 0 0 1 0
at t = 2	at t = 3
a        0 1 1 0	a        1 1 1 0
b        + 0 0 1 1	b        + 1 0 1 0
s        1 0 1 0 0	s        0 0 0 1 1
at t = 4	
a        1 1 0 0	
b        + 1 1 1 0	
s        0 0 1 0 1	

To show the influence of delay a large delay time of 0.1s is choosen. Furthermore, all signals are initialized with U, the uninitialized value. Please remember, that the nine logic values are coded by the numbers 1,...,9. The summands a and b can be found at the output signals of the taba and tabb sources. The result can be seen in the output signals of the Fulladders according to:

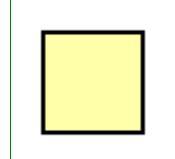
a	a4.y	a3.y	a2.y	a1.y
b	b4.y	b3.y	b2.y	b1.y
sum	Adder4.c_out	Adder4.s	Adder3.s	Adder2.s
				Adder1.s

The simulation stop time has to be 5s.

---

## Modelica.Electrical.Digital.Examples.Counter3

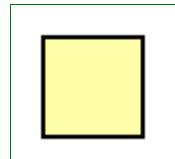
### 3 Bit Counter Example



## Information

## Modelica.Electrical.Digital.Examples.Counter

### Generic N Bit Counter Example



## Information

## Modelica.Electrical.Digital.Examples.Utilities

### Utility components used by package Examples

## Information

This package contains utility components used by package Examples.

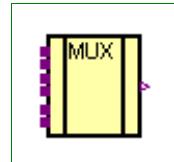
## Package Content

Name	Description
MUX4	4 to 1 Bit Multiplexer
RS	Unclocked RS FlipFlop
RSFF	Unclocked RS FlipFlop
DFF	D FlipFlop
JKFF	JK FlipFlop
HalfAdder	
FullAdder	adding circuit for binary numbers with input carry bit
Adder	Generic N Bit Adder
Counter3	3 Bit Counter
Counter	Generic N Bit Counter

## Modelica.Electrical.Digital.Examples.Utilities.MUX4

### 4 to 1 Bit Multiplexer

#### Information



#### Parameters

Name	Default	Description
delayTime	0.001	[s]
q0	L.'0'	

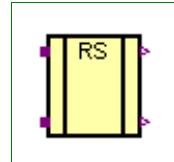
#### Connectors

Name	Description
d0	
d1	
d2	
d3	
a0	
a1	
d	

## Modelica.Electrical.Digital.Examples.Utilities.RS

### Unclocked RS FlipFlop

#### Information



#### Parameters

Name	Default	Description

## 262 Modelica.Electrical.Digital.Examples.Utilities.RS

---

delayTime	0	delay time [s]
q0	L.'U'	initial value of output

### Connectors

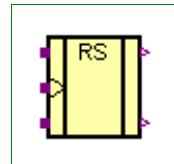
Name	Description
s	
r	
q	
qn	

---

## Modelica.Electrical.Digital.Examples.Utilities.RSFF

### Unclocked RS FlipFlop

#### Information



#### Parameters

Name	Default	Description
delayTime	0.01	[s]
q0	L.'U'	

### Connectors

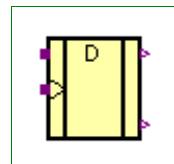
Name	Description
s	
r	
q	
qn	not Q
clk	

---

## Modelica.Electrical.Digital.Examples.Utilities.DFF

### D FlipFlop

#### Information



#### Parameters

Name	Default	Description
Tdel	0.01	[s]
QInit	L.'U'	

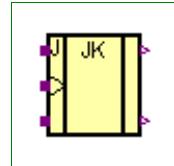
### Connectors

Name	Description
d	
q	

qn	not Q
clk	

**Modelica.Electrical.Digital.Examples.Utilities.JKFF**

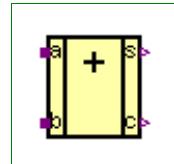
JK FlipFlop

**Information****Parameters**

Name	Default	Description
delayTime	0.001	[s]
q0	L.'0'	

**Connectors**

Name	Description
j	
q	
qn	not Q
clk	
k	

**Modelica.Electrical.Digital.Examples.Utilities.HalfAdder****Information****Parameters**

Name	Default	Description
delayTime	0	

**Connectors**

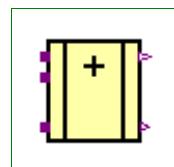
Name	Description
b	
s	
a	
c	

**Modelica.Electrical.Digital.Examples.Utilities.FullAdder**

adding circuit for binary numbers with input carry bit

**Information**

a      b      c in      c out      s



1	1	1	0
0	0	0	0
1	0	0	1
0	1	0	1

## Connectors

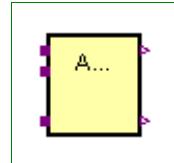
Name	Description
a	
b	
c_in	
s	
c_out	

---

## Modelica.Electrical.Digital.Examples.Utilities.Adder

### Generic N Bit Adder

## Information



## Parameters

Name	Default	Description
n	2	

## Connectors

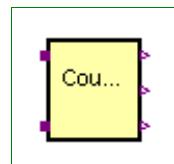
Name	Description
a[n]	
b[n]	
c_in	
s[n]	
c_out	

---

## Modelica.Electrical.Digital.Examples.Utilities.Counter3

### 3 Bit Counter

## Information



## Connectors

Name	Description
enable	
q2	
count	
q1	

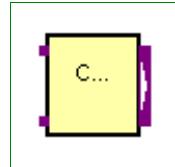
q0	
----	--

---

## Modelica.Electrical.Digital.Examples.Utilities.Counter

### Generic N Bit Counter

#### Information



#### Parameters

Name	Default	Description
n	3	
delayTime	0.001	[s]
q0	L.'0'	

#### Connectors

Name	Description
enable	
count	
q[n]	

---

## Modelica.Electrical.Digital.Interfaces

### Connectors for Digital electrical components

#### Information

This package contains interface definitions (connectors) digital electrical components.

#### Package Content

Name	Description
Logic	Signal type in package Digital according to the IEEE 1164 STD_ULOGIC type
(E) LogicValue	Logic values and their coding
DigitalSignal	Digital port (both input/output possible)
DigitalInput	input DigitalSignal as connector
DigitalOutput	output DigitalSignal as connector
SISO	Single input, single output
MISO	Multiple input - single output

#### Types and constants

```
type Logic = Integer(min=1,max=9)
"Signal type in package Digital according to the IEEE 1164 STD_ULOGIC type";
```

---

## Modelica.Electrical.Digital.Interfaces.Logic

Signal type in package Digital according to the IEEE 1164 STD\_ULOGIC type

### Information

A variable of type Logic is an Integer in the range 1-9. The Integer values have the following meaning:

Logic value	Integer code	Meaning
'U'	1	Uninitialized
'X'	2	Forcing Unknown
'0'	3	Forcing 0
'1'	4	Forcing 1
'Z'	5	High Impedance
'W'	6	Weak Unknown
'L'	7	Weak 0
'H'	8	Weak 1
'.'	9	Don't care

### Parameters

Name	Default	Description
min	1	
max	9	

---

## Modelica.Electrical.Digital.Interfaces.LogicValue

Logic values and their coding



### Information

Code Table:

Logic value	Integer code	Meaning
'U'	1	Uninitialized
'X'	2	Forcing Unknown
'0'	3	Forcing 0
'1'	4	Forcing 1
'Z'	5	High Impedance
'W'	6	Weak Unknown
'L'	7	Weak 0
'H'	8	Weak 1
'.'	9	Don't care

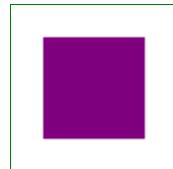
---

## Modelica.Electrical.Digital.Interfaces.DigitalSignal

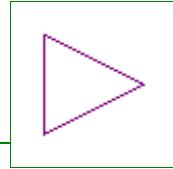
Digital port (both input/output possible)

**Information****Modelica.Electrical.Digital.Interfaces.DigitalInput**

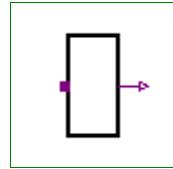
input DigitalSignal as connector

**Information****Modelica.Electrical.Digital.Interfaces.DigitalOutput**

output DigitalSignal as connector

**Modelica.Electrical.Digital.Interfaces.SISO**

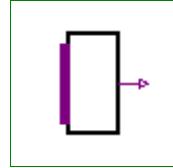
Single input, single output

**Information****Connectors**

Name	Description
x	Connector of Digital input signal
y	Connector of Digital output signal

**Modelica.Electrical.Digital.Interfaces.MISO**

Multiple input - single output

**Information****Parameters**

Name	Default	Description
n	2	Number of inputs

**Connectors**

Name	Description
x[n]	Connector of Digital input signal vector
y	Connector of Digital output signal

**Modelica.Electrical.Digital.Tables**

Truth tables for all components of package Digital

## Information

## Package Content

## Types and constants

```

constant D.Interfaces.Logic AndTable[L.max, L.max]=[  

    L.'U', L.'U', L.'0', L.'U', L.'U', L.'U', L.'0', L.'U', L.'U';  

    L.'U', L.'X', L.'0', L.'X', L.'X', L.'X', L.'0', L.'X', L.'X';  

    L.'0', L.'0', L.'0', L.'0', L.'0', L.'0', L.'0', L.'0', L.'0';  

    L.'U', L.'X', L.'0', L.'1', L.'X', L.'X', L.'0', L.'1', L.'X';  

    L.'U', L.'X', L.'0', L.'X', L.'X', L.'X', L.'0', L.'X', L.'X';  

    L.'U', L.'X', L.'0', L.'X', L.'X', L.'X', L.'0', L.'X', L.'X';  

    L.'0', L.'0', L.'0', L.'0', L.'0', L.'0', L.'0', L.'0', L.'0';  

    L.'U', L.'X', L.'0', L.'1', L.'X', L.'X', L.'0', L.'1', L.'X';  

    L.'U', L.'X', L.'0', L.'X', L.'X', L.'X', L.'0', L.'X', L.'X'];  

"9-value logic for 'and'";  

constant D.Interfaces.Logic OrTable[L.max, L.max]=[  

    L.'U', L.'U', L.'U', L.'1', L.'U', L.'U', L.'U', L.'1', L.'U';  

    L.'U', L.'X', L.'X', L.'1', L.'X', L.'X', L.'X', L.'1', L.'X';  

    L.'U', L.'X', L.'0', L.'1', L.'X', L.'X', L.'X', L.'0', L.'1', L.'X';  

    L.'1', L.'1', L.'1', L.'1', L.'1', L.'1', L.'1', L.'1', L.'1';  

    L.'U', L.'X', L.'X', L.'1', L.'X', L.'X', L.'X', L.'1', L.'X';  

    L.'U', L.'X', L.'X', L.'1', L.'X', L.'X', L.'X', L.'1', L.'X';  

    L.'U', L.'X', L.'0', L.'1', L.'X', L.'X', L.'0', L.'1', L.'X';  

    L.'1', L.'1', L.'1', L.'1', L.'1', L.'1', L.'1', L.'1', L.'1';  

    L.'U', L.'X', L.'X', L.'1', L.'X', L.'X', L.'X', L.'1', L.'X'];  

"9-value logic for 'or'";

```

```

constant D.Interfaces.Logic NotTable[L.max]={
  L.'U',L.'X',L.'1',L.'0',L.'X',L.'X',L.'1',L.'0',L.'X'}
"9-value logic for 'not'";

constant D.Interfaces.Logic XorTable[L.max, L.max]=[  

  L.'U', L.'U', L.'U', L.'U', L.'U', L.'U', L.'U', L.'U', L.'U';
  L.'U', L.'X', L.'X', L.'X', L.'X', L.'X', L.'X', L.'X', L.'X';
  L.'U', L.'X', L.'0', L.'1', L.'X', L.'X', L.'X', L.'X', L.'X';
  L.'U', L.'X', L.'1', L.'0', L.'X', L.'X', L.'1', L.'0', L.'X';
  L.'U', L.'X', L.'X', L.'X', L.'X', L.'X', L.'X', L.'X', L.'X';
  L.'U', L.'X', L.'X', L.'X', L.'X', L.'X', L.'X', L.'X', L.'X';
  L.'U', L.'X', L.'X', L.'X', L.'X', L.'X', L.'X', L.'X', L.'X';
  L.'U', L.'X', L.'0', L.'1', L.'X', L.'X', L.'0', L.'1', L.'X';
  L.'U', L.'X', L.'1', L.'0', L.'X', L.'X', L.'1', L.'0', L.'X';
  L.'U', L.'X', L.'X', L.'X', L.'X', L.'X', L.'X', L.'X', L.'X'];
"9-value logic for 'xor'";

constant D.Interfaces.Logic X01Table[L.max]={
  L.'X',L.'X',L.'0',L.'1',L.'X',L.'X',L.'0',L.'1',L.'X'};
constant D.Interfaces.Logic X01ZTable[L.max]={
  L.'X',L.'X',L.'0',L.'1',L.'Z',L.'X',L.'0',L.'1',L.'Z'};
constant D.Interfaces.Logic UX01Table[L.max]={
  L.'U',L.'X',L.'0',L.'1',L.'X',L.'X',L.'0',L.'1',L.'X'};

constant Integer DelayTable[9, 9]=[  

  0, 0, -1, 1, 0, 0, -1, 1, 0;  

  0, 0, -1, 1, 0, 0, -1, 1, 0;  

  1, 1, 0, 1, 1, 1, 0, 1, 1;  

  -1, -1, -1, 0, -1, -1, -1, 0, -1;  

  0, 0, -1, 1, 0, 0, -1, 1, 0;  

  0, 0, -1, 1, 0, 0, -1, 1, 0;  

  1, 1, 0, 1, 1, 1, 0, 1, 1;  

  -1, -1, -1, 0, -1, -1, -1, 0, -1;  

  0, 0, -1, 1, 0, 0, -1, 1, 0];

```

## Modelica.Electrical.Digital.Delay

### Transport and inertial delay blocks

#### Information

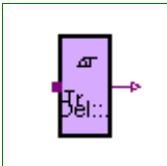
#### Package Content

Name	Description
DelayParams	Definition of delay parameters
TransportDelay	Transport delay with initial parameter
InertialDelay	Inertial delay with initial parameter
InertialDelaySensitive	Provide the input as output if it holds its value for a specific amount of time

**Modelica.Electrical.Digital.Delay.DelayParams****Definition of delay parameters****Information****Parameters**

Name	Default	Description
tLH	0	rise inertial delay [s]
tHL	0	fall inertial delay [s]
y0	L.'U'	initial value of output

---

**Modelica.Electrical.Digital.Delay.TransportDelay****Transport delay with initial parameter****Information**

Provide the input as output exactly delayed by  $T_{del}$ . If time less than  $T_{del}$  the initial value  $initout$  holds.

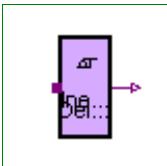
**Parameters**

Name	Default	Description
delayTime	0	delay time [s]
y0	L.'U'	initial value of output

**Connectors**

Name	Description
x	Connector of Digital input signal
y	Connector of Digital output signal

---

**Modelica.Electrical.Digital.Delay.InertialDelay****Inertial delay with initial parameter****Information**

Provides the input as output delayed by  $T_{del}$  if the input holds its value for a longer time than  $T_{del}$ . If time is less than  $T_{del}$  the initial value  $initout$  holds.

**Parameters**

Name	Default	Description
delayTime	0	Minimum time to hold value [s]
y0	L.'U'	Initial value of output y

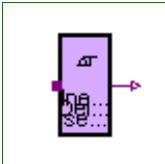
**Connectors**

Name	Description

x	Connector of Digital input signal
y	Connector of Digital output signal

## Modelica.Electrical.Digital.Delay.InertialDelaySensitive

Provide the input as output if it holds its value for a specific amount of time



### Information

Provides the input as output delayed by  $T_{del}$  if the input holds its value for a longer time than  $T_{del}$ . If the time is less than  $T_{del}$  the initial value  $initout$  holds.

The delay  $T_{del}$  depends on the values of the signal change. To calculate  $T_{del}$ , the delaymap specified in Digital.Tables is used. If the corresponding value is 1, then  $tLH$  is used, if it is -1, then  $tHL$  is used, if it is zero, the input is not delayed.

### Parameters

Name	Default	Description
tLH	0	rise inertial delay [s]
tHL	0	fall inertial delay [s]
y0	L.'U'	initial value of output

### Connectors

Name	Description
x	Connector of Digital input signal
y	Connector of Digital output signal

## Modelica.Electrical.Digital.Basic

Basic logic blocks without delays

### Information

### Package Content

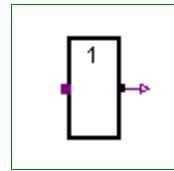
Name	Description
Not	Not Logic
And	And logic with multiple input and one output
Nand	Nand logic with multiple input and one output
Or	Or logic with multiple input and one output
Nor	Nor logic with multiple input and one output
Xor	Xor logic with multiple input and one output
Xnor	Xnor logic with multiple input and one output

## Modelica.Electrical.Digital.Basic.Not

**Not Logic**

### Information

Not with 1 input value, without delay.

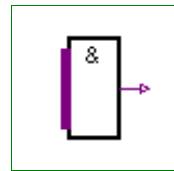


### Connectors

Name	Description
x	Connector of Digital input signal
y	Connector of Digital output signal

## Modelica.Electrical.Digital.Basic.And

And logic with multiple input and one output



### Information

And with n input values, without delay.

### Parameters

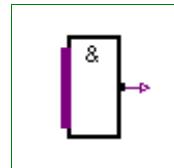
Name	Default	Description
n	2	Number of inputs

### Connectors

Name	Description
x[n]	Connector of Digital input signal vector
y	Connector of Digital output signal

## Modelica.Electrical.Digital.Basic.Nand

Nand logic with multiple input and one output



### Information

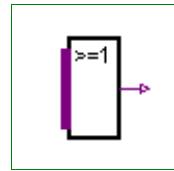
Nand with n input values, without delay.

### Parameters

Name	Default	Description
n	2	Number of inputs

### Connectors

Name	Description
x[n]	Connector of Digital input signal vector
y	Connector of Digital output signal

**Modelica.Electrical.Digital.Basic.Or****Or logic with multiple input and one output****Information**

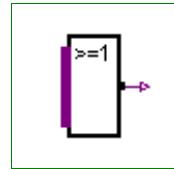
Or with n input values, without delay.

**Parameters**

Name	Default	Description
n	2	Number of inputs

**Connectors**

Name	Description
x[n]	Connector of Digital input signal vector
y	Connector of Digital output signal

**Modelica.Electrical.Digital.Basic.Nor****Nor logic with multiple input and one output****Information**

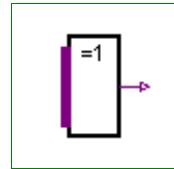
Nor with n input values, without delay.

**Parameters**

Name	Default	Description
n	2	Number of inputs

**Connectors**

Name	Description
x[n]	Connector of Digital input signal vector
y	Connector of Digital output signal

**Modelica.Electrical.Digital.Basic.Xor****Xor logic with multiple input and one output****Information**

Xor with n input values, without delay.

**Parameters**

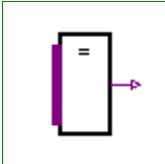
Name	Default	Description
n	2	Number of inputs

## Connectors

Name	Description
x[n]	Connector of Digital input signal vector
y	Connector of Digital output signal

## Modelica.Electrical.Digital.Basic.Xnor

Xnor logic with multiple input and one output



## Information

XNor with n input values, without delay.

## Parameters

Name	Default	Description
n	2	Number of inputs

## Connectors

Name	Description
x[n]	Connector of Digital input signal vector
y	Connector of Digital output signal

## Modelica.Electrical.Digital.Gates

Logic gates including delays

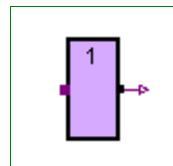
## Information

## Package Content

Name	Description
InvGate	InvGate with 1 input value, composed by Not and sensitive intertial delay
AndGate	AndGate with multiple input
NandGate	NandGate with multiple input
OrGate	OrGate with multiple input
NorGate	NorGate with multiple input
XorGate	XorGate with multiple input
XnorGate	XnorGate with multiple input
BufGate	BufGate with 1 input value, composed by Not and sensitive intertial delay

## Modelica.Electrical.Digital.Gates.InvGate

InvGate with 1 input value, composed by Not and sensitive intertial delay



## Information

InvGate with 1 input value, composed by Not and sensitive inertial delay.

## Parameters

Name	Default	Description
tLH	0	rise inertial delay [s]
tHL	0	fall inertial delay [s]
y0	L.'U'	initial value of output

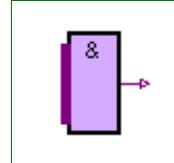
## Connectors

Name	Description
x	Connector of Digital input signal
y	Connector of Digital output signal

---

## Modelica.Electrical.Digital.Gates.AndGate

### AndGate with multiple input



## Information

AndGate with n input values, composed by And and sensitive inertial delay.

## Parameters

Name	Default	Description
n	2	Number of inputs
tLH	0	rise inertial delay [s]
tHL	0	fall inertial delay [s]
y0	L.'U'	initial value of output

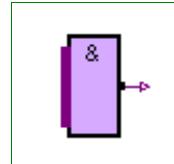
## Connectors

Name	Description
x[n]	Connector of Digital input signal vector
y	Connector of Digital output signal

---

## Modelica.Electrical.Digital.Gates.NandGate

### NandGate with multiple input



## Information

NandGate with n input values, composed by Nand and sensitive inertial delay.

## Parameters

Name	Default	Description
tLH	0	rise inertial delay [s]

## 276 Modelica.Electrical.Digital.Gates.NandGate

---

tHL	0	fall inertial delay [s]
y0	L.'U'	initial value of output
n	2	Number of inputs

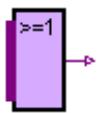
### Connectors

Name	Description
x[n]	Connector of Digital input signal vector
y	Connector of Digital output signal

---

## Modelica.Electrical.Digital.Gates.OrGate

OrGate with multiple input



### Information

OrGate with n input values, composed by Or and sensitive inertial delay.

### Parameters

Name	Default	Description
tLH	0	rise inertial delay [s]
tHL	0	fall inertial delay [s]
y0	L.'U'	initial value of output
n	2	Number of inputs

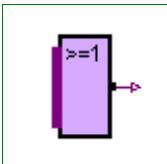
### Connectors

Name	Description
x[n]	Connector of Digital input signal vector
y	Connector of Digital output signal

---

## Modelica.Electrical.Digital.Gates.NorGate

NorGate with multiple input



### Information

NorGate with n input values, composed by Nor and sensitive inertial delay.

### Parameters

Name	Default	Description
tLH	0	rise inertial delay [s]
tHL	0	fall inertial delay [s]
y0	L.'U'	initial value of output
n	2	Number of inputs

### Connectors

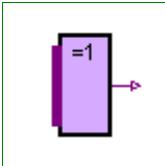
Name	Description

---

x[n]	Connector of Digital input signal vector
y	Connector of Digital output signal

## Modelica.Electrical.Digital.Gates.XorGate

XorGate with multiple input



### Information

XorGate with n input values, composed by Xor and sensitive inertial delay.

### Parameters

Name	Default	Description
tLH	0	rise inertial delay [s]
tHL	0	fall inertial delay [s]
y0	L.'U'	initial value of output
n	2	Number of inputs

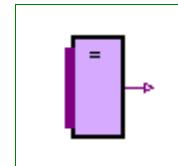
### Connectors

Name	Description
x[n]	Connector of Digital input signal vector
y	Connector of Digital output signal

---

## Modelica.Electrical.Digital.Gates.XnorGate

XnorGate with multiple input



### Information

XNorGate with n input values, composed by XNor and sensitive inertial delay.

### Parameters

Name	Default	Description
tLH	0	rise inertial delay [s]
tHL	0	fall inertial delay [s]
y0	L.'U'	initial value of output
n	2	Number of inputs

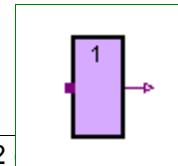
### Connectors

Name	Description
x[n]	Connector of Digital input signal vector
y	Connector of Digital output signal

---

## Modelica.Electrical.Digital.Gates.BufGate

BufGate with 1 input value, composed by Not and sensitive inertial delay



## Information

BufGate with 1 input value, composed by Not and sensitive inertial delay.

## Parameters

Name	Default	Description
tLH	0	rise inertial delay [s]
tHL	0	fall inertial delay [s]
y0	L.'U'	initial value of output

## Connectors

Name	Description
x	Connector of Digital input signal
y	Connector of Digital output signal

---

## Modelica.Electrical.Digital.Sources

### Time-dependend digital signal sources

## Information

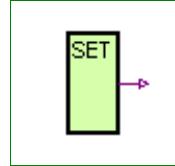
## Package Content

Name	Description
 Set	Digital Set Source
 Step	Digital Step Source
 Table	Digital Tabular Source
 Pulse	Digital Pulse Source
 Clock	Digital Clock Source

---

## Modelica.Electrical.Digital.Sources.Set

### Digital Set Source



## Information

Sets a nine valued digital signal, which is specified by the *setval* parameter.

To specify *setval*, the integer code has to be used.

## Code Table

Logic value	Integer code	Meaning
'U'	1	Uninitialized
'X'	2	Forcing Unknown
'0'	3	Forcing 0
'1'	4	Forcing 1
'Z'	5	High Impedance

'W'	6	Weak Unknown
'L'	7	Weak 0
'H'	8	Weak 1
'.'	9	Don't care

If the logic values are imported by

**import L = Modelica.Electrical.Digital.Interfaces.LogicValue;**  
they can be used to specify the parameter, e.g. L.'0' for forcing 0.

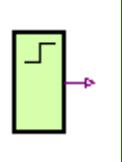
## Parameters

Name	Default	Description
x	L.'1'	Logic value to be set

## Connectors

Name	Description
y	

## Modelica.Electrical.Digital.Sources.Step



Digital Step Source

## Information

The step source output signal steps from the value *before* to the value *after* at the time *stepTime*.

To specify the logic value parameters, the integer code has to be used.

## Code Table

Logic value	Integer code	Meaning
'U'	1	Uninitialized
'X'	2	Forcing Unknown
'0'	3	Forcing 0
'1'	4	Forcing 1
'Z'	5	High Impedance
'W'	6	Weak Unknown
'L'	7	Weak 0
'H'	8	Weak 1
'.'	9	Don't care

If the logic values are imported by

**import L = Modelica.Electrical.Digital.Interfaces.LogicValue;**  
they can be used to specify the parameter, e.g. L.'0' for forcing 0.

## Parameters

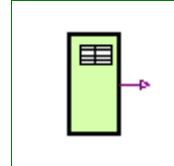
Name	Default	Description
before	L.'0'	Logic value before step
after	L.'1'	Logic value after step
stepTime	1	step time

## Connectors

Name	Description
y	

## Modelica.Electrical.Digital.Sources.Table

Digital Tabular Source



## Information

The table source output signal  $y$  steps to the values of the  $x$  table at the corresponding timepoints in the  $t$  table.

The initial value is specified by  $y0$ .

To specify the logic value parameters, the integer code has to be used.

## Code Table

Logic value	Integer code	Meaning
'U'	1	Uninitialized
'X'	2	Forcing Unknown
'0'	3	Forcing 0
'1'	4	Forcing 1
'Z'	5	High Impedance
'W'	6	Weak Unknown
'L'	7	Weak 0
'H'	8	Weak 1
'.'	9	Don't care

If the logic values are imported by

```
import L = Modelica.Electrical.Digital.Interfaces.LogicValue;
```

they can be used to specify the parameter, e.g. `L.'0'` for forcing 0.

## Parameters

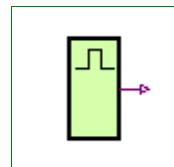
Name	Default	Description
$x[:]$	{1}	
$t[\text{size}(x, 1)]$	{1}	
$y0$	<code>L.'U'</code>	

## Connectors

Name	Description
y	

## Modelica.Electrical.Digital.Sources.Pulse

Digital Pulse Source



## Information

The pulse source forms pulses between the *quiet* value and the *pulse* value. The pulse length *width* is specified in percent of the period length *period*. The number of periods is specified by *nperiod*. If *nperiod* is

less than zero, the number of periods is unlimited.

To specify the logic value parameters, the integer code has to be used.

### Code Table

Logic value	Integer code	Meaning
'U'	1	Uninitialized
'X'	2	Forcing Unknown
'0'	3	Forcing 0
'1'	4	Forcing 1
'Z'	5	High Impedance
'W'	6	Weak Unknown
'L'	7	Weak 0
'H'	8	Weak 1
'.'	9	Don't care

If the logic values are imported by

`import L = Modelica.Electrical.Digital.Interfaces.LogicValue;`  
they can be used to specify the parameter, e.g. `L.'0'` for forcing 0.

### Parameters

Name	Default	Description
width	50	Widths of pulses in % of periods
period	1	Time for one period [s]
startTime	0	Output = offset for time < startTime [s]
pulse	L.'0'	
quiet	L.'1'	
nperiod	-1	Number of periods (< 0 means infinite number of periods)

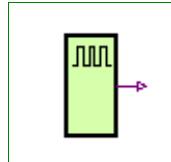
### Connectors

Name	Description
y	

---

## Modelica.Electrical.Digital.Sources.Clock

Digital Clock Source



### Information

The clock source forms pulses between the '0' value (forcing 0) and the '1' value (forcing 1). The pulse length *width* is specified in percent of the period length *period*. The number of periods is unlimited. The first pulse starts at *startTime*.

The clock source is a special but often used variant of the pulse source.

### Parameters

Name	Default	Description
startTime	0	Output = offset for time < startTime [s]
period	1	Time for one period [s]

width	50	Width of pulses in % of period
-------	----	--------------------------------

## Connectors

Name	Description
y	Connector of Digital output signal

---

## Modelica.Electrical.Digital.Converters

Converters between 2-,3-,4- and 9-valued logic

## Information

### Package Content

Name	Description
LogicToXO1	Conversion to XO1
LogicToXO1Z	Conversion to XO1Z
LogicToUXO1	Conversion to UXO1
BooleanToLogic	Boolean to Logic converter
LogicToBoolean	Logic to Boolean converter
RealToLogic	Real to Logic converter
LogicToReal	Logic to Real converter

---

## Modelica.Electrical.Digital.Converters.LogicToXO1



### Conversion to XO1

#### Information

Conversion of a nine valued digital input into a XO1 digital output without any delay according to IEEE 1164 To\_XO1 function.

#### Conversion Table:

input	output
'U' (coded by 1)	'X' (coded by 2)
'X' (coded by 2)	'X' (coded by 2)
'0' (coded by 3)	'0' (coded by 3)
'1' (coded by 4)	'1' (coded by 4)
'Z' (coded by 5)	'X' (coded by 2)
'W' (coded by 6)	'X' (coded by 2)
'L' (coded by 7)	'0' (coded by 3)
'H' (coded by 8)	'1' (coded by 4)
'-' (coded by 9)	'X' (coded by 2)

If the signal width is greater than 1 this conversion is done for each signal.

## Parameters

Name	Default	Description
n	1	signal width

## Connectors

Name	Description
x[n]	
y[n]	

## Modelica.Electrical.Digital.Converters.LogicToXO1Z



### Conversion to XO1Z

## Information

Conversion of a nine valued digital input into a X01Z digital output without any delay according to IEEE 1164 To\_XO1Z function.

### Conversion Table:

input	output
'U' (coded by 1)	'X' (coded by 2)
'X' (coded by 2)	'X' (coded by 2)
'0' (coded by 3)	'0' (coded by 3)
'1' (coded by 4)	'1' (coded by 4)
'Z' (coded by 5)	'Z' (coded by 5)
'W' (coded by 6)	'X' (coded by 2)
'L' (coded by 7)	'0' (coded by 3)
'H' (coded by 8)	'1' (coded by 4)
'-' (coded by 9)	'X' (coded by 2)

If the signal width is greater than 1 this conversion is done for each signal.

## Parameters

Name	Default	Description
n	1	signal width

## Connectors

Name	Description
x[n]	
y[n]	

## Modelica.Electrical.Digital.Converters.LogicToUX01



### Conversion to UX01

## Information

Conversion of a nine valued digital input into a UX01 digital output without any delay according to IEEE 1164 To\_UX01 function.

**Conversion Table:**

input	output
'U' (coded by 1)	'U' (coded by 1)
'X' (coded by 2)	'X' (coded by 2)
'0' (coded by 3)	'0' (coded by 3)
'1' (coded by 4)	'1' (coded by 4)
'Z' (coded by 5)	'X' (coded by 2)
'W' (coded by 6)	'X' (coded by 2)
'L' (coded by 7)	'0' (coded by 3)
'H' (coded by 8)	'1' (coded by 4)
'-' (coded by 9)	'X' (coded by 2)

If the signal width is greater than 1 this conversion is done for each signal.

**Parameters**

Name	Default	Description
n	1	signal width

**Connectors**

Name	Description
x[n]	
y[n]	

**Modelica.Electrical.Digital.Converters.BooleanToLogic**

Boolean to Logic converter

**Information**

Conversion of a Boolean input into a digital output without any delay according to:

input	output
true	'1' (coded by 4)
false	'0' (coded by 3)

If the signal width is greater than 1 this conversion is done for each signal.

**Parameters**

Name	Default	Description
n	1	signal width

**Connectors**

Name	Description
x[n]	
y[n]	

**Modelica.Electrical.Digital.Converters.LogicToBoolean**

Logic to Boolean converter



## Information

Conversion of a digital input into a Boolean output without any delay according to:

input	output
'U' (coded by 1)	false
'X' (coded by 2)	false
'0' (coded by 3)	false
'1' (coded by 4)	true
'Z' (coded by 5)	false
'W' (coded by 6)	false
'L' (coded by 7)	false
'H' (coded by 8)	true
'-' (coded by 9)	false

If the signal width is greater than 1 this conversion is done for each signal.

## Parameters

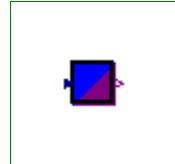
Name	Default	Description
n	1	signal width

## Connectors

Name	Description
x[n]	
y[n]	

## Modelica.Electrical.Digital.Converters.RealToLogic

Real to Logic converter



## Information

Conversion of a real input into a digital output without any delay according to:

	condition	output
first check:	input greater upp	lupp
second check:	input larger low	llow
	else	lmid

If the signal width is greater than 1 this conversion is done for each signal.

## Parameters

Name	Default	Description
n	1	signal width
upper_limit	1	upper limit
lower_limit	0	lower limit
upper_value	L.'1'	output if input > upper_limit
lower_value	L.'0'	output if input < lower_limit
middle_value	L.'X'	output else

## Connectors

Name	Description
x[n]	
y[n]	

## Modelica.Electrical.Digital.Converters.LogicToReal

Logic to Real converter



### Information

Conversion of a digital input into a Real output without any delay according to:

input	output
'U'	(coded by 1) val_U
'X'	(coded by 2) val_X
'0'	(coded by 3) val_0
'1'	(coded by 4) val_1
'Z'	(coded by 5) val_Z
'W'	(coded by 6) val_W
'L'	(coded by 7) val_L
'H'	(coded by 8) val_H
'-'	(coded by 9) val_m

The values val... are given by parameters.

If the signal width is greater than 1 this conversion is done for each signal.

## Parameters

Name	Default	Description
n	1	signal width
value_U	0.5	value for digital U (uninitialized)
value_X	0.5	value for digital X (Forcing Unknown)
value_0	0	value for digital 0 (Forcing 0)
value_1	1	value for digital 1 (Forcing 1)
value_Z	0.5	value for digital Z (High Impedance)
value_W	0.5	value for digital W (Weak Unknown)
value_L	0	value for digital L (Weak 0)
value_H	1	value for digital H (Weak 1)
value_m	0.5	value for digital m (Don't care)

## Connectors

Name	Description
x[n]	
y[n]	

## Modelica.Electrical.Machines

Library for electric machines

## Information

This package contains components to model electrical machines:

- Examples: test examples
- BasicMachines: basic machine models
- Sensors: sensors, usefull when modelling machines
- SpacePhasors: an independent library for using space phasors
- Interfaces: Space phasor connector and partial machine models

### Limitations and assumptions:

- number of phases (of induction machines) is limited to 3, therefore definition as a constant  $m=3$
- phase symmetric windings as well as symmetry of the whole machine structure
- all values are used in physical units, no scaling to p.u. is done
- only basic harmonics (in space) are taken into account
- waveform (with respect to time) of voltages and currents is not restricted
- constant parameters, i.e. no saturation, no skin effect
- no iron losses, eddy currents, friction losses;
- only ohmic losses in stator and rotor winding

You may have a look at a short summary of space phasor theory at  
<http://www.haumer.at/refimg/SpacePhasors.pdf>

### Further development:

- generalizing space phasor theory to  $m$  phases with arbitrary spatial angle of the coils
- generalizing space phasor theory to arbitrary number of windings and winding factor of the coils
- MachineModels: other machine types
- effects: saturation, skin-effect, other losses than ohmic, ...

### Main Authors:

Anton Haumer

Technical Consulting & Electrical Engineering  
A-3423 St.Andrae-Woerdern  
Austria  
email: [a.haumer@haumer.at](mailto:a.haumer@haumer.at)

Copyright © 1998-2007, Modelica Association and Anton Haumer.

*This Modelica package is free software; it can be redistributed and/or modified under the terms of the Modelica license, see the license conditions and the accompanying disclaimer here.*

## Package Content

Name	Description
 Examples	Test examples
 BasicMachines	Basic machine models
 Sensors	Sensors for machine modelling
 SpacePhasors	Library with space phasor-models
 Interfaces	SpacePhasor connector and PartialMachines

## Modelica.Electrical.Machines.Examples

### Test examples

## Information

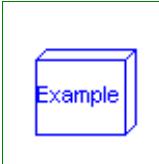
This package contains test examples of electric machines, and a package utilities with components used for the examples.

## Package Content

Name	Description
<a href="#">AIMC_DOL</a>	Test example 1: AsynchronousInductionMachineSquirrelCage direct-on-line
<a href="#">AIMC_YD</a>	Test example 2: AsynchronousInductionMachineSquirrelCage Y-D
<a href="#">AIMS_start</a>	Test example 3: AsynchronousInductionMachineSlipRing
<a href="#">AIMC_Inverter</a>	Test example 4: AsynchronousInductionMachineSquirrelCage with inverter
<a href="#">SMR_Inverter</a>	Test example 5: SynchronousInductionMachineReluctanceRotor with inverter
<a href="#">SMPM_Inverter</a>	Test example 6: PermanentMagnetSynchronousInductionMachine with inverter
<a href="#">SMEG_Gen</a>	Test example 7: ElectricalExcitedSynchronousInductionMachine as Generator
<a href="#">DCPM_start</a>	Test example 8: DC with permanent magnet starting with voltage ramp
<a href="#">DCEE_start</a>	Test example 9: DC with electrical excitation starting with voltage ramp
<a href="#">DCSE_start</a>	Test example 10: DC with serial excitation starting with voltage ramp
<a href="#">TransformerTestbench</a>	Transformer Testbench
<a href="#">Rectifier6pulse</a>	6-pulse rectifier with 1 transformer
<a href="#">Rectifier12pulse</a>	12-pulse rectifier with 2 transformers
<a href="#">AIMC_Steinmetz</a>	AsynchronousInductionMachineSquirrelCage Steinmetz-connection
<a href="#">Utilities</a>	Library with auxiliary models for testing

## Modelica.Electrical.Machines.Examples.AIMC\_DOL

### Test example 1: AsynchronousInductionMachineSquirrelCage direct-on-line



#### Information

##### 1st Test example: Asynchronous induction machine with squirrel cage - direct on line starting

At start time tStart three phase voltage is supplied to the asynchronous induction machine with squirrel cage; the machine starts from standstill, accelerating inertias against load torque quadratic dependent on speed, finally reaching nominal speed.

Simulate for 1.5 seconds and plot (versus time):

- CurrentRMSsensor1.I: stator current RMS
- AIMC1.rpm\_mechanical: motor's speed
- AIMC1.tau\_electrical: motor's torque

Default machine parameters of model *AIMC\_SquirrelCage* are used.

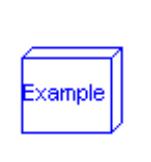
#### Parameters

Name	Default	Description
VNominal	100	nominal RMS voltage per phase [V]
fNominal	50	nominal frequency [Hz]
tStart1	0.1	start time [s]

T_Load	161.4	nominal load torque [N.m]
rpmLoad	1440.45	nominal load speed [rev/min]
J_Load	0.29	load's moment of inertia [kg.m <sup>2</sup> ]

**Modelica.Electrical.Machines.Examples.AIMC\_YD**

Test example 2: AsynchronousInductionMachineSquirrelCage Y-D

**Information****2nd Test example: Asynchronous induction machine with squirrel cage - Y-D starting**

At start time tStart three phase voltage is supplied to the asynchronous induction machine with squirrel cage, first star-connected, then delta-connected; the machine starts from standstill, accelerating inertias against load torque quadratic dependent on speed, finally reaching nominal speed.

Simulate for 2.5 seconds and plot (versus time):

- CurrentRMSsensor1.l: stator current RMS
- AIMC1.rpm\_mechanical: motor's speed
- AIMC1.tau\_electrical: motor's torque

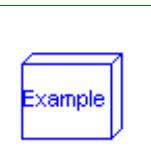
Default machine parameters of model *AIM\_SquirrelCage* are used.

**Parameters**

Name	Default	Description
VNominal	100	nominal RMS voltage per phase [V]
fNominal	50	nominal frequency [Hz]
tStart1	0.1	start time [s]
tStart2	2.0	2nd start time [s]
T_Load	161.4	nominal load torque [N.m]
rpmLoad	1440.45	nominal load speed [rev/min]
J_Load	0.29	load's moment of inertia [kg.m <sup>2</sup> ]

**Modelica.Electrical.Machines.Examples.AIMS\_start**

Test example 3: AsynchronousInductionMachineSlipRing

**Information****3rd Test example: Asynchronous induction machine with slipring rotor - resistance starting**

At start time tStart1 three phase voltage is supplied to the asynchronous induction machine with slippings; the machine starts from standstill, accelerating inertias against load torque quadratic dependent on speed, using a starting resistance. At time tStart2 external rotor resistance is shortened, finally reaching nominal speed.

Simulate for 1.5 seconds and plot (versus time):

- CurrentRMSsensor1.l: stator current RMS
- AIMS1.rpm\_mechanical: motor's speed
- AIMS1.tau\_electrical: motor's torque

Default machine parameters of model *AIM\_SlipRing* are used.

**Parameters**

Name	Default	Description

VNominal	100	nominal RMS voltage per phase [V]
fNominal	50	nominal frequency [Hz]
tStart1	0.1	1st start time [s]
Rstart	0.16	starting resistance [Ohm]
tStart2	1.0	2nd start time [s]
T_Load	161.4	nominal load torque [N.m]
rpmLoad	1440.45	nominal load speed [rev/min]
J_Load	0.29	load's moment of inertia [kg.m2]

**Modelica.Electrical.Machines.Examples.AIMC\_Inverter**

Test example 4: AsynchronousInductionMachineSquirrelCage with inverter

**Information****4th Test example: Asynchronous induction machine with squirrel cage fed by an ideal inverter**

An ideal frequency inverter is modeled by using a VfController and a threephase SignalVoltage.

Frequency is raised by a ramp, causing the asynchronous induction machine with squirrel cage to start, and accelerating inertias.

At time tStep a load step is applied.

Simulate for 1.5 seconds and plot (versus time):

- CurrentRMSSensor1.I: stator current RMS
- AIMC1.rpm\_mechanical: motor's speed
- AIMC1.tau\_electrical: motor's torque

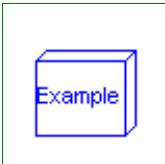
Default machine parameters of model AIM\_SquirrelCage are used.

**Parameters**

Name	Default	Description
VNominal	100	nominal RMS voltage per phase [V]
fNominal	50	nominal frequency [Hz]
f	50	actual frequency [Hz]
tRamp	1	frequency ramp [s]
T_Load	161.4	nominal load torque [N.m]
tStep	1.2	time of load torque step [s]
J_Load	0.29	load's moment of inertia [kg.m2]

**Modelica.Electrical.Machines.Examples.SMR\_Inverter**

Test example 5: SynchronousInductionMachineReluctanceRotor with inverter

**Information****5th Test example: Synchronous induction machine with reluctance rotor fed by an ideal inverter**

An ideal frequency inverter is modeled by using a VfController and a threephase SignalVoltage.

Frequency is raised by a ramp, causing the reluctance machine to start, and accelerating inertias.

At time tStep a load step is applied.

Simulate for 1.5 seconds and plot (versus time):

- CurrentRMSSensor1.I: stator current RMS

- SMRD1.rpm\_mechanical: motor's speed
- SMRD1.tau\_electrical: motor's torque
- RotorAngle.rotorAngle: rotor displacement angle

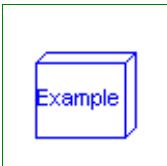
Default machine parameters of model *SM\_ReluctanceRotorDamperCage* are used.

## Parameters

Name	Default	Description
VNominal	100	nominal RMS voltage per phase [V]
fNominal	50	nominal frequency [Hz]
f	50	actual frequency [Hz]
tRamp	1	frequency ramp [s]
T_Load	46	nominal load torque [N.m]
tStep	1.2	time of load torque step [s]
J_Load	0.29	load's moment of inertia [kg.m <sup>2</sup> ]

## Modelica.Electrical.Machines.Examples.SMPM\_Inverter

Test example 6: PermanentMagnetSynchronousInductionMachine with inverter



## Information

### 6th Test example: Permanent magnet synchronous induction machine fed by an ideal inverter

An ideal frequency inverter is modeled by using a VfController and a threephase SignalVoltage.

Frequency is raised by a ramp, causing the permanent magnet synchronous induction machine to start, and accelerating inertias.

At time tStep a load step is applied.

Simulate for 1.5 seconds and plot (versus time):

- CurrentRMSsensor1.I: stator current RMS
- PMSMD1.rpm\_mechanical: motor's speed
- PMSMD1.tau\_electrical: motor's torque
- RotorAngle.rotorAngle: rotor displacement angle

Default machine parameters of model *SM\_PermanentMagnetDamperCage* are used.

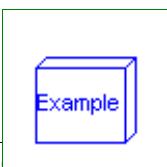
In practice it is nearly impossible to drive a PMSMD without current controller.

## Parameters

Name	Default	Description
VNominal	100	nominal RMS voltage per phase [V]
fNominal	50	nominal frequency [Hz]
f	50	actual frequency [Hz]
tRamp	1	frequency ramp [s]
T_Load	181.4	nominal load torque [N.m]
tStep	1.2	time of load torque step [s]
J_Load	0.29	load's moment of inertia [kg.m <sup>2</sup> ]

## Modelica.Electrical.Machines.Examples.SMEE\_Gen

Test example 7: ElectricalExcitedSynchronousInductionMachine as Generator



## Information

### 7th Test example: Electrical excited synchronous induction machine as generator

An electrically excited synchronous generator is connected to the grid and driven with constant speed. Since speed is slightly smaller than synchronous speed corresponding to mains frequency, rotor angle is very slowly increased. This allows to see several characteristics dependent on rotor angle. Simulate for 30 seconds and plot (versus RotorAngle1.rotorAngle):

- SMEED1.tau\_electrical
- CurrentRMSsensor1.I
- ElectricalPowerSensor1.P
- ElectricalPowerSensor1.Q
- MechanicalPowerSensor1.P

Default machine parameters of model *SM\_ElectricalExcitedDamperCage* are used.

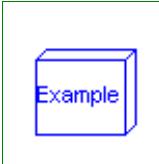
## Parameters

Name	Default	Description
VNominal	100	nominal RMS voltage per phase [V]
fNominal	50	nominal frequency [Hz]
rpm	1499	nominal speed [rev/min]
Ie	19	excitation current [A]
Ie0	10	initial excitation current [A]
gamma0	0	initial rotor displacement angle [deg]

---

### Modelica.Electrical.Machines.Examples.DCPM\_start

#### Test example 8: DC with permanent magnet starting with voltage ramp



## Information

### 8th Test example: Permanent magnet DC machine started with an armature voltage ramp

A voltage ramp is applied to the armature, causing the DC machine to start, and accelerating inertias.

At time tStep a load step is applied.

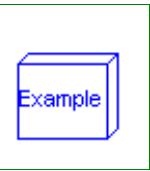
Simulate for 2 seconds and plot (versus time):

- DCPM1.ia: armature current
- DCPM1.rpm\_mechanical: motor's speed
- DCPM1.tau\_electrical: motor's torque

Default machine parameters of model *DC\_PermanentMagnet* are used.

## Parameters

Name	Default	Description
Va	100	actual armature voltage [V]
tStart	0.2	armature voltage ramp [s]
tRamp	0.8	armature voltage ramp [s]
T_Load	63.66	nominal load torque [N.m]
tStep	1.5	time of load torque step [s]
J_Load	0.15	load's moment of inertia [kg.m <sup>2</sup> ]

**Modelica.Electrical.Machines.Examples.DCEE\_start****Test example 9: DC with electrical excitation starting with voltage ramp****Information****9th Test example: Electrically separate excited DC machine started with an armature voltage ramp**

A voltage ramp is applied to the armature, causing the DC machine to start, and accelerating inertias.

At time tStep a load step is applied.

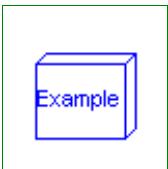
Simulate for 2 seconds and plot (versus time):

- DCEE1.ia: armature current
- DCEE1.rpm\_mechanical: motor's speed
- DCEE1.tau\_electrical: motor's torque
- DCEE1.ie: excitation current

Default machine parameters of model *DC\_ElectricalExcited* are used.

**Parameters**

Name	Default	Description
Va	100	actual armature voltage [V]
tStart	0.2	armature voltage ramp [s]
tRamp	0.8	armature voltage ramp [s]
Ve	100	actual excitation voltage [V]
T_Load	63.66	nominal load torque [N.m]
tStep	1.5	time of load torque step [s]
J_Load	0.15	load's moment of inertia [kg.m <sup>2</sup> ]

**Modelica.Electrical.Machines.Examples.DCSE\_start****Test example 10: DC with serial excitation starting with voltage ramp****Information****10th Test example: Series excited DC machine started with an armature voltage ramp**

A voltage ramp is applied to the armature, causing the DC machine to start, and accelerating inertias against load torque quadratic dependent on speed, finally reaching nominal speed.

Simulate for 2 seconds and plot (versus time):

- DCSE1.ia: armature current
- DCSE1.rpm\_mechanical: motor's speed
- DCSE1.tau\_electrical: motor's torque

Default machine parameters of model *DC\_SeriesExcited* are used.

**Parameters**

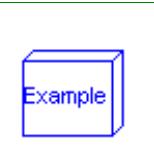
Name	Default	Description
Va	100	actual armature voltage [V]
tStart	0.2	armature voltage ramp [s]
tRamp	0.8	armature voltage ramp [s]
T_Load	63.66	nominal load torque [N.m]
rpmLoad	1410	nominal load speed [rev/min]

J_Load	0.15	load's moment of inertia [kg.m2]
--------	------	----------------------------------

---

## Modelica.Electrical.Machines.Examples.TransformerTestbench

### Transformer Testbench



#### Information

Transformer testbench:

You may choose different connections as well as vary the load (even not symmetrical).

**Please pay attention** to proper grounding of the primary and secondary part of the whole circuit.

The primary and secondary starpoint are available as connectors, if the connection is not delta (D or d).

In some cases it may be necessary to ground the transformer's starpoint even though the source's or load's starpoint are grounded; you may use a reasonable high earthing resistance.

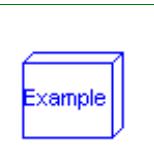
#### Parameters

Name	Default	Description
RL[3]	fill(1/3, 3)	Load resistance [Ohm]

---

## Modelica.Electrical.Machines.Examples.Rectifier6pulse

### 6-pulse rectifier with 1 transformer



#### Information

Test example with multiphase components:

Star-connected voltage source feeds via a transformer a diode bridge rectifier with a DC burden.

Using f=50 Hz, simulate for 0.1 seconds (5 periods) and compare voltages and currents of source and DC burden, neglecting initial transient.

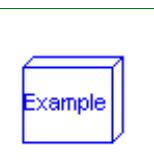
#### Parameters

Name	Default	Description
V	100*sqrt(2/3)	Amplitude of star-voltage [V]
f	50	Frequency [Hz]
RL	0.4	Load resistance [Ohm]
C	0.005	Total DC-capacitance [F]
VC0	sqrt(3)*V	Initial voltage of capacitance [V]

---

## Modelica.Electrical.Machines.Examples.Rectifier12pulse

### 12-pulse rectifier with 2 transformers



#### Information

Test example with multiphase components:

Star-connected voltage source feeds via two transformers (Dd0 and Dy1) two diode bridge rectifiers with a single DC burden.

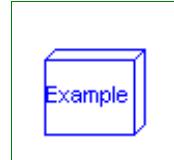
Using f=50 Hz, simulate for 0.1 seconds (5 periods) and compare voltages and currents of source and DC burden, neglecting initial transient.

## Parameters

Name	Default	Description
V	100*sqrt(2/3)	Amplitude of star-voltage [V]
f	50	Frequency [Hz]
RL	0.2	Load resistance [Ohm]
C	0.005	Total DC-capacitance [F]
VC0	sqrt(3)*V	Initial voltage of capacitance [V]

## Modelica.Electrical.Machines.Examples.AIMC\_Steinmetz

AsynchronousInductionMachineSquirrelCage Steinmetz-connection



## Information

### Asynchronous induction machine with squirrel cage - Steinmetz-connection

At start time tStart single phase voltage is supplied to the asynchronous induction machine with squirrel cage; the machine starts from standstill, accelerating inertias against load torque quadratic dependent on speed, finally reaching nominal speed.

Default machine parameters of model AIM\_SquirrelCage are used.

## Parameters

Name	Default	Description
VNominal	100	nominal RMS voltage per phase [V]
fNominal	50	nominal frequency [Hz]
tStart1	0.1	start time [s]
Cr	0.0035	motor's running capacitor [F]
Cs	5*Cr	motor's (additional) starting capacitor [F]
rpmSwitch	1350	speed for switching off the starting capacitor [rev/min]
T_Load	2/3*161.4	nominal load torque [N.m]
rpmLoad	1462.5	nominal load speed [rev/min]
J_Load	0.29	load's moment of inertia [kg.m <sup>2</sup> ]

## Modelica.Electrical.Machines.Examples.Utilities

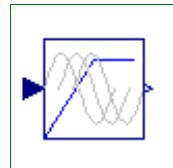
Library with auxiliary models for testing

## Information

This package contains components utility components for testing examples.

## Package Content

Name	Description
VfController	Voltage-Frequency-Controller
SwitchYD	Y-D-switch
TerminalBox	terminal box Y/D-connection

**Modelica.Electrical.Machines.Examples.Utilities.VfController****Voltage-Frequency-Controller****Information**

Simple Voltage-Frequency-Controller.

Amplitude of voltage is linear dependent ( $V_{\text{Nominal}}/f_{\text{Nominal}}$ ) on frequency (input signal "u"), but limited by  $V_{\text{Nominal}}$  (nominal RMS voltage per phase).

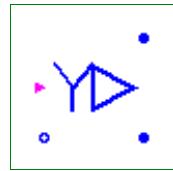
$m$  sine-waves with amplitudes as described above are provided as output signal "y".

The sine-waves are intended to feed a  $m$ -phase SignalVoltage.

Phase shifts between sine-waves may be chosen by the user; default values are  $(k-1)/m \cdot \pi$  for  $k$  in  $1:m$ .

**Parameters**

Name	Default	Description
$m$	3	number of phases
$V_{\text{Nominal}}$		nominal RMS voltage per phase [V]
$f_{\text{Nominal}}$		nominal frequency [Hz]
BasePhase	0	common phase shift [rad]

**Modelica.Electrical.Machines.Examples.Utilities.SwitchYD****Y-D-switch****Information**

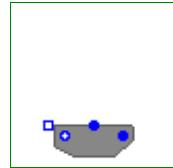
Simple Star-Delta-switch.

If  $control$  is false, plug\_PS and plug\_NS are star connected and plug\_PS connected to plug\_P.

If  $control$  is true, plug\_PS and plug\_NS are delta connected and they are connected to plug\_P.

**Connectors**

Name	Description
plug_P	
plug_PS	
plug_NS	
control[m]	

**Modelica.Electrical.Machines.Examples.Utilities.TerminalBox****terminal box Y/D-connection****Information**

TerminalBox: at the bottom connected to both machine plugs, connect at the top to the grid as usual, choosing Y-connection (StarDelta=Y) or D-connection (StarDelta=D).

**Parameters**

Name	Default	Description
StarDelta	"Y"	Choose Y=star/D=delta

## Connectors

Name	Description
positiveMachinePlug	
negativeMachinePlug	
plugToGrid	
starpoint	

---

## Modelica.Electrical.Machines.BasicMachines

### Basic machine models

#### Information

This package contains components for modeling electrical machines, specially threephase induction machines, based on space phasor theory:

- package AsynchronousInductionMachines: models of three phase asynchronous induction machines
- package SynchronousInductionMachines: models of three phase synchronous induction machines
- package DC Machines: models of DC machines with different excitation
- package Transformers: Threephase transformers (see detailed documentation in subpackage)
- package Components: components for modeling machines and transformers

The induction machine models use package SpacePhasors.

#### Package Content

Name	Description
 AsynchronousInductionMachines	Models of asynchronous induction machines
 SynchronousInductionMachines	Models of synchronous induction machines
 DC Machines	Models of DC machines
 Transformers	Library for technical 3phase transformers
 Components	Machine components like AirGaps

---

## Modelica.Electrical.Machines.BasicMachines.AynchronousInductionMachines

### Models of asynchronous induction machines

#### Information

This package contains models of asynchronous induction machines, based on space phasor theory:

- AIM\_SquirrelCage: asynchronous induction machine with squirrel cage
- AIM\_SlipRing: asynchronous induction machine with wound rotor

These models use package SpacePhasors.

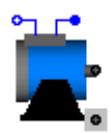
#### Package Content

Name	Description
 AIM_SquirrelCage	Asynchronous induction machine with squirrel cage rotor



AIM\_SlipRing

Asynchronous induction machine with sliring rotor

**Modelica.Electrical.Machines.BasicMachines.AynchronousInductionMachines.AIM\_S  
squirrelCage**


Asynchronous induction machine with squirrel cage rotor

### Information

#### Model of a three phase asynchronous induction machine with squirrel cage.

Resistance and stray inductance of stator is modeled directly in stator phases, then using space phasor transformation. Resistance and stray inductance of rotor's squirrel cage is modeled in two axis of the rotor-fixed coordinate system. Both together connected via a stator-fixed *AirGap* model. Only losses in stator and rotor resistance are taken into account.

**Default values for machine's parameters (a realistic example) are:**

number of pole pairs p	2	
stator's moment of inertia	0.29	kg.m <sup>2</sup>
rotor's moment of inertia	0.29	kg.m <sup>2</sup>
nominal frequency fNominal	50	Hz
nominal voltage per phase	100	V RMS
nominal current per phase	100	A RMS
nominal torque	161.4	Nm
nominal speed	1440.45	rpm
nominal mechanical output	24.346	kW
efficiency	92.7	%
power factor	0.875	
stator resistance	0.03	Ohm per phase in warm condition
rotor resistance	0.04	Ohm in warm condition
stator reactance Xs	3	Ohm per phase
rotor reactance Xr	3	Ohm
total stray coefficient sigma	0.0667	

These values give the following inductances, assuming equal stator and rotor stray inductances:

stator stray inductance per phase	$Xs * (1 - \sqrt{1 - \sigma}) / (2\pi f_{Nominal})$
rotor stray inductance	$Xr * (1 - \sqrt{1 - \sigma}) / (2\pi f_{Nominal})$
main field inductance per phase	$\sqrt{Xs * Xr * (1 - \sigma)} / (2\pi f_{Nominal})$

### Parameters

Name	Default	Description
J_Rotor	0.29	rotor's moment of inertia [kg.m <sup>2</sup> ]
J_Stator	J_Rotor	stator's moment of inertia [kg.m <sup>2</sup> ]
p	2	number of pole pairs (Integer)
fNominal	50	nominal frequency [Hz]

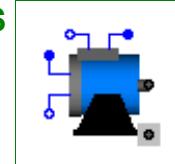
Nominal resistances and inductances		
Rs	0.03	warm stator resistance per phase [Ohm]
Lssigma	$3*(1 - \sqrt{1 - 0.0667})/(2*\pi*fNo...)$	stator stray inductance per phase [H]
Lm	$3*\sqrt{1 - 0.0667}/(2*\pi*fNo...)$	main field inductance [H]
Lrsigma	$3*(1 - \sqrt{1 - 0.0667})/(2*\pi*fNo...)$	rotor stray inductance (equivalent three phase winding) [H]
Rr	0.04	warm rotor resistance (equivalent three phase winding) [Ohm]

## Connectors

Name	Description
flange_a	
support	support at which the reaction torque is acting
plug_sp	
plug_sn	

## Modelica.Electrical.Machines.BasicMachines.AynchronousInductionMachines.AIM\_SlipRing

Asynchronous induction machine with slipring rotor



## Information

### Model of a three phase asynchronous induction machine with slipring rotor.

Resistance and stray inductance of stator and rotor are modeled directly in stator respectively rotor phases, then using space phasor transformation and a stator-fixed *AirGap* model. Only losses in stator and rotor resistance are taken into account.

Default values for machine's parameters (a realistic example) are:

number of pole pairs p	2	
stator's moment of inertia	0.29	kg.m <sup>2</sup>
rotor's moment of inertia	0.29	kg.m <sup>2</sup>
nominal frequency fNominal	50	Hz
nominal voltage per phase	100	V RMS
nominal current per phase	100	A RMS
nominal torque	161.4	Nm
nominal speed	1440.45	rpm
nominal mechanical output	24.346	kW
efficiency	92.7	%
power factor	0.875	
stator resistance	0.03	Ohm per phase in warm condition
rotor resistance	0.04	Ohm per phase in warm condition
stator reactance Xs	3	Ohm per phase
rotor reactance Xr	3	Ohm per phase
total stray coefficient sigma	0.0667	
TurnsRatio	1	effective ratio of stator and rotor current (ws*xis) / (wr*xir)

These values give the following  
inductances:

## 300 Modelica.Electrical.Machines.BasicMachines.AynchronousInductionMachines.AIM\_SlipRing

stator stray inductance per phase  $X_s * (1 - \sqrt{1 - \sigma}) / (2\pi f_{Nominal})$

rotor stray inductance  $X_r * (1 - \sqrt{1 - \sigma}) / (2\pi f_{Nominal})$

main field inductance per phase  $\sqrt{X_s X_r} * (1 - \sigma) / (2\pi f)$

Parameter TurnsRatio could be obtained from the following relationship at standstill with open rotor circuit at nominal voltage and nominal frequency,

using the locked-rotor voltage VR, no-load stator current I0 and powerfactor PF0:

$$\text{TurnsRatio} * V_R = V_s - (R_s + j X_{s,\sigma}) I_0$$

### Parameters

Name	Default	Description
J_Rotor	0.29	rotor's moment of inertia [kg.m <sup>2</sup> ]
J_Stator	J_Rotor	stator's moment of inertia [kg.m <sup>2</sup> ]
p	2	number of pole pairs (Integer)
fNominal	50	nominal frequency [Hz]
useTurnsRatio	true	use TurnsRatio or calculate from locked-rotor voltage?
TurnsRatio	1	(ws*xis) / (wr*xir)
VsNom	100	Nominal stator voltage per phase [V]
Vr_LR	$100 * (2\pi f_{Nominal} * L_m) / \sqrt{\dots}$	Locked-rotor voltage per phase [V]
Nominal resistances and inductances		
Rs	0.03	warm stator resistance per phase [Ohm]
Lssigma	$3 * (1 - \sqrt{1 - 0.0667}) / (2\pi f_{Nominal})$	stator stray inductance per phase [H]
Lm	$3 * \sqrt{1 - 0.0667} / (2\pi f_{Nominal})$	main field inductance [H]
Lrsigma	$3 * (1 - \sqrt{1 - 0.0667}) / (2\pi f_{Nominal})$	rotor stray inductance per phase [H]
Rr	0.04	warm rotor resistance per phase [Ohm]

### Connectors

Name	Description
flange_a	
support	support at which the reaction torque is acting
plug_sp	
plug_sn	
plug_rp	
plug_rn	

## Modelica.Electrical.Machines.BasicMachines.SynchronousInductionMachines

### Models of synchronous induction machines

#### Information

This package contains models of synchronous induction machines, based on space phasor theory:

- SM\_PermanentMagnetDamperCage: synchronous induction machine with permanent magnet excitation, with damper cage

- SM\_ElectricalExcitedDamperCage: synchronous induction machine with electrical excitation and damper cage
- SM\_ReluctanceRotorDamperCage: induction machine with reluctance rotor and damper cage i.e. a squirrel cage rotor with magnetic poles due to different airgap width

These models use package SpacePhasors.

**Please keep in mind:**

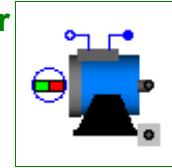
- We keep the same reference system as for motors, i.e.:  
Positive RotorDisplacementAngle means acting as motor,  
with positive electric power consumption and positive mechanical power output.
- ElectricalAngle = p \* MechanicalAngle
- real axis = d-axis  
imaginary= q-axis
- Voltage induced by the magnet wheel (d-axis) is located in the q-axis.

### Package Content

Name	Description
SM_PermanentMagnetDamperCage	Permanent magnet synchronous induction machine
SM_ElectricalExcitedDamperCage	Electrical excited synchronous induction machine with damper cage
SM_ReluctanceRotorDamperCage	Synchronous induction machine with reluctance rotor and damper cage

### Modelica.Electrical.Machines.BasicMachines.SynchronousInductionMachines.SM\_PermanentMagnetDamperCage

Permanent magnet synchronous induction machine



### Information

#### Model of a three phase permanent magnet synchronous induction machine.

Resistance and stray inductance of stator is modeled directly in stator phases, then using space phasor transformation and a rotor-fixed AirGap model. Resistance and stray inductance of rotor's squirrel cage is modeled in two axis of the rotor-fixed coordinate system. Permanent magnet excitation is modelled by a constant equivalent excitation current feeding the d-axis. Only losses in stator and damper resistance are taken into account.

Whether a damper cage is present or not, can be selected with Boolean parameter DamperCage (default = true).

#### Default values for machine's parameters (a realistic example) are:

number of pole pairs p	2	
stator's moment of inertia	0.29	kg.m <sup>2</sup>
rotor's moment of inertia	0.29	kg.m <sup>2</sup>
nominal frequency fNominal	50	Hz
nominal voltage per phase	100	V RMS
no-load voltage per phase	112.3	V RMS @ nominal speed
nominal current per phase	100	A RMS
nominal torque	181.4	Nm
nominal speed	1500	rpm
nominal mechanical output	28.5	kW
nominal rotor angle	20.75	degree
efficiency	95.0	%

power factor	0.98	
stator resistance	0.03	Ohm per phase in warm condition
stator reactance $X_d$	0.4	Ohm per phase in d-axis
stator reactance $X_q$	0.4	Ohm per phase in q-axis
stator stray reactance $X_{ss}$	0.1	Ohm per phase
damper resistance in d-axis	0.04	Ohm in warm condition
damper resistance in q-axis	same as d-axis	
damper stray reactance in d-axis $X_{Dds}$	0.05	Ohm
damper stray reactance in q-axis $X_{Dqs}$	same as d-axis	

These values give the following inductances:

main field inductance in d-axis	$(X_d - X_{ss})/(2\pi f_{Nominal})$
main field inductance in q-axis	$(X_q - X_{ss})/(2\pi f_{Nominal})$
stator stray inductance per phase	$X_{ss}/(2\pi f_{Nominal})$
damper stray inductance in d-axis	$X_{Dds}/(2\pi f_{Nominal})$
damper stray inductance in q-axis	$X_{Dqs}/(2\pi f_{Nominal})$

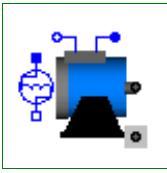
## Parameters

Name	Default	Description
J_Rotor	0.29	rotor's moment of inertia [kg.m <sup>2</sup> ]
J_Stator	J_Rotor	stator's moment of inertia [kg.m <sup>2</sup> ]
p	2	number of pole pairs (Integer)
fNominal	50	nominal frequency [Hz]
Nominal resistances and inductances		
Rs	0.03	warm stator resistance per phase [Ohm]
Lsigma	$0.1/(2\pi f_{Nominal})$	stator stray inductance per phase [H]
Lmd	$0.3/(2\pi f_{Nominal})$	main field inductance in d-axis [H]
Lmq	$0.3/(2\pi f_{Nominal})$	main field inductance in q-axis [H]
Excitation		
V0	112.3	no-load RMS voltage per phase @ fNominal [V]
DamperCage		
DamperCage	true	damper cage is present?
Lrsigma	$0.05/(2\pi f_{Nominal})$	damper stray inductance in d-axis [H]
Lrsigmaq	Lrsigma	damper stray inductance in q-axis [H]
Rr	0.04	warm damper resistance in d-axis [Ohm]
Rrq	Rr	warm damper resistance in q-axis [Ohm]

## Connectors

Name	Description
flange_a	
support	support at which the reaction torque is acting
plug_sp	
plug_sn	

## Modelica.Electrical.Machines.BasicMachines.SynchronousInductionMachines.SM\_ElectricalExcitedDamperCage



Electrical excited synchronous induction machine with damper cage

### Information

#### Model of a three phase electrical excited synchronous induction machine with damper cage.

Resistance and stray inductance of stator is modeled directly in stator phases, then using space phasor transformation and a rotor-fixed *AirGap* model. Resistance and stray inductance of rotor's squirrel cage is modeled in two axis of the rotor-fixed coordinate system. Electrical excitation is modelled by converting excitation current and voltage to d-axis space phasors. Only losses in stator, damper and excitation resistance are taken into account.

Whether a damper cage is present or not, can be selected with Boolean parameter DamperCage (default = true).

#### Default values for machine's parameters (a realistic example) are:

number of pole pairs p	2	
stator's moment of inertia	0.29	kg.m <sup>2</sup>
rotor's moment of inertia	0.29	kg.m <sup>2</sup>
nominal frequency fNominal	50	Hz
nominal voltage per phase	100	V RMS
no-load excitation current @ nominal voltage and frequency	10	A DC
warm excitation resistance	2.5	Ohm
nominal current per phase	100	A RMS
nominal apparent power	-30000	VA
power factor	-1.0	ind./cap.
nominal excitation current	19	A
efficiency w/o excitation	97.1	%
nominal torque	-196.7	Nm
nominal speed	1500	rpm
nominal rotor angle	-57.23	degree
stator resistance	0.03	Ohm per phase in warm condition
stator reactance Xd	1.6	Ohm per phase in d-axis
giving Kc	0.625	
stator reactance Xq	1.6	Ohm per phase in q-axis
stator stray reactance Xss	0.1	Ohm per phase
damper resistance in d-axis	0.04	Ohm in warm condition
damper resistance in q-axis	same as d-axis	
damper stray reactance in d-axis XDds	0.1	Ohm
damper stray reactance in q-axis XDqs	same as d-axis	
excitation stray inductance	2.5	% of total excitation inductance

These values give the following inductances:

main field inductance in d-axis	$(X_d - X_{ss})/(2\pi f_{Nominal})$
main field inductance in q-axis	$(X_q - X_{ss})/(2\pi f_{Nominal})$
stator stray inductance per phase	$X_{ss}/(2\pi f_{Nominal})$
damper stray inductance in d-axis	$X_{Dds}/(2\pi f_{Nominal})$
damper stray inductance in q-axis	$X_{Dqs}/(2\pi f_{Nominal})$

## Parameters

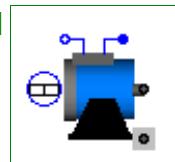
Name	Default	Description
J_Rotor	0.29	rotor's moment of inertia [kg.m <sup>2</sup> ]
J_Stator	J_Rotor	stator's moment of inertia [kg.m <sup>2</sup> ]
p	2	number of pole pairs (Integer)
fNominal	50	nominal frequency [Hz]
Nominal resistances and inductances		
Rs	0.03	warm stator resistance per phase [Ohm]
Lssigma	0.1/(2*pi*fNominal)	stator stray inductance per phase [H]
Lmd	1.5/(2*pi*fNominal)	main field inductance in d-axis [H]
Lmq	1.5/(2*pi*fNominal)	main field inductance in q-axis [H]
DamperCage		
DamperCage	true	damper cage is present?
Lrsigma	0.05/(2*pi*fNominal)	damper stray inductance in d-axis [H]
Lrsigmaq	Lrsigma	damper stray inductance in q-axis [H]
Rr	0.04	warm damper resistance in d-axis [Ohm]
Rrq	Rr	warm damper resistance in q-axis [Ohm]
Excitation		
VNominal	100	nominal stator RMS voltage per phase [V]
Ie0	10	no-load excitation current @ nominal voltage and frequency [A]
Re	2.5	warm excitation resistance [Ohm]
sigmae	0.025	stray fraction of total excitation inductance

## Connectors

Name	Description
flange_a	
support	support at which the reaction torque is acting
plug_sp	
plug_sn	
pin_ep	
pin_en	

## Modelica.Electrical.Machines.BasicMachines.SynchronousInductionMachines.SM\_ReluctanceRotorDamperCage

Synchronous induction machine with reluctance rotor and damper cage



## Information

### Model of a three phase synchronous induction machine with reluctance rotor and damper cage.

Resistance and stray inductance of stator is modeled directly in stator phases, then using space phasor transformation. Resistance and stray inductance of rotor's squirrel cage is modeled in two axis of the rotor-fixed coordinate system. Both together connected via a rotor-fixed AirGap model. Only losses in stator and rotor resistance are taken into account.

Whether a damper cage is present or not, can be selected with Boolean parameter DamperCage (default = true).

**Default values for machine's parameters (a realistic example) are:**

number of pole pairs p	2	
stator's moment of inertia	0.29	kg.m2
rotor's moment of inertia	0.29	kg.m2
nominal frequency fNominal	50	Hz
nominal voltage per phase	100	V RMS
nominal current per phase	50	A RMS
nominal torque	46	Nm
nominal speed	1500	rpm
nominal mechanical output	7.23	kW
efficiency	96.98	%
power factor	0.497	
stator resistance	0.03	Ohm per phase in warm condition
rotor resistance in d-axis	0.04	Ohm in warm condition
rotor resistance in q-axis	same as d-axis	
stator reactance Xsd in d-axis	3	Ohm per phase
stator reactance Xsq in q-axis	1	Ohm
stator stray reactance XSS	0.1	Ohm per phase
rotor stray reactance in d-axis Xrds	0.1	Ohm per phase
rotor stray reactance in q-axis Xrq	same as d-axis	

These values give the following inductances:

stator stray inductance per phase	Xss/(2*pi*fNominal)
rotor stray inductance in d-axis	Xrds/(2*pi*fNominal)
rotor stray inductance in q-axis	Xrq/(2*pi*fNominal)
main field inductance per phase in d-axis	(Xsd-Xss)/(2*pi*fNominal)
main field inductance per phase in q-axis	(Xsq-Xss)/(2*pi*fNominal)

## Parameters

Name	Default	Description
J_Rotor	0.29	rotor's moment of inertia [kg.m2]
J_Stator	J_Rotor	stator's moment of inertia [kg.m2]
p	2	number of pole pairs (Integer)
fNominal	50	nominal frequency [Hz]
Nominal resistances and inductances		
Rs	0.03	warm stator resistance per phase [Ohm]
Lsigma	0.1/(2*pi*fNominal)	stator stray inductance per phase [H]
Lmd	2.9/(2*pi*fNominal)	main field inductance in d-axis [H]
Lmq	0.9/(2*pi*fNominal)	main field inductance in q-axis [H]
DamperCage		
DamperCage	true	damper cage is present?
Lrsigma	0.05/(2*pi*fNominal)	damper stray inductance in d-axis [H]
Lrsigmaq	Lrsigma	damper stray inductance in q-axis [H]
Rr	0.04	warm damper resistance in d-axis [Ohm]
Rrq	Rr	warm damper resistance in q-axis [Ohm]

## Connectors

Name	Description
flange_a	
support	support at which the reaction torque is acting
plug_sp	
plug_sn	

## Modelica.Electrical.Machines.BasicMachines.DCMachines

### Models of DC machines

## Information

This package contains models of DC machines:

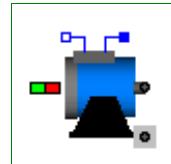
- DC\_PermanentMagnet: DC machine with permanent magnet excitation
- DC\_ElectricalExcited: DC machine with electrical shunt or separate excitation
- DC\_SeriesExcited: DC machine with series excitation

## Package Content

Name	Description
 DC_PermanentMagnet	Permanent magnet DC machine
 DC_ElectricalExcited	Electrical shunt/separate excited linear DC machine
 DC_SeriesExcited	Series excited linear DC machine

## Modelica.Electrical.Machines.BasicMachines.DCMachines.DC\_PermanentMagnet

### Permanent magnet DC machine



## Information

### Model of a DC Machine with Permanent magnet.

Armature resistance and inductance are modeled directly after the armature pins, then using a *AirGapDC* model. Permanent magnet excitation is modelled by a constant equivalent excitation current feeding *AirGapDC*. Only losses in armature resistance are taken into account. No saturation is modelled.

**Default values for machine's parameters (a realistic example) are:**

stator's moment of inertia	0.29	kg.m <sup>2</sup>
rotor's moment of inertia	0.15	kg.m <sup>2</sup>
nominal armature voltage	100	V
nominal armature current	100	A
nominal speed	1425	rpm
nominal torque	63.66	Nm
nominal mechanical output	9.5	kW
efficiency	95.0	%
armature resistance	0.05	Ohm in warm condition
armature inductance	0.0015	H

Armature resistance resp. inductance include resistance resp. inductance of commutating pole winding and

compensation windig, if present.

## Parameters

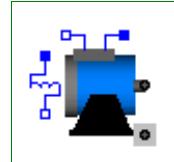
Name	Default	Description
J_Rotor	0.15	rotor's moment of inertia [kg.m2]
J_Stator	J_Rotor	stator's moment of inertia [kg.m2]
<b>Nominal parameters</b>		
VaNominal	100	nominal armature voltage [V]
IaNominal	100	nominal armature current [A]
rpmNominal	1425	nominal speed [rev/min]
<b>Nominal resistances and inductances</b>		
Ra	0.05	warm armature resistance [Ohm]
La	0.0015	armature inductance [H]

## Connectors

Name	Description
flange_a	
support	support at which the reaction torque is acting
pin_ap	
pin_an	

## Modelica.Electrical.Machines.BasicMachines.DCMachines.DC\_ElectricalExcited

Electrical shunt/separate excited linear DC machine



## Information

### Model of a DC Machine with Electrical shunt or separate excitation.

Armature resistance and inductance are modeled directly after the armature pins, then using a *AirGapDC* model.

Only losses in armature and excitation resistance are taken into account. No saturation is modelled.

Shunt or separate excitation is defined by the user's external circuit.

**Default values for machine's parameters (a realistic example) are:**

stator's moment of inertia	0.29	kg.m2
rotor's moment of inertia	0.15	kg.m2
nominal armature voltage	100	V
nominal armature current	100	A
nominal torque	63.66	Nm
nominal speed	1425	rpm
nominal mechanical output	9.5	kW
efficiency	95.0	% only armature
efficiency	94.06	% including excitation
armature resistance	0.05	Ohm in warm condition
armature inductance	0.0015	H
nominal excitation voltage	100	V
nominal excitation current	1	A
excitation resistance	100	Ohm in warm condition

excitation inductance 1 H

Armature resistance resp. inductance include resistance resp. inductance of commutating pole winding and compensation windig, if present.

Armature current does not cover excitation current of a shunt excitation; in this case total current drawn from the grid = armature current + excitation current.

## Parameters

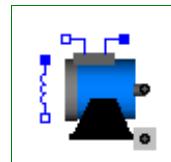
Name	Default	Description
J_Rotor	0.15	rotor's moment of inertia [kg.m2]
J_Stator	J_Rotor	stator's moment of inertia [kg.m2]
Nominal parameters		
VaNominal	100	nominal armature voltage [V]
IaNominal	100	nominal armature current [A]
rpmNominal	1425	nominal speed [rev/min]
Nominal resistances and inductances		
Ra	0.05	warm armature resistance [Ohm]
La	0.0015	armature inductance [H]
Excitation		
IeNominal	1	nominal excitation current [A]
Re	100	warm field excitation resistance [Ohm]
Le	1	total field excitation inductance [H]

## Connectors

Name	Description
flange_a	
support	support at which the reaction torque is acting
pin_ap	
pin_an	
pin_ep	
pin_en	

## Modelica.Electrical.Machines.BasicMachines.DCMachines.DC\_SeriesExcited

Series excited linear DC machine



## Information

### Model of a DC Machine with Series excitation.

Armature resistance and inductance are modeled directly after the armature pins, then using a *AirGapDC* model.

Only losses in armature and excitation resistance are taken into account. No saturation is modelled.

Series excitation has to be connected by the user's external circuit.

**Default values for machine's parameters (a realistic example) are:**

stator's moment of inertia	0.29	kg.m2
rotor's moment of inertia	0.15	kg.m2
nominal armature voltage	100	V
nominal armature current	100	A
nominal torque	63.66	Nm

nominal speed	1410	rpm
nominal mechanical output	9.4	kW
efficiency	94.0	% only armature
armature resistance	0.05	Ohm in warm condition
armature inductance	0.0015	H
excitation resistance	0.01	Ohm in warm condition
excitation inductance	0.0005	H

Armature resistance resp. inductance include resistance resp. inductance of commutating pole winding and compensation windig, if present.

Parameter nominal armature voltage includes voltage drop of series excitation; but for output the voltage is splitted into:

va = armature voltage without voltage drop of series excitation

ve = voltage drop of series excitation

## Parameters

Name	Default	Description
J_Rotor	0.15	rotor's moment of inertia [kg.m <sup>2</sup> ]
J_Stator	J_Rotor	stator's moment of inertia [kg.m <sup>2</sup> ]
Nominal parameters		
VaNominal	100	nominal armature voltage [V]
IaNominal	100	nominal armature current [A]
rpmNominal	1410	nominal speed [rev/min]
Nominal resistances and inductances		
Ra	0.05	warm armature resistance [Ohm]
La	0.0015	armature inductance [H]
Excitation		
Re	0.01	warm field excitation resistance [Ohm]
Le	0.0005	total field excitation inductance [H]

## Connectors

Name	Description
flange_a	
support	support at which the reaction torque is acting
pin_ap	
pin_an	
pin_ep	
pin_en	

---

## Modelica.Electrical.Machines.BasicMachines.Transformers

### Library for technical 3phase transformers

## Information

This package contains components to model technical threephase transformers:

- Transformer: transformer model to choose connection / vector group
- Yy: Transformers with primary primary Y / secondary y

- Yd: Transformers with primary primary Y / secondary d
- Yz: Transformers with primary primary Y / secondary zig-zag
- Dy: Transformers with primary primary D / secondary y
- Dd: Transformers with primary D / secondary d
- Dz: Transformers with primary D / secondary zig-zag

Transformers are modeled by an ideal transformer, adding primary and secondary winding resistances and stray inductances.

All transformers extend from the base model *PartialTransformer*, adding the primary and secondary connection.

**VectorGroup** defines the phase shift between primary and secondary voltages, expressed by a number phase shift/30 degree (i.e. the hour on a clock face). Therefore each transformer is identified by two characters and a two-digit number, e.g. Yd11 ... primary connection Y (star), secondary connection d (delta), vector group 11 (phase shift 330 degree)

With the "supermodel" *Transformer* the user may choose primary and secondary connection as well as the vector group.

It calculates winding ratio as well as primary and secondary winding resistances and stray inductances, distributing them equally to primary and secondary winding, from the following parameters:

- nominal frequency
- primary voltage (RMS line-to-line)
- secondary voltage (RMS line-to-line)
- nominal apparent power
- impedance voltage drop
- short-circuit copper losses

The **impedance voltage drop** indicates the (absolute value of the) voltage drop at nominal load (current) as well as the voltage we have to apply to the primary winding to achieve nominal current in the short-circuited secondary winding.

**Please pay attention** to proper grounding of the primary and secondary part of the whole circuit.

The primary and secondary starpoint are available as connectors, if the connection is not delta (D or d).

**In some cases (Yy or Yz) it may be necessary to ground one of the transformer's starpoints even though the source's and/or load's starpoint are grounded; you may use a reasonable high earthing resistance.**

#### Limitations and assumptions:

- number of phases is limited to 3, therefore definition as a constant m=3
- symmetry of the 3 phases resp. limbs
- temperature dependency is neglected, i.e. resistances are constant
- saturation is neglected, i.e. inductances are constant
- magnetizing current is neglected
- magnetizing losses are neglected
- additional (stray) losses are neglected

#### Further development:

- modeling magnetizing current, including saturation
- temperature dependency of winding resistances

#### Main Authors:

[Anton Haumer](#)

Technical Consulting & Electrical Engineering  
A-3423 St.Andrae-Woerdern  
Austria  
email: [a.haumer@haumer.at](mailto:a.haumer@haumer.at)

Copyright © 1998-2007, Modelica Association and Anton Haumer.

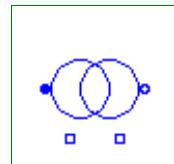
*The Modelica package is free software; it can be redistributed and/or modified under the terms of the Modelica license, see the license conditions and the accompanying disclaimer [here](#).*

## Package Content

Name	Description
 Transformer	Transformer: choose connection/vector group
 Yy	Transformers: primary Y / secondary y
 Yd	Transformers: primary Y / secondary d
 Yz	Transformers: primary Y / secondary zig-zag
 Dy	Transformers: primary D / secondary y
 Dd	Transformers: primary D / secondary d
 Dz	Transformers: primary D / secondary zig-zag

## Modelica.Electrical.Machines.BasicMachines.Transformers.Transformer

Transformer: choose connection/vector group



### Information

"Supertransformer": lets the user choose connection/vector group

### Parameters

Name	Default	Description
f	50	Nominal frequency [Hz]
V1	100	Primary nominal line-to-line voltage (RMS) [V]
V2	100	Secondary open circuit line-to-line voltage (RMS) @ primary nominal voltage [V]
SNominal	30E3	Nominal apparent power [VA]
v_sc	0.05	Impedance voltage drop pu
P_sc	300	Short-circuit (copper) losses [W]
core	redeclare IdealCore core(fin...	

### Connectors

Name	Description
plug1	
plug2	
starpoint1	
starpoint2	

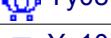
## Modelica.Electrical.Machines.BasicMachines.Transformers.Yy

Transformers: primary Y / secondary y

### Information

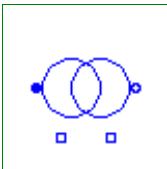
This package contains transformers primary Y connected / secondary y connected in all possible vector groups.

## Package Content

Name	Description
 Yy00	Transformer Yy0
 Yy02	Transformer Yy2
 Yy04	Transformer Yy4
 Yy06	Transformer Yy6
 Yy08	Transformer Yy8
 Yy10	Transformer Yy10

## Modelica.Electrical.Machines.BasicMachines.Transformers.Yy.Yy00

### Transformer Yy0



#### Information

Transformer Yy0

#### Parameters

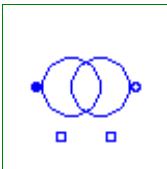
Name	Default	Description
n	1	Primary voltage (line-to-line) / Secondary voltage (line-to-line)
R1	5E-3/(if C1 == "D" then 1 else 0.5)	Warm primary resistance per phase [Ohm]
L1sigma	78E-6/(if C1 == "D" then 1 else 0.00078)	Primary stray inductance per phase [H]
R2	5E-3/(if C2 == "d" then 1 else 0.5)	Warm secondary resistance per phase [Ohm]
L2sigma	78E-6/(if C2 == "d" then 1 else 0.00078)	Secondary stray inductance per phase [H]
core	redeclare IdealCore core(fin...	

#### Connectors

Name	Description
plug1	
plug2	
starpoint1	
starpoint2	

## Modelica.Electrical.Machines.BasicMachines.Transformers.Yy.Yy02

### Transformer Yy2



#### Information

Transformer Yy2

#### Parameters

Name	Default	Description
n	1	Primary voltage (line-to-line) / Secondary voltage (line-to-line)

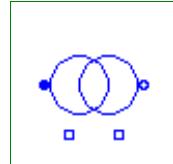
R1	5E-3/(if C1 == "D" then 1 else 0)	Warm primary resistance per phase [Ohm]
L1sigma	78E-6/(if C1 == "D" then 1 else 0)	Primary stray inductance per phase [H]
R2	5E-3/(if C2 == "d" then 1 else 0)	Warm secondary resistance per phase [Ohm]
L2sigma	78E-6/(if C2 == "d" then 1 else 0)	Secondary stray inductance per phase [H]
core	redeclare IdealCore core(financial)	

## Connectors

Name	Description
plug1	
plug2	
starpoint1	
starpoint2	

## Modelica.Electrical.Machines.BasicMachines.Transformers.Yy.Yy04

Transformer Yy4



## Information

Transformer Yy4

## Parameters

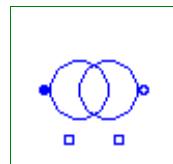
Name	Default	Description
n	1	Primary voltage (line-to-line) / Secondary voltage (line-to-line)
R1	5E-3/(if C1 == "D" then 1 else 0)	Warm primary resistance per phase [Ohm]
L1sigma	78E-6/(if C1 == "D" then 1 else 0)	Primary stray inductance per phase [H]
R2	5E-3/(if C2 == "d" then 1 else 0)	Warm secondary resistance per phase [Ohm]
L2sigma	78E-6/(if C2 == "d" then 1 else 0)	Secondary stray inductance per phase [H]
core	redeclare IdealCore core(financial)	

## Connectors

Name	Description
plug1	
plug2	
starpoint1	
starpoint2	

## Modelica.Electrical.Machines.BasicMachines.Transformers.Yy.Yy06

Transformer Yy6



## Information

Transformer Yy6

## Parameters

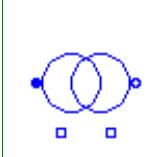
Name	Default	Description
n	1	Primary voltage (line-to-line) / Secondary voltage (line-to-line)
R1	5E-3/(if C1 == "D" then 1 el...)	Warm primary resistance per phase [Ohm]
L1sigma	78E-6/(if C1 == "D" then 1 e...)	Primary stray inductance per phase [H]
R2	5E-3/(if C2 == "d" then 1 el...)	Warm secondary resistance per phase [Ohm]
L2sigma	78E-6/(if C2 == "d" then 1 e...)	Secondary stray inductance per phase [H]
core	redeclare IdealCore core(fin...	

## Connectors

Name	Description
plug1	
plug2	
starpoint1	
starpoint2	

## Modelica.Electrical.Machines.BasicMachines.Transformers.Yy.Yy08

### Transformer Yy8



## Information

Transformer Yy8

## Parameters

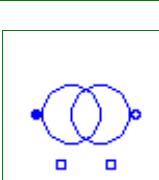
Name	Default	Description
n	1	Primary voltage (line-to-line) / Secondary voltage (line-to-line)
R1	5E-3/(if C1 == "D" then 1 el...)	Warm primary resistance per phase [Ohm]
L1sigma	78E-6/(if C1 == "D" then 1 e...)	Primary stray inductance per phase [H]
R2	5E-3/(if C2 == "d" then 1 el...)	Warm secondary resistance per phase [Ohm]
L2sigma	78E-6/(if C2 == "d" then 1 e...)	Secondary stray inductance per phase [H]
core	redeclare IdealCore core(fin...	

## Connectors

Name	Description
plug1	
plug2	
starpoint1	
starpoint2	

## Modelica.Electrical.Machines.BasicMachines.Transformers.Yy.Yy10

### Transformer Yy10



## Information

Transformer Yy10

## Parameters

Name	Default	Description
n	1	Primary voltage (line-to-line) / Secondary voltage (line-to-line)
R1	5E-3/(if C1 == "D" then 1 else 0)	Warm primary resistance per phase [Ohm]
L1sigma	78E-6/(if C1 == "D" then 1 else 0)	Primary stray inductance per phase [H]
R2	5E-3/(if C2 == "d" then 1 else 0)	Warm secondary resistance per phase [Ohm]
L2sigma	78E-6/(if C2 == "d" then 1 else 0)	Secondary stray inductance per phase [H]
core	redeclare IdealCore core(fin...	

## Connectors

Name	Description
plug1	
plug2	
starpoint1	
starpoint2	

## Modelica.Electrical.Machines.BasicMachines.Transformers.Yd

Transformers: primary Y / secondary d

## Information

This package contains transformers primary Y connected / secondary d connected in all possible vector groups.

## Package Content

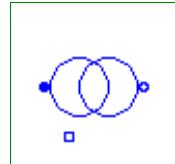
Name	Description
 Yd01	Transformer Yd1
 Yd03	Transformer Yd3
 Yd05	Transformer Yd5
 Yd07	Transformer Yd7
 Yd09	Transformer Yd9
 Yd11	Transformer Yd11

## Modelica.Electrical.Machines.BasicMachines.Transformers.Yd.Yd01

Transformer Yd1

## Information

Transformer Yd1



## Parameters

Name	Default	Description
n	1	Primary voltage (line-to-line) / Secondary voltage (line-to-line)
R1	5E-3/(if C1 == "D" then 1 el...)	Warm primary resistance per phase [Ohm]
L1sigma	78E-6/(if C1 == "D" then 1 e...)	Primary stray inductance per phase [H]
R2	5E-3/(if C2 == "d" then 1 el...)	Warm secondary resistance per phase [Ohm]
L2sigma	78E-6/(if C2 == "d" then 1 e...)	Secondary stray inductance per phase [H]
core	redeclare IdealCore core(fin...	

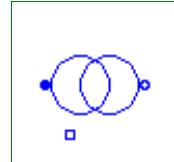
## Connectors

Name	Description
plug1	
plug2	
starpoint1	

---

## Modelica.Electrical.Machines.BasicMachines.Transformers.Yd.Yd03

Transformer Yd3



## Information

Transformer Yd3

## Parameters

Name	Default	Description
n	1	Primary voltage (line-to-line) / Secondary voltage (line-to-line)
R1	5E-3/(if C1 == "D" then 1 el...)	Warm primary resistance per phase [Ohm]
L1sigma	78E-6/(if C1 == "D" then 1 e...)	Primary stray inductance per phase [H]
R2	5E-3/(if C2 == "d" then 1 el...)	Warm secondary resistance per phase [Ohm]
L2sigma	78E-6/(if C2 == "d" then 1 e...)	Secondary stray inductance per phase [H]
core	redeclare IdealCore core(fin...	

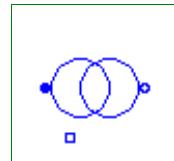
## Connectors

Name	Description
plug1	
plug2	
starpoint1	

---

## Modelica.Electrical.Machines.BasicMachines.Transformers.Yd.Yd05

Transformer Yd5



## Information

Transformer Yd5

## Parameters

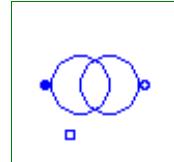
Name	Default	Description
n	1	Primary voltage (line-to-line) / Secondary voltage (line-to-line)
R1	5E-3/(if C1 == "D" then 1 el...)	Warm primary resistance per phase [Ohm]
L1sigma	78E-6/(if C1 == "D" then 1 e...)	Primary stray inductance per phase [H]
R2	5E-3/(if C2 == "d" then 1 el...)	Warm secondary resistance per phase [Ohm]
L2sigma	78E-6/(if C2 == "d" then 1 e...)	Secondary stray inductance per phase [H]
core	redeclare IdealCore core(fin...	

## Connectors

Name	Description
plug1	
plug2	
starpoint1	

## Modelica.Electrical.Machines.BasicMachines.Transformers.Yd.Yd07

Transformer Yd7



## Information

Transformer Yd7

## Parameters

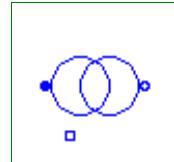
Name	Default	Description
n	1	Primary voltage (line-to-line) / Secondary voltage (line-to-line)
R1	5E-3/(if C1 == "D" then 1 el...)	Warm primary resistance per phase [Ohm]
L1sigma	78E-6/(if C1 == "D" then 1 e...)	Primary stray inductance per phase [H]
R2	5E-3/(if C2 == "d" then 1 el...)	Warm secondary resistance per phase [Ohm]
L2sigma	78E-6/(if C2 == "d" then 1 e...)	Secondary stray inductance per phase [H]
core	redeclare IdealCore core(fin...	

## Connectors

Name	Description
plug1	
plug2	
starpoint1	

## Modelica.Electrical.Machines.BasicMachines.Transformers.Yd.Yd09

Transformer Yd9



## Information

Transformer Yd9

## Parameters

Name	Default	Description
n	1	Primary voltage (line-to-line) / Secondary voltage (line-to-line)
R1	5E-3/(if C1 == "D" then 1 el...)	Warm primary resistance per phase [Ohm]
L1sigma	78E-6/(if C1 == "D" then 1 e...)	Primary stray inductance per phase [H]
R2	5E-3/(if C2 == "d" then 1 el...)	Warm secondary resistance per phase [Ohm]
L2sigma	78E-6/(if C2 == "d" then 1 e...)	Secondary stray inductance per phase [H]
core	redeclare IdealCore core(fin...	

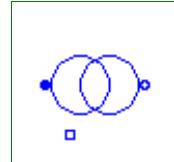
## Connectors

Name	Description
plug1	
plug2	
starpoint1	

---

## Modelica.Electrical.Machines.BasicMachines.Transformers.Yd.Yd11

### Transformer Yd11



## Information

Transformer Yd11

## Parameters

Name	Default	Description
n	1	Primary voltage (line-to-line) / Secondary voltage (line-to-line)
R1	5E-3/(if C1 == "D" then 1 el...)	Warm primary resistance per phase [Ohm]
L1sigma	78E-6/(if C1 == "D" then 1 e...)	Primary stray inductance per phase [H]
R2	5E-3/(if C2 == "d" then 1 el...)	Warm secondary resistance per phase [Ohm]
L2sigma	78E-6/(if C2 == "d" then 1 e...)	Secondary stray inductance per phase [H]
core	redeclare IdealCore core(fin...	

## Connectors

Name	Description
plug1	
plug2	
starpoint1	

---

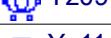
## Modelica.Electrical.Machines.BasicMachines.Transformers.Yz

### Transformers: primary Y / secondary zig-zag

## Information

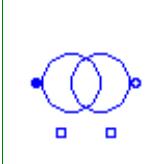
This package contains transformers primary Y connected / secondary zig-zag connected in all possible vector groups.

## Package Content

Name	Description
 Yz01	Transformer Yz1
 Yz03	Transformer Yz3
 Yz05	Transformer Yz5
 Yz07	Transformer Yz7
 Yz09	Transformer Yz9
 Yz11	Transformer Yz11

## Modelica.Electrical.Machines.BasicMachines.Transformers.Yz.Yz01

### Transformer Yz1



#### Information

Transformer Yz1

#### Parameters

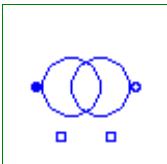
Name	Default	Description
n	1	Primary voltage (line-to-line) / Secondary voltage (line-to-line)
R1	5E-3/(if C1 == "D" then 1 else 0)	Warm primary resistance per phase [Ohm]
L1sigma	78E-6/(if C1 == "D" then 1 else 0)	Primary stray inductance per phase [H]
R2	5E-3/(if C2 == "d" then 1 else 0)	Warm secondary resistance per phase [Ohm]
L2sigma	78E-6/(if C2 == "d" then 1 else 0)	Secondary stray inductance per phase [H]
core	redeclare IdealCore core(fin...	

#### Connectors

Name	Description
plug1	
plug2	
starpoint1	
starpoint2	

## Modelica.Electrical.Machines.BasicMachines.Transformers.Yz.Yz03

### Transformer Yz3



#### Information

Transformer Yz3

#### Parameters

Name	Default	Description
n	1	Primary voltage (line-to-line) / Secondary voltage (line-to-line)

## 320 Modelica.Electrical.Machines.BasicMachines.Transformers.Yz.Yz03

R1	5E-3/(if C1 == "D" then 1 el...	Warm primary resistance per phase [Ohm]
L1sigma	78E-6/(if C1 == "D" then 1 e...	Primary stray inductance per phase [H]
R2	5E-3/(if C2 == "d" then 1 el...	Warm secondary resistance per phase [Ohm]
L2sigma	78E-6/(if C2 == "d" then 1 e...	Secondary stray inductance per phase [H]
core	redeclare IdealCore core(fin...	

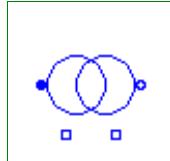
### Connectors

Name	Description
plug1	
plug2	
starpoint1	
starpoint2	

---

## Modelica.Electrical.Machines.BasicMachines.Transformers.Yz.Yz05

### Transformer Yz5



### Information

Transformer Yz5

### Parameters

Name	Default	Description
n	1	Primary voltage (line-to-line) / Secondary voltage (line-to-line)
R1	5E-3/(if C1 == "D" then 1 el...	Warm primary resistance per phase [Ohm]
L1sigma	78E-6/(if C1 == "D" then 1 e...	Primary stray inductance per phase [H]
R2	5E-3/(if C2 == "d" then 1 el...	Warm secondary resistance per phase [Ohm]
L2sigma	78E-6/(if C2 == "d" then 1 e...	Secondary stray inductance per phase [H]
core	redeclare IdealCore core(fin...	

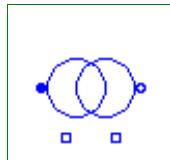
### Connectors

Name	Description
plug1	
plug2	
starpoint1	
starpoint2	

---

## Modelica.Electrical.Machines.BasicMachines.Transformers.Yz.Yz07

### Transformer Yz7



### Information

Transformer Yz7

## Parameters

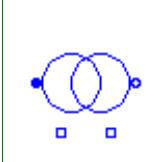
Name	Default	Description
n	1	Primary voltage (line-to-line) / Secondary voltage (line-to-line)
R1	5E-3/(if C1 == "D" then 1 el...)	Warm primary resistance per phase [Ohm]
L1sigma	78E-6/(if C1 == "D" then 1 e...)	Primary stray inductance per phase [H]
R2	5E-3/(if C2 == "d" then 1 el...)	Warm secondary resistance per phase [Ohm]
L2sigma	78E-6/(if C2 == "d" then 1 e...)	Secondary stray inductance per phase [H]
core	redeclare IdealCore core(fin...	

## Connectors

Name	Description
plug1	
plug2	
starpoint1	
starpoint2	

## Modelica.Electrical.Machines.BasicMachines.Transformers.Yz.Yz09

### Transformer Yz9



## Information

Transformer Yz9

## Parameters

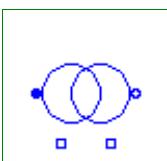
Name	Default	Description
n	1	Primary voltage (line-to-line) / Secondary voltage (line-to-line)
R1	5E-3/(if C1 == "D" then 1 el...)	Warm primary resistance per phase [Ohm]
L1sigma	78E-6/(if C1 == "D" then 1 e...)	Primary stray inductance per phase [H]
R2	5E-3/(if C2 == "d" then 1 el...)	Warm secondary resistance per phase [Ohm]
L2sigma	78E-6/(if C2 == "d" then 1 e...)	Secondary stray inductance per phase [H]
core	redeclare IdealCore core(fin...	

## Connectors

Name	Description
plug1	
plug2	
starpoint1	
starpoint2	

## Modelica.Electrical.Machines.BasicMachines.Transformers.Yz.Yz11

### Transformer Yz11



## Information

Transformer Yz11

## Parameters

Name	Default	Description
n	1	Primary voltage (line-to-line) / Secondary voltage (line-to-line)
R1	5E-3/(if C1 == "D" then 1 else 0)	Warm primary resistance per phase [Ohm]
L1sigma	78E-6/(if C1 == "D" then 1 else 0)	Primary stray inductance per phase [H]
R2	5E-3/(if C2 == "d" then 1 else 0)	Warm secondary resistance per phase [Ohm]
L2sigma	78E-6/(if C2 == "d" then 1 else 0)	Secondary stray inductance per phase [H]
core	redeclare IdealCore core(fin...	

## Connectors

Name	Description
plug1	
plug2	
starpoint1	
starpoint2	

---

## Modelica.Electrical.Machines.BasicMachines.Transformers.Dy

Transformers: primary D / secondary y

## Information

This package contains transformers primary D connected / secondary y connected in all possible vector groups.

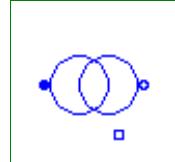
## Package Content

Name	Description
 Dy01	Transformer Dy1
 Dy03	Transformer Dy3
 Dy05	Transformer Dy5
 Dy07	Transformer Dy7
 Dy09	Transformer Dy9
 Dy11	Transformer Dy11

---

## Modelica.Electrical.Machines.BasicMachines.Transformers.Dy.Dy01

Transformer Dy1



## Information

Transformer Dy1

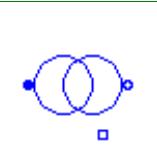
## Parameters

Name	Default	Description
n	1	Primary voltage (line-to-line) / Secondary voltage (line-to-line)
R1	5E-3/(if C1 == "D" then 1 el...)	Warm primary resistance per phase [Ohm]
L1sigma	78E-6/(if C1 == "D" then 1 e...)	Primary stray inductance per phase [H]
R2	5E-3/(if C2 == "d" then 1 el...)	Warm secondary resistance per phase [Ohm]
L2sigma	78E-6/(if C2 == "d" then 1 e...)	Secondary stray inductance per phase [H]
core	redeclare IdealCore core(fin...	

## Connectors

Name	Description
plug1	
plug2	
starpoint2	

## Modelica.Electrical.Machines.BasicMachines.Transformers.Dy.Dy03



### Transformer Dy3

## Information

Transformer Dy3

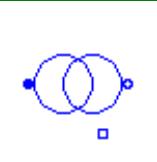
## Parameters

Name	Default	Description
n	1	Primary voltage (line-to-line) / Secondary voltage (line-to-line)
R1	5E-3/(if C1 == "D" then 1 el...)	Warm primary resistance per phase [Ohm]
L1sigma	78E-6/(if C1 == "D" then 1 e...)	Primary stray inductance per phase [H]
R2	5E-3/(if C2 == "d" then 1 el...)	Warm secondary resistance per phase [Ohm]
L2sigma	78E-6/(if C2 == "d" then 1 e...)	Secondary stray inductance per phase [H]
core	redeclare IdealCore core(fin...	

## Connectors

Name	Description
plug1	
plug2	
starpoint2	

## Modelica.Electrical.Machines.BasicMachines.Transformers.Dy.Dy05



### Transformer Dy5

## Information

Transformer Dy5

## Parameters

Name	Default	Description
n	1	Primary voltage (line-to-line) / Secondary voltage (line-to-line)
R1	5E-3/(if C1 == "D" then 1 el...)	Warm primary resistance per phase [Ohm]
L1sigma	78E-6/(if C1 == "D" then 1 e...)	Primary stray inductance per phase [H]
R2	5E-3/(if C2 == "d" then 1 el...)	Warm secondary resistance per phase [Ohm]
L2sigma	78E-6/(if C2 == "d" then 1 e...)	Secondary stray inductance per phase [H]
core	redeclare IdealCore core(fin...	

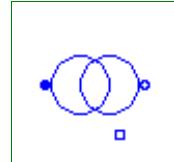
## Connectors

Name	Description
plug1	
plug2	
starpoint2	

---

## Modelica.Electrical.Machines.BasicMachines.Transformers.Dy.Dy07

Transformer Dy7



## Information

Transformer Dy7

## Parameters

Name	Default	Description
n	1	Primary voltage (line-to-line) / Secondary voltage (line-to-line)
R1	5E-3/(if C1 == "D" then 1 el...)	Warm primary resistance per phase [Ohm]
L1sigma	78E-6/(if C1 == "D" then 1 e...)	Primary stray inductance per phase [H]
R2	5E-3/(if C2 == "d" then 1 el...)	Warm secondary resistance per phase [Ohm]
L2sigma	78E-6/(if C2 == "d" then 1 e...)	Secondary stray inductance per phase [H]
core	redeclare IdealCore core(fin...	

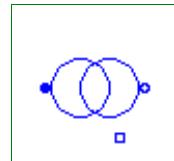
## Connectors

Name	Description
plug1	
plug2	
starpoint2	

---

## Modelica.Electrical.Machines.BasicMachines.Transformers.Dy.Dy09

Transformer Dy9



## Information

Transformer Dy9

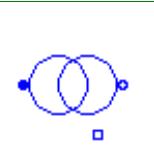
## Parameters

Name	Default	Description
n	1	Primary voltage (line-to-line) / Secondary voltage (line-to-line)
R1	5E-3/(if C1 == "D" then 1 el...)	Warm primary resistance per phase [Ohm]
L1sigma	78E-6/(if C1 == "D" then 1 e...)	Primary stray inductance per phase [H]
R2	5E-3/(if C2 == "d" then 1 el...)	Warm secondary resistance per phase [Ohm]
L2sigma	78E-6/(if C2 == "d" then 1 e...)	Secondary stray inductance per phase [H]
core	redeclare IdealCore core(fin...	

## Connectors

Name	Description
plug1	
plug2	
starpoint2	

## Modelica.Electrical.Machines.BasicMachines.Transformers.Dy.Dy11



### Transformer Dy11

## Information

Transformer Dy11

## Parameters

Name	Default	Description
n	1	Primary voltage (line-to-line) / Secondary voltage (line-to-line)
R1	5E-3/(if C1 == "D" then 1 el...)	Warm primary resistance per phase [Ohm]
L1sigma	78E-6/(if C1 == "D" then 1 e...)	Primary stray inductance per phase [H]
R2	5E-3/(if C2 == "d" then 1 el...)	Warm secondary resistance per phase [Ohm]
L2sigma	78E-6/(if C2 == "d" then 1 e...)	Secondary stray inductance per phase [H]
core	redeclare IdealCore core(fin...	

## Connectors

Name	Description
plug1	
plug2	
starpoint2	

## Modelica.Electrical.Machines.BasicMachines.Transformers.Dd

### Transformers: primary D / secondary d

## Information

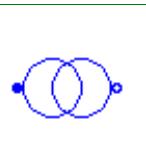
This package contains transformers primary D connected / secondary d connected in all possible vector groups.

## Package Content

Name	Description
Dd00	Transformer Dd0
Dd02	Transformer Dd2
Dd04	Transformer Dd4
Dd06	Transformer Dd6
Dd08	Transformer Dd8
Dd10	Transformer Dd10

## Modelica.Electrical.Machines.BasicMachines.Transformers.Dd.Dd00

### Transformer Dd0



#### Information

Transformer Dd0

#### Parameters

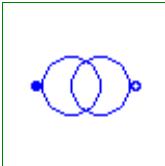
Name	Default	Description
n	1	Primary voltage (line-to-line) / Secondary voltage (line-to-line)
R1	5E-3/(if C1 == "D" then 1 el...)	Warm primary resistance per phase [Ohm]
L1sigma	78E-6/(if C1 == "D" then 1 e...)	Primary stray inductance per phase [H]
R2	5E-3/(if C2 == "d" then 1 el...)	Warm secondary resistance per phase [Ohm]
L2sigma	78E-6/(if C2 == "d" then 1 e...)	Secondary stray inductance per phase [H]
core	redeclare IdealCore core(fin...	

#### Connectors

Name	Description
plug1	
plug2	

## Modelica.Electrical.Machines.BasicMachines.Transformers.Dd.Dd02

### Transformer Dd2



#### Information

Transformer Dd2

#### Parameters

Name	Default	Description
n	1	Primary voltage (line-to-line) / Secondary voltage (line-to-line)
R1	5E-3/(if C1 == "D" then 1 el...)	Warm primary resistance per phase [Ohm]
L1sigma	78E-6/(if C1 == "D" then 1 e...)	Primary stray inductance per phase [H]

R2	$5E-3/(if C2 == "d" then 1 else 0)$	Warm secondary resistance per phase [Ohm]
L2sigma	$78E-6/(if C2 == "d" then 1 else 0)$	Secondary stray inductance per phase [H]
core	redeclare IdealCore core(fin...	

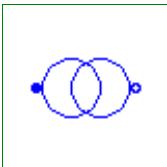
## Connectors

Name	Description
plug1	
plug2	

---

## Modelica.Electrical.Machines.BasicMachines.Transformers.Dd.Dd04

Transformer Dd4



## Information

Transformer Dd4

## Parameters

Name	Default	Description
n	1	Primary voltage (line-to-line) / Secondary voltage (line-to-line)
R1	$5E-3/(if C1 == "D" then 1 else 0)$	Warm primary resistance per phase [Ohm]
L1sigma	$78E-6/(if C1 == "D" then 1 else 0)$	Primary stray inductance per phase [H]
R2	$5E-3/(if C2 == "d" then 1 else 0)$	Warm secondary resistance per phase [Ohm]
L2sigma	$78E-6/(if C2 == "d" then 1 else 0)$	Secondary stray inductance per phase [H]
core	redeclare IdealCore core(fin...	

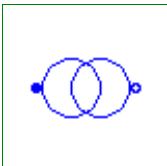
## Connectors

Name	Description
plug1	
plug2	

---

## Modelica.Electrical.Machines.BasicMachines.Transformers.Dd.Dd06

Transformer Dd6



## Information

Transformer Dd6

## Parameters

Name	Default	Description
n	1	Primary voltage (line-to-line) / Secondary voltage (line-to-line)
R1	$5E-3/(if C1 == "D" then 1 else 0)$	Warm primary resistance per phase [Ohm]
L1sigma	$78E-6/(if C1 == "D" then 1 else 0)$	Primary stray inductance per phase [H]
R2	$5E-3/(if C2 == "d" then 1 else 0)$	Warm secondary resistance per phase [Ohm]
L2sigma	$78E-6/(if C2 == "d" then 1 else 0)$	Secondary stray inductance per phase [H]

## 328 Modelica.Electrical.Machines.BasicMachines.Transformers.Dd.Dd06

---

core	redeclare IdealCore core(fin...
------	---------------------------------

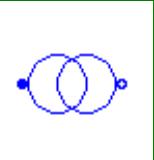
### Connectors

Name	Description
plug1	
plug2	

---

## Modelica.Electrical.Machines.BasicMachines.Transformers.Dd.Dd08

Transformer Dd8



### Information

Transformer Dd8

### Parameters

Name	Default	Description
n	1	Primary voltage (line-to-line) / Secondary voltage (line-to-line)
R1	5E-3/(if C1 == "D" then 1 el...	Warm primary resistance per phase [Ohm]
L1sigma	78E-6/(if C1 == "D" then 1 e...	Primary stray inductance per phase [H]
R2	5E-3/(if C2 == "d" then 1 el...	Warm secondary resistance per phase [Ohm]
L2sigma	78E-6/(if C2 == "d" then 1 e...	Secondary stray inductance per phase [H]
core	redeclare IdealCore core(fin...	

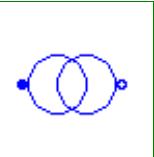
### Connectors

Name	Description
plug1	
plug2	

---

## Modelica.Electrical.Machines.BasicMachines.Transformers.Dd.Dd10

Transformer Dd10



### Information

Transformer Dd10

### Parameters

Name	Default	Description
n	1	Primary voltage (line-to-line) / Secondary voltage (line-to-line)
R1	5E-3/(if C1 == "D" then 1 el...	Warm primary resistance per phase [Ohm]
L1sigma	78E-6/(if C1 == "D" then 1 e...	Primary stray inductance per phase [H]
R2	5E-3/(if C2 == "d" then 1 el...	Warm secondary resistance per phase [Ohm]
L2sigma	78E-6/(if C2 == "d" then 1 e...	Secondary stray inductance per phase [H]
core	redeclare IdealCore core(fin...	

## Connectors

Name	Description
plug1	
plug2	

## Modelica.Electrical.Machines.BasicMachines.Transformers.Dz

Transformers: primary D / secondary zig-zag

## Information

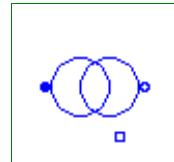
This package contains transformers primary D connected / secondary d connected in all possible vector groups.

## Package Content

Name	Description
Dz00	Transformer Dz0
Dz02	Transformer Dz2
Dz04	Transformer Dz4
Dz06	Transformer Dz6
Dz08	Transformer Dz8
Dz10	Transformer Dz10

## Modelica.Electrical.Machines.BasicMachines.Transformers.Dz.Dz00

Transformer Dz0



## Information

Transformer Dz0

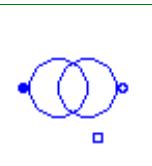
## Parameters

Name	Default	Description
n	1	Primary voltage (line-to-line) / Secondary voltage (line-to-line)
R1	5E-3/(if C1 == "D" then 1 else 0)	Warm primary resistance per phase [Ohm]
L1sigma	78E-6/(if C1 == "D" then 1 else 0)	Primary stray inductance per phase [H]
R2	5E-3/(if C2 == "d" then 1 else 0)	Warm secondary resistance per phase [Ohm]
L2sigma	78E-6/(if C2 == "d" then 1 else 0)	Secondary stray inductance per phase [H]
core	redeclare IdealCore core(first)	

## Connectors

Name	Description
plug1	
plug2	

starpoint2	
------------	--

**Modelica.Electrical.Machines.BasicMachines.Transformers.Dz.Dz02****Transformer Dz2****Information**

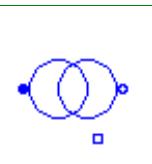
Transformer Dz2

**Parameters**

Name	Default	Description
n	1	Primary voltage (line-to-line) / Secondary voltage (line-to-line)
R1	5E-3/(if C1 == "D" then 1 el...)	Warm primary resistance per phase [Ohm]
L1sigma	78E-6/(if C1 == "D" then 1 e...)	Primary stray inductance per phase [H]
R2	5E-3/(if C2 == "d" then 1 el...)	Warm secondary resistance per phase [Ohm]
L2sigma	78E-6/(if C2 == "d" then 1 e...)	Secondary stray inductance per phase [H]
core	redeclare IdealCore core(fin...	

**Connectors**

Name	Description
plug1	
plug2	
starpoint2	

**Modelica.Electrical.Machines.BasicMachines.Transformers.Dz.Dz04****Transformer Dz4****Information**

Transformer Dz4

**Parameters**

Name	Default	Description
n	1	Primary voltage (line-to-line) / Secondary voltage (line-to-line)
R1	5E-3/(if C1 == "D" then 1 el...)	Warm primary resistance per phase [Ohm]
L1sigma	78E-6/(if C1 == "D" then 1 e...)	Primary stray inductance per phase [H]
R2	5E-3/(if C2 == "d" then 1 el...)	Warm secondary resistance per phase [Ohm]
L2sigma	78E-6/(if C2 == "d" then 1 e...)	Secondary stray inductance per phase [H]
core	redeclare IdealCore core(fin...	

**Connectors**

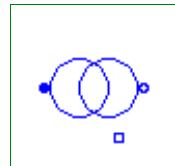
Name	Description
plug1	
plug2	

starpoint2	
------------	--

---

## Modelica.Electrical.Machines.BasicMachines.Transformers.Dz.Dz06

### Transformer Dz6



#### Information

Transformer Dz6

#### Parameters

Name	Default	Description
n	1	Primary voltage (line-to-line) / Secondary voltage (line-to-line)
R1	5E-3/(if C1 == "D" then 1 el...)	Warm primary resistance per phase [Ohm]
L1sigma	78E-6/(if C1 == "D" then 1 e...)	Primary stray inductance per phase [H]
R2	5E-3/(if C2 == "d" then 1 el...)	Warm secondary resistance per phase [Ohm]
L2sigma	78E-6/(if C2 == "d" then 1 e...)	Secondary stray inductance per phase [H]
core	redeclare IdealCore core(fin...	

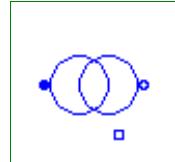
#### Connectors

Name	Description
plug1	
plug2	
starpoint2	

---

## Modelica.Electrical.Machines.BasicMachines.Transformers.Dz.Dz08

### Transformer Dz8



#### Information

Transformer Dz8

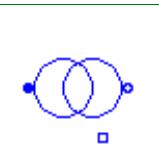
#### Parameters

Name	Default	Description
n	1	Primary voltage (line-to-line) / Secondary voltage (line-to-line)
R1	5E-3/(if C1 == "D" then 1 el...)	Warm primary resistance per phase [Ohm]
L1sigma	78E-6/(if C1 == "D" then 1 e...)	Primary stray inductance per phase [H]
R2	5E-3/(if C2 == "d" then 1 el...)	Warm secondary resistance per phase [Ohm]
L2sigma	78E-6/(if C2 == "d" then 1 e...)	Secondary stray inductance per phase [H]
core	redeclare IdealCore core(fin...	

#### Connectors

Name	Description
plug1	
plug2	

starpoint2	
------------	--

**Modelica.Electrical.Machines.BasicMachines.Transformers.Dz.Dz10****Transformer Dz10****Information**

Transformer Dz10

**Parameters**

Name	Default	Description
n	1	Primary voltage (line-to-line) / Secondary voltage (line-to-line)
R1	5E-3/(if C1 == "D" then 1 else 0)	Warm primary resistance per phase [Ohm]
L1sigma	78E-6/(if C1 == "D" then 1 else 0)	Primary stray inductance per phase [H]
R2	5E-3/(if C2 == "d" then 1 else 0)	Warm secondary resistance per phase [Ohm]
L2sigma	78E-6/(if C2 == "d" then 1 else 0)	Secondary stray inductance per phase [H]
core	redeclare IdealCore core(fin...	

**Connectors**

Name	Description
plug1	
plug2	
starpoint2	

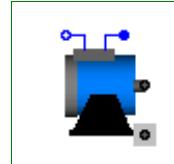
**Modelica.Electrical.Machines.BasicMachines.Components****Machine components like AirGaps****Information**

This package contains components for modeling electrical machines, specially threephase induction machines, based on space phasor theory. These models use package SpacePhasors.

**Package Content**

Name	Description
BasicAIM	Partial model for asynchronous induction machine
BasicSM	Partial model for synchronous induction machine
PartialAirGap	Partial airgap model
AirGapS	Airgap in stator-fixed coordinate system
AirGapR	Airgap in rotor-fixed coordinate system
SquirrelCage	Squirrel Cage
DamperCage	Squirrel Cage
ElectricalExcitation	Electrical excitation

 PermanentMagnet	Permanent magnet excitation
 BasicDCMachine	Partial model for DC machine
 PartialAirGapDC	Partial airgap model of a DC machine
 AirGapDC	Linear airgap model of a DC machine
 BasicTransformer	Partial model of threephase transformer
 PartialCore	Partial model of transformer core with 3 windings
 IdealCore	Ideal transformer with 3 windings

**Modelica.Electrical.Machines.BasicMachines.Components.BasicAIM****Partial model for asynchronous induction machine****Information**

Partial model for induction machine models, containing:

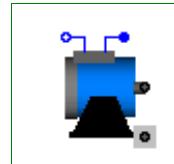
- main parts of the icon
- stator plugs
- mechanical connectors

**Parameters**

Name	Default	Description
J_Rotor	0.29	rotor's moment of inertia [kg.m <sup>2</sup> ]
J_Stator	J_Rotor	stator's moment of inertia [kg.m <sup>2</sup> ]
p	2	number of pole pairs (Integer)
fNominal	50	nominal frequency [Hz]
airGapS	redeclare AirGapS airGapS(f...)	
Nominal resistances and inductances		
Rs	0.03	warm stator resistance per phase [Ohm]
Lssigma	$3*(1 - \sqrt{1 - 0.0667})/(2 * \pi * f_{\text{Nominal}})$	stator stray inductance per phase [H]

**Connectors**

Name	Description
flange_a	
support	support at which the reaction torque is acting
plug_sp	
plug_sn	

**Modelica.Electrical.Machines.BasicMachines.Components.BasicSM****Partial model for synchronous induction machine****Information**

Partial model for induction machine models, containing:

- main parts of the icon

- stator plugs
- mechanical connectors

## Parameters

Name	Default	Description
J_Rotor	0.29	rotor's moment of inertia [kg.m2]
J_Stator	J_Rotor	stator's moment of inertia [kg.m2]
p	2	number of pole pairs (Integer)
fNominal	50	nominal frequency [Hz]
airGapR	redeclare AirGapR airGapR(fi...	
Nominal resistances and inductances		
Rs	0.03	warm stator resistance per phase [Ohm]
Lsigma	$3*(1 - \sqrt{1 - 0.0667})/(2 * ...$	stator stray inductance per phase [H]

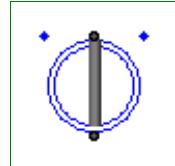
## Connectors

Name	Description
flange_a	
support	support at which the reaction torque is acting
plug_sp	
plug_sn	

---

## Modelica.Electrical.Machines.BasicMachines.Components.PartialAirGap

Partial airgap model



## Information

Partial model of the airgap, using only equations.

## Parameters

Name	Default	Description
m	3	number of phases
p		number of pole pairs

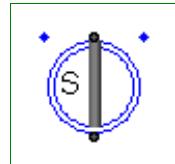
## Connectors

Name	Description
flange_a	
support	support at which the reaction torque is acting
spacePhasor_s	
spacePhasor_r	

---

## Modelica.Electrical.Machines.BasicMachines.Components.AirGapS

Airgap in stator-fixed coordinate system



## Information

Model of the airgap in stator-fixed coordinate system, using only equations.

## Parameters

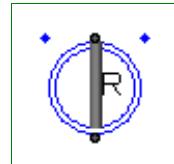
Name	Default	Description
Lm		main field inductance [H]
m	3	number of phases
p		number of pole pairs

## Connectors

Name	Description
flange_a	
support	support at which the reaction torque is acting
spacePhasor_s	
spacePhasor_r	

## Modelica.Electrical.Machines.BasicMachines.Components.AirGapR

Airgap in rotor-fixed coordinate system



## Information

Model of the airgap in rotor-fixed coordinate system, using only equations.

## Parameters

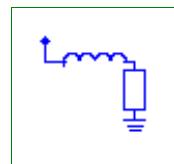
Name	Default	Description
Lmd		main field inductance d-axis [H]
Lmq		main field inductance q-axis [H]
m	3	number of phases
p		number of pole pairs

## Connectors

Name	Description
flange_a	
support	support at which the reaction torque is acting
spacePhasor_s	
spacePhasor_r	

## Modelica.Electrical.Machines.BasicMachines.Components.SquirrelCage

Squirrel Cage



## Information

Model of a squirrel cage / damper cage in two axis.

## Parameters

Name	Default	Description
Lrsigma		rotor stray inductance per phase translated to stator [H]
Rr		warm rotor resistance per phase translated to stator [Ohm]

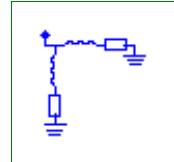
## Connectors

Name	Description
spacePhasor_r	

---

## Modelica.Electrical.Machines.BasicMachines.Components.DamperCage

Squirrel Cage



## Information

Model of an unsymmetrical damper cage in two axis.

## Parameters

Name	Default	Description
Lrsigma		stray inductance in d-axis per phase translated to stator [H]
Lrsigmaq		stray inductance in q-axis per phase translated to stator [H]
Rr		warm resistance in d-axis per phase translated to stator [Ohm]
Rrq		warm resistance in q-axis per phase translated to stator [Ohm]

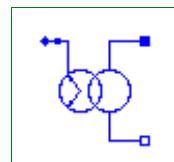
## Connectors

Name	Description
spacePhasor_r	

---

## Modelica.Electrical.Machines.BasicMachines.Components.ElectricalExcitation

Electrical excitation



## Information

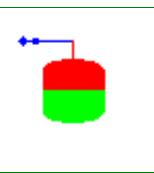
Model of an electrical excitation, converting excitation to space phasor.

## Parameters

Name	Default	Description
TurnsRatio	1	stator current / excitation current

## Connectors

Name	Description
spacePhasor_r	
pin_ep	
pin_en	

**Modelica.Electrical.Machines.BasicMachines.Components.PermanentMagnet**
**Permanent magnet excitation**
**Information**

Model of a permanent magnet excitation, characterized by an equivalent excitation current.

**Parameters**

Name	Default	Description
le		Equivalent excitation current [A]

**Connectors**

Name	Description
spacePhasor_r	

**Modelica.Electrical.Machines.BasicMachines.Components.BasicDCMachine**
**Partial model for DC machine**
**Information**

Partial model for DC machine models, containing:

- main parts of the icon
- armature pins
- mechanical connectors

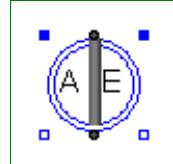
**Parameters**

Name	Default	Description
J_Rotor	0.15	rotor's moment of inertia [kg.m <sup>2</sup> ]
J_Stator	J_Rotor	stator's moment of inertia [kg.m <sup>2</sup> ]
TurnsRatio		Ratio of armature turns over number of turns of the excitation winding
airGapDC	redeclare AirGapDC airGapDC(...)	
Nominal parameters		
VaNominal	100	nominal armature voltage [V]
IaNominal	100	nominal armature current [A]
rpmNominal	1425	nominal speed [rev/min]
Nominal resistances and inductances		
Ra	0.05	warm armature resistance [Ohm]
La	0.0015	armature inductance [H]

**Connectors**

Name	Description
flange_a	

support	support at which the reaction torque is acting
pin_ap	
pin_an	

**Modelica.Electrical.Machines.BasicMachines.Components.PartialAirGapDC****Partial airgap model of a DC machine****Information**

Linear model of the airgap (without saturation effects) of a DC machine, using only equations.  
 Induced excitation voltage is calculated from  $\text{der}(\text{flux})$ , where flux is defined by excitation inductance times excitation current.  
 Induced armature voltage is calculated from flux times angular velocity.

**Parameters**

Name	Default	Description
TurnsRatio		Ratio of armature turns over number of turns of the excitation winding

**Connectors**

Name	Description
flange_a	
support	support at which the reaction torque is acting
pin_ap	
pin_ep	
pin_an	
pin_en	

**Modelica.Electrical.Machines.BasicMachines.Components.AirGapDC****Linear airgap model of a DC machine****Information**

Linear model of the airgap (without saturation effects) of a DC machine, using only equations.  
 Induced excitation voltage is calculated from  $\text{der}(\text{flux})$ , where flux is defined by excitation inductance times excitation current.  
 Induced armature voltage is calculated from flux times angular velocity.

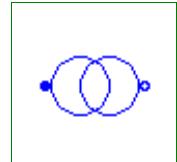
**Parameters**

Name	Default	Description
TurnsRatio		Ratio of armature turns over number of turns of the excitation winding
Le		Excitation inductance [H]

**Connectors**

Name	Description
flange_a	

support	support at which the reaction torque is acting
pin_ap	
pin_ep	
pin_an	
pin_en	

**Modelica.Electrical.Machines.BasicMachines.Components.BasicTransformer****Partial model of threephase transformer****Information**

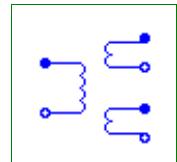
Partialmodel of a threephase transformer, containing primary and secondary resistances and stray inductances, as well as the iron core. Circuit layout (vector group) of primary and secondary windings have to be defined.

**Parameters**

Name	Default	Description
n	1	Primary voltage (line-to-line) / Secondary voltage (line-to-line)
R1	5E-3/(if C1 == "D" then 1 else 0)	Warm primary resistance per phase [Ohm]
L1sigma	78E-6/(if C1 == "D" then 1 else 0)	Primary stray inductance per phase [H]
R2	5E-3/(if C2 == "d" then 1 else 0)	Warm secondary resistance per phase [Ohm]
L2sigma	78E-6/(if C2 == "d" then 1 else 0)	Secondary stray inductance per phase [H]
core	redeclare IdealCore core(fin...	

**Connectors**

Name	Description
plug1	
plug2	

**Modelica.Electrical.Machines.BasicMachines.Components.PartialCore****Partial model of transformer core with 3 windings****Information**

Partial model of transformer core with 3 windings; saturation function flux versus magentizing current has to be defined.

**Parameters**

Name	Default	Description
m	3	Number of phases
n12	1	Turns ratio 1:2
n13	2	Turns ratio 1:3

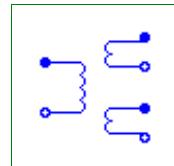
## Connectors

Name	Description
plug_p1	
plug_n1	
plug_p2	
plug_n2	
plug_p3	
plug_n3	

---

## Modelica.Electrical.Machines.BasicMachines.Components.IdealCore

Ideal transformer with 3 windings



## Information

Ideal transformer with 3 windings: no magnetizing current.

## Parameters

Name	Default	Description
m	3	Number of phases
n12	1	Turns ratio 1:2
n13	2	Turns ratio 1:3

## Connectors

Name	Description
plug_p1	
plug_n1	
plug_p2	
plug_n2	
plug_p3	
plug_n3	

---

## Modelica.Electrical.Machines.Sensors

Sensors for machine modelling

## Information

This package contains sensors that are usefull when modelling machines.

## Package Content

Name	Description
VoltageRMSsensor	Length of spcae phasor -> RMS voltage
CurrentRMSsensor	Length of spcae phasor -> RMS current
ElectricalPowerSensor	Instantaneous power from spcae phasors

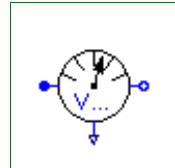
---

 MechanicalPowerSensor	Mechanical power = torque x speed
 RotorAngle	Rotor lagging angle

---

### Modelica.Electrical.Machines.Sensors.VoltageRMSSensor

Length of space phasor -> RMS voltage



#### Information

Measured 3-phase instantaneous voltages are transformed to the corresponding space phasor; output is length of the space phasor divided by sqrt(2), thus giving in sinusoidal stationary state RMS voltage.

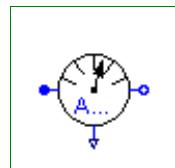
#### Connectors

Name	Description
V	
plug_p	
plug_n	

---

### Modelica.Electrical.Machines.Sensors.CurrentRMSSensor

Length of space phasor -> RMS current



#### Information

Measured 3-phase instantaneous currents are transformed to the corresponding space phasor; output is length of the space phasor divided by sqrt(2), thus giving in sinusoidal stationary state RMS current.

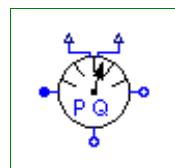
#### Connectors

Name	Description
I	
plug_p	
plug_n	

---

### Modelica.Electrical.Machines.Sensors.ElectricalPowerSensor

Instantaneous power from space phasors



#### Information

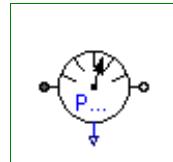
3-phase instantaneous voltages (plug\_p - plug\_nv) and currents (plug\_p - plug\_ni) are transformed to the corresponding space phasors, which are used to calculate power quantities:  
 P = instantaneous power, thus giving in stationary state active power.  
 Q = giving in stationary state reactive power.

## Connectors

Name	Description
P	
Q	
plug_p	
plug_ni	
plug_nv	

## Modelica.Electrical.Machines.Sensors.MechanicalPowerSensor

Mechanical power = torque x speed



## Information

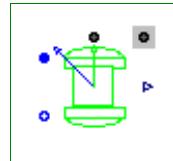
Calculates (mechanical) power from torque times angular speed.

## Connectors

Name	Description
flange_a	
flange_b	
P	

## Modelica.Electrical.Machines.Sensors.RotorAngle

Rotor lagging angle



## Information

Calculates rotor lagging angle by measuring the stator phase voltages, transforming them to the correspondig space phasor in stator-fixed coordinate system, rotating the space phasor to the rotor-fixed coordinate system and calculating the angle of this space phasor.

The sensor's housing is implicitly fixed.

Since the machine's stator also implicitly fixed, the angle at the flange is equal to the angle of the machine's rotor against the stator.

## Parameters

Name	Default	Description
p		number of pole pairs

## Connectors

Name	Description
rotorAngle	
plug_p	
plug_n	
flange	
support	support at which the reaction torque is acting

---

## Modelica.Electrical.Machines.SpacePhasors

Library with space phasor-models

### Information

This package contains components, blocks and functions to utilize space phasor theory.

### Package Content

Name	Description
Components	Basic space phasor models
Blocks	Blocks for space phasor transformation
Functions	Functions for space phasor transformation

---

## Modelica.Electrical.Machines.SpacePhasors.Components

Basic space phasor models

### Information

This package contains basic space phasor models.

Real and imaginary part of voltage space phasor are the potentials  $v_{[2]}$  of the space phasor connector; (implicit grounded).

Real and imaginary part of current space phasor are the currents  $i_{[2]}$  at the space phasor connector; a ground has to be used where necessary for currents flowing back.

### Package Content

Name	Description
SpacePhasor	Physical transformation: three phase <-> space phasors
Rotator	Rotates space phasor

---

## Modelica.Electrical.Machines.SpacePhasors.Components.SpacePhasor

Physical transformation: three phase <-> space phasors

### Information

Physical transformation of voltages and currents: three phases <-> space phasors:

$$x[k] = X_0 + \{\cos(-(k-1)/m^2\pi), -\sin(-(k-1)/m^2\pi)\} * X[Re,Im]$$

and vice versa:

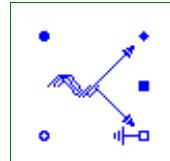
$$X_0 = \text{sum}(x[k])/m$$

$$X[Re,Im] = \text{sum}(2/m^2 \{\cos((k-1)/m^2\pi), \sin((k-1)/m^2\pi)\}) * x[k]$$

where  $x$  designates three phase values,  $X[Re,Im]$  designates the space phasor and  $X_0$  designates the zero sequence system.

*Physical transformation* means that both voltages and currents are transformed in both directions.

Zero-sequence voltage and current are present at pin zero. An additional zero-sequence impedance could be connected between pin zero and pin ground.



## Connectors

Name	Description
plug_p	
plug_n	
zero	
ground	
spacePhasor	

## Modelica.Electrical.Machines.SpacePhasors.Components.Rotator

Rotates space phasor



## Information

Rotates space phasors of left connector to right connector by the angle provided by the input signal "angle" from one coordinate system into another.

## Connectors

Name	Description
spacePhasor_a	
spacePhasor_b	
angle	

## Modelica.Electrical.Machines.SpacePhasors.Blocks

Blocks for space phasor transformation

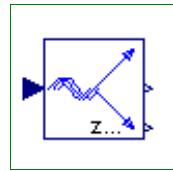
## Information

This package contains space phasor transformation blocks for use in controllers:

- ToSpacePhasor: transforms a set of threephase values to space phasor and zero sequence system
- FromSpacePhasor: transforms a space phasor and zero sequence system to a set of threephase values
- Rotator: rotates a space phasor (from one coordinate system into another)
- ToPolar: Converts a space phasor from rectangular coordinates to polar coordinates
- FromPolar: Converts a space phasor from polar coordinates to rectangular coordinates

## Package Content

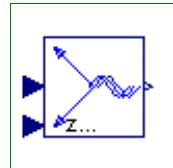
Name	Description
ToSpacePhasor	Conversion: three phase -> space phasor
FromSpacePhasor	Conversion: space phasor -> three phase
Rotator	Rotates space phasor
ToPolar	Converts a space phasor to polar coordinates
FromPolar	Converts a space phasor from polar coordinates

**Modelica.Electrical.Machines.SpacePhasors.Blocks.ToSpacePhasor****Conversion:** three phase -> space phasor**Information**

Transformation of threephase values (voltages or currents) to space phasor and zero sequence value.

**Connectors**

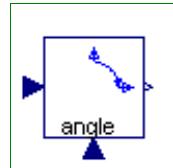
Name	Description
u[nin]	Connector of Real input signals
y[nout]	Connector of Real output signals
zero	

**Modelica.Electrical.Machines.SpacePhasors.Blocks.FromSpacePhasor****Conversion:** space phasor -> three phase**Information**

Transformation of space phasor and zero sequence value to threephase values (voltages or currents).

**Connectors**

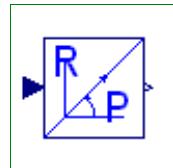
Name	Description
u[nin]	Connector of Real input signals
y[nout]	Connector of Real output signals
zero	

**Modelica.Electrical.Machines.SpacePhasors.Blocks.Rotator****Rotates space phasor****Information**

Rotates a space phasor (voltage or current) by the angle provided by the input signal "angle" from one coordinate system into another.

**Connectors**

Name	Description
u[n]	Connector of Real input signals
y[n]	Connector of Real output signals
angle	

**Modelica.Electrical.Machines.SpacePhasors.Blocks.ToPolar****Converts a space phasor to polar coordinates**

## Information

Converts a space phasor from rectangular coordinates to polar coordinates.

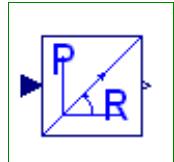
## Connectors

Name	Description
u[n]	Connector of Real input signals
y[n]	Connector of Real output signals

---

## Modelica.Electrical.Machines.SpacePhasors.Blocks.FromPolar

Converts a space phasor from polar coordinates



## Information

Converts a space phasor from polar coordinates to rectangular coordinates.

## Connectors

Name	Description
u[n]	Connector of Real input signals
y[n]	Connector of Real output signals

---

## Modelica.Electrical.Machines.SpacePhasors.Functions

Functions for space phasor transformation

## Information

This package contains space phasor transformation functions for use in calculations:

- ToSpacePhasor: transforms a set of threephase values to space phasor and zero sequence system
- FromSpacePhasor: transforms a space phasor and zero sequence system to a set of threephase values
- Rotator: rotates a space phasor (from one coordinate system into another)
- ToPolar: Converts a space phasor from rectangular coordinates to polar coordinates
- FromPolar: Converts a space phasor from polar coordinates to rectangular coordinates

## Package Content

Name	Description
(f) ToSpacePhasor	Conversion: three phase -> space phasor
(f) FromSpacePhasor	Conversion: space phasor -> three phase
(f) Rotator	Rotates space phasor
(f) ToPolar	Converts a space phasor to polar coordinates
(f) FromPolar	Converts a space phasor from polar coordinates

**Modelica.Electrical.Machines.SpacePhasors.Functions.ToSpacePhasor**

**Conversion:** three phase -> space phasor

**Information**

Transformation of three phase values (voltages or currents) to space phasor and zero sequence value:  
 $y[k] = X0 + \{\cos(-(k - 1)/m^2\pi), -\sin(-(k - 1)/m^2\pi)\} * X[Re,Im]$   
 where  $y$  designates three phase values,  $X[Re,Im]$  designates the space phasor and  $X0$  designates the zero sequence system.

**Inputs**

Name	Default	Description
x[3]		

**Outputs**

Name	Description
y[2]	
y0	

**Modelica.Electrical.Machines.SpacePhasors.Functions.FromSpacePhasor**

**Conversion:** space phasor -> three phase

**Information**

Transformation of space phasor and zero sequence value to three phase values (voltages or currents):  
 $Y0 = \text{sum}(x[k])/m$   
 $Y[Re,Im] = \text{sum}(2/m^2\{\cos((k - 1)/m^2\pi), \sin((k - 1)/m^2\pi)\}) * x[k]$   
 where  $x$  designates three phase values,  $Y[Re,Im]$  designates the space phasor and  $Y0$  designates the zero sequence system.

**Inputs**

Name	Default	Description
x[2]		
x0		

**Outputs**

Name	Description
y[3]	

**Modelica.Electrical.Machines.SpacePhasors.Functions.Rotator**

**Rotates space phasor**

**Information**

Rotates a space phasor (voltage or current) by the angle provided by input argument "angle" from one coordinate system into another:

## 348 Modelica.Electrical.Machines.SpacePhasors.Functions.Rotator

---

$y[Re,Im] := \{\{+\cos(-\text{angle}), -\sin(-\text{angle})\}, \{+\sin(-\text{angle}), +\cos(-\text{angle})\}\} * x[Re,Im]$   
where  $y[Re,Im]$  designates the space phasor in the new coordinate system (twisted by angle against old coordinate system) and  $y[Re,Im]$  designates the space phasor in the old coordinate system.

### Inputs

Name	Default	Description
x[2]		
angle		[rad]

### Outputs

Name	Description
y[2]	

---

## Modelica.Electrical.Machines.SpacePhasors.Functions.ToPolar

Converts a space phasor to polar coordinates



### Information

Converts a space phasor from rectangular coordinates to polar coordinates, providing angle=0 for {0,0}.

### Inputs

Name	Default	Description
x[2]		

### Outputs

Name	Description
absolute	
angle	[rad]

---

## Modelica.Electrical.Machines.SpacePhasors.Functions.FromPolar

Converts a space phasor from polar coordinates



### Information

Converts a space phasor from polar coordinates to rectangular coordinates.

### Inputs

Name	Default	Description
absolute		
angle		[rad]

### Outputs

Name	Description
x[2]	

---

## Modelica.Electrical.Machines.Interfaces

### SpacePhasor connector and PartialMachines

#### Information

This package contains the space phasor connector and partial models for machine models.

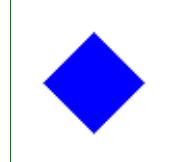
#### Package Content

Name	Description
 SpacePhasor	Connector for Space Phasors
 Adapter	From Modelica.Mechanics.Rotational.Interfaces.TwoFlangesAndBearingH
 PartialBasicMachine	Partial model for all machines
 PartialBasicInductionMachine	Partial model for induction machine
 PartialBasicDCMachine	Partial model for DC machine

---

## Modelica.Electrical.Machines.Interfaces.SpacePhasor

### Connector for Space Phasors



#### Information

Connector for Space Phasors:

- Voltage  $v_{[2]}$  ... Real and Imaginary part of voltage space phasor
- Current  $i_{[2]}$  ... Real and Imaginary part of current space phasor

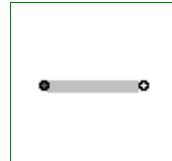
#### Contents

Name	Description
$v_{[2]}$	[V]
$i_{[2]}$	[A]

---

## Modelica.Electrical.Machines.Interfaces.Adapter

### From Modelica.Mechanics.Rotational.Interfaces.TwoFlangesAndBearingH



#### Information

From Modelica.Mechanics.Rotational.Interfaces.TwoFlangesAndBearingH:

If bearingConnected = true torque is given from flange\_a to flange\_b

If bearingConnected = false flange\_a is fixed

#### Parameters

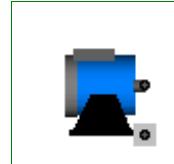
Name	Default	Description
bearingConnected		Choose whether bearing is connected

## Connectors

Name	Description
flange_a	
flange_b	

## Modelica.Electrical.Machines.Interfaces.PartialBasicMachine

Partial model for all machines



### Information

Base partial model DC machines:

- main parts of the icon
- mechanical flange
- mechanical support

Besides the mechanical connector *flange\_a* (i.e. the shaft) the machines have a second mechanical connector *support*.

If nothing is connected to *support*, it is assumed that the stator is fixed.

Otherwise reaction torque (i.e. airGap torque, minus acceleration torque for stator's moment of inertia) can be measured at *support*.

One may also fix the the shaft and let rotate the stator; parameter *Js* is only of importance when the stator is rotating.

**Take care:** Even if You connect only a sensor (e.g. Modelica.Mechanics.Rotational.Sensors.RelAngleSensor or Machines.Sensors.RotorDisplacementAngle) to the *support*, You have to fix the *support*!

## Parameters

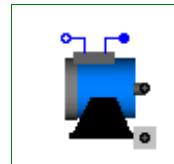
Name	Default	Description
J_Rotor		rotor's moment of inertia [kg.m <sup>2</sup> ]
J_Stator	J_Rotor	stator's moment of inertia [kg.m <sup>2</sup> ]

## Connectors

Name	Description
flange_a	
support	support at which the reaction torque is acting

## Modelica.Electrical.Machines.Interfaces.PartialBasicInductionMachine

Partial model for induction machine



### Information

Partial model for induction machine models, containing:

- main parts of the icon
- stator plugs
- mechanical connectors

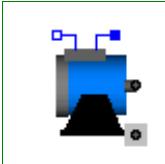
## Parameters

Name	Default	Description
J_Rotor	0.29	rotor's moment of inertia [kg.m2]
J_Stator	J_Rotor	stator's moment of inertia [kg.m2]
p	2	number of pole pairs (Integer)

## Connectors

Name	Description
flange_a	
support	support at which the reaction torque is acting
plug_sp	
plug_sn	

## Modelica.Electrical.Machines.Interfaces.PartialBasicDCMachine



Partial model for DC machine

## Information

Partial model for DC machine models, containing:

- main parts of the icon
- armature pins
- mechanical connectors

## Parameters

Name	Default	Description
J_Rotor	0.15	rotor's moment of inertia [kg.m2]
J_Stator	J_Rotor	stator's moment of inertia [kg.m2]

## Connectors

Name	Description
flange_a	
support	support at which the reaction torque is acting
pin_ap	
pin_an	

## Modelica.Electrical.MultiPhase

Library for electrical components with 2, 3 or more phases

## Information

This package contains packages for electrical multiphase components, based on Modelica.Electrical.Analog:

- Basic: basic components (resistor, capacitor, inductor, ...)
- Ideal: ideal elements (switches, diode, transformer, ...)
- Sensors: sensors to measure potentials, voltages, and currents
- Sources: time-dependend and controlled voltage and current sources

## 352 Modelica.Electrical.MultiPhase

---

This package is intended to be used the same way as Modelica.Electrical.Analog but to make design of multiphase models easier.

The package is based on the plug: a composite connector containing m pins.

It is possible to connect plugs to plugs or single pins of a plug to single pins.

Potentials may be accessed as `plug.pin[] .v`, currents may be accessed as `plug.pin[] .i`.

Further development:

- temperature-dependent resistor
- lines (m-phase models)

### Main Author:

[Anton Haumer](#)

Technical Consulting & Electrical Engineering

A-3423 St.Andrae-Woerdern

Austria

email: [a.haumer@haumer.at](mailto:a.haumer@haumer.at)

Copyright © 1998-2007, Modelica Association and Anton Haumer.

*This Modelica package is free software; it can be redistributed and/or modified under the terms of the Modelica license, see the license conditions and the accompanying disclaimer [here](#).*

---

## Package Content

Name	Description
 Basic	Basic components for electrical multiphase models
 Examples	Multiphase test examples
 Ideal	Multiphase components with idealized behaviour
 Interfaces	Interfaces for electrical multiphase models
 Sensors	Multiphase potential, voltage and current Sensors
 Sources	Multiphase voltage and current sources

---

## Modelica.Electrical.MultiPhase.Basic

### Basic components for electrical multiphase models

#### Information

This package contains basic analog electrical multiphase components.

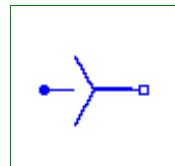
#### Package Content

Name	Description
 Star	Star-connection
 Delta	Delta (polygon) connection
 PlugToPin_p	Connect one (positive) Pin
 PlugToPin_n	Connect one (negative) Pin
 Resistor	Ideal linear electrical resistors

 Conductor	Ideal linear electrical conductors
 Capacitor	Ideal linear electrical capacitors
 Inductor	Ideal linear electrical inductors
 SaturatingInductor	Simple model of inductors with saturation
 Transformer	Multiphase Transformer
 VariableResistor	Ideal linear electrical resistors with variable resistance
 VariableConductor	Ideal linear electrical conductors with variable conductance
 VariableCapacitor	Ideal linear electrical capacitors with variable capacitance
 VariableInductor	Ideal linear electrical inductors with variable inductance

## Modelica.Electrical.MultiPhase.Basic.Star

Star-connection



### Information

Connects all pins of plug\_p to pin\_n, thus establishing a so-called star-connection.

### Parameters

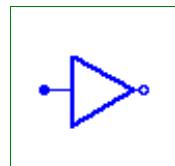
Name	Default	Description
m	3	number of phases

### Connectors

Name	Description
plug_p	
pin_n	

## Modelica.Electrical.MultiPhase.Basic.Delta

Delta (polygon) connection



### Information

Connects in a cyclic way plug\_n.pin[j] to plug\_p.pin[j+1], thus establishing a so-called delta (or polygon) connection when used in parallel to another component.

### Parameters

Name	Default	Description
m	3	number of phases

### Connectors

Name	Description
plug_p	
plug_n	

---

**Modelica.Electrical.MultiPhase.Basic.PlugToPin\_p****Connect one (positive) Pin****Information**

Connects pin  $k$  of plug\_p to pin\_p, leaving the other pins of plug\_p unconnected.

**Parameters**

Name	Default	Description
m	3	number of phases
k	1	phase index

**Connectors**

Name	Description
plug_p	
pin_p	

---

**Modelica.Electrical.MultiPhase.Basic.PlugToPin\_n****Connect one (negative) Pin****Information**

Connects pin  $k$  of plug\_n to pin\_n, leaving the other pins of plug\_n unconnected.

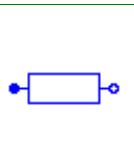
**Parameters**

Name	Default	Description
m	3	number of phases
k	1	phase index

**Connectors**

Name	Description
plug_n	
pin_n	

---

**Modelica.Electrical.MultiPhase.Basic.Resistor****Ideal linear electrical resistors****Information**

Contains  $m$  resistors (Modelica.Electrical.Analog.Basic.Resistor)

## Parameters

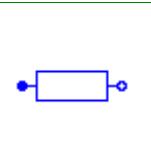
Name	Default	Description
m	3	number of phases
R[m]	fill(1, m)	Resistance [Ohm]

## Connectors

Name	Description
plug_p	
plug_n	

## Modelica.Electrical.MultiPhase.Basic.Conductor

Ideal linear electrical conductors



## Information

Contains m conductors (Modelica.Electrical.Analog.Basic.Conductor)

## Parameters

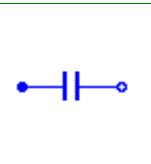
Name	Default	Description
m	3	number of phases
G[m]	fill(1, m)	Conductance [S]

## Connectors

Name	Description
plug_p	
plug_n	

## Modelica.Electrical.MultiPhase.Basic.Capacitor

Ideal linear electrical capacitors



## Information

Contains m capacitors (Modelica.Electrical.Analog.Basic.Capacitor)

## Parameters

Name	Default	Description
m	3	number of phases
C[m]	fill(1, m)	Capacitance [F]

## Connectors

Name	Description
plug_p	
plug_n	

**Modelica.Electrical.MultiPhase.Basic.Inductor**

Ideal linear electrical inductors

**Information**

Contains m inductors (Modelica.Electrical.Analog.Basic.Inductor)

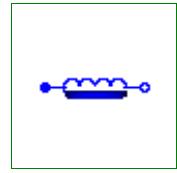
**Parameters**

Name	Default	Description
m	3	number of phases
L[m]	fill(1, m)	Inductance [H]

**Connectors**

Name	Description
plug_p	
plug_n	

---

**Modelica.Electrical.MultiPhase.Basic.SaturatingInductor**

Simple model of inductors with saturation

**Information**

Contains m saturating inductors (Modelica.Electrical.Analog.Basic.SaturatingInductor)

**Attention!!!**

Each element of the array of saturatingInductors is only dependent on the current flowing through this element.

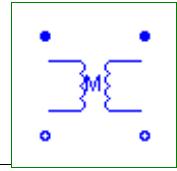
**Parameters**

Name	Default	Description
m	3	number of phases
Inom[m]	fill(1, m)	Nominal current [A]
Lnom[m]	fill(1, m)	Nominal inductance at Nominal current [H]
Lzer[m]	{2*Lnom[j] for j in 1:m}	Inductance near current=0 [H]
Linf[m]	{Lnom[j]/2 for j in 1:m}	Inductance at large currents [H]

**Connectors**

Name	Description
plug_p	
plug_n	

---

**Modelica.Electrical.MultiPhase.Basic.Transformer**

Multiphase Transformer

## Information

Contains m transformers (Modelica.Electrical.Analog.Basic.Transformer)

## Parameters

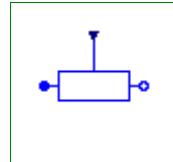
Name	Default	Description
m	3	number of phases
L1[m]	fill(1, m)	Primary inductance [H]
L2[m]	fill(1, m)	Secondary inductance [H]
M[m]	fill(1, m)	Coupling inductance [H]

## Connectors

Name	Description
plug_p1	
plug_p2	
plug_n1	
plug_n2	

## Modelica.Electrical.MultiPhase.Basic.VariableResistor

Ideal linear electrical resistors with variable resistance



## Information

Contains m variable resistors (Modelica.Electrical.Analog.Basic.VariableResistor)

## Attention!!!

It is recommended that none of the R\_Port signals should not cross the zero value. Otherwise depending on the surrounding circuit the probability of singularities is high.

## Parameters

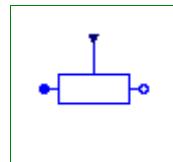
Name	Default	Description
m	3	number of phases

## Connectors

Name	Description
plug_p	
plug_n	
R[m]	

## Modelica.Electrical.MultiPhase.Basic.VariableConductor

Ideal linear electrical conductors with variable conductance



## Information

Contains m variable conductors (Modelica.Electrical.Analog.Basic.VariableConductor)

## Attention!!!

## 358 Modelica.Electrical.MultiPhase.Basic.VariableConductor

---

It is recommended that none of the G\_Port signals should not cross the zero value. Otherwise depending on the surrounding circuit the probability of singularities is high.

### Parameters

Name	Default	Description
m	3	number of phases

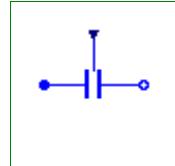
### Connectors

Name	Description
plug_p	
plug_n	
G[m]	

---

## Modelica.Electrical.MultiPhase.Basic.VariableCapacitor

Ideal linear electrical capacitors with variable capacitance



### Information

Contains m variable capacitors (Modelica.Electrical.Analog.Basic.VariableCapacitor)

It is required that each C\_Port.signal  $\geq 0$ , otherwise an assertion is raised. To avoid a variable index system, C = Cmin, if  $0 \leq C\_Port.signal < Cmin$ , where Cmin is a parameter with default value Modelica.Constants.eps.

### Parameters

Name	Default	Description
m	3	number of phases
Cmin[m]	fill(Modelica.Constants.eps,...)	minimum Capacitance [F]

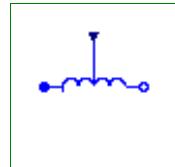
### Connectors

Name	Description
plug_p	
plug_n	
C[m]	

---

## Modelica.Electrical.MultiPhase.Basic.VariableInductor

Ideal linear electrical inductors with variable inductance



### Information

Contains m variable inductors (Modelica.Electrical.Analog.Basic.VariableInductor)

It is required that each L\_Port.signal  $\geq 0$ , otherwise an assertion is raised. To avoid a variable index system, L = Lmin, if  $0 \leq L\_Port.signal < Lmin$ , where Lmin is a parameter with default value Modelica.Constants.eps.

## Parameters

Name	Default	Description
m	3	number of phases
Lmin[m]	fill(Modelica.Constants.eps,...)	minimum Inductance [H]

## Connectors

Name	Description
plug_p	
plug_n	
L[m]	

---

## Modelica.Electrical.MultiPhase.Examples

### Multiphase test examples

## Information

This package contains test examples of analog electrical multiphase circuits.

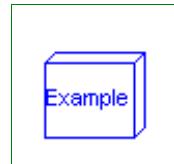
## Package Content

Name	Description
 TransformerYY	Test example with multiphase components
 TransformerYD	Test example with multiphase components
 Rectifier	Test example with multiphase components

---

## Modelica.Electrical.MultiPhase.Examples.TransformerYY

### Test example with multiphase components



## Information

Test example with multiphase components:

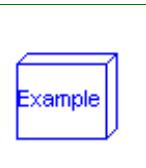
Star-connected voltage source feeds via a Y-Y-transformer with internal impedance (RT, LT) a load resistor RT.

Using f=5 Hz LT=3mH defines nominal voltage drop of approximately 10 %.

Simulate for 1 second (2 periods) and compare voltages and currents of source, transformer and load.

## Parameters

Name	Default	Description
m	3	Number of phases
V	1	Amplitude of Star-Voltage [V]
f	5	Frequency [Hz]
LT	0.003	Transformer stray inductance [H]
RT	0.05	Transformer resistance [Ohm]
RL	1	Load Resistance [Ohm]

**Modelica.Electrical.MultiPhase.Examples.TransformerYD****Test example with multiphase components****Information**

Test example with multiphase components:

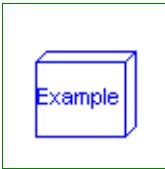
Star-connected voltage source feeds via a Y-D-transformer with internal impedance (RT, LT) a load resistor RT.

Using  $f=5\text{ Hz}$   $LT=3\text{mH}$  defines nominal voltage drop of approximately 10 %.

Simulate for 1 second (2 periods) and compare voltages and currents of source, transformer and load.

**Parameters**

Name	Default	Description
m	3	Number of phases
V	1	Amplitude of Star-Voltage [V]
f	5	Frequency [Hz]
LT	0.003	Transformer stray inductance [H]
RT	0.05	Transformer resistance [Ohm]
RL	1	Load Resistance [Ohm]
nT	$1/\sqrt{(1 - \text{Modelica.Math.co...})}$	Transformer ratio

**Modelica.Electrical.MultiPhase.Examples.Rectifier****Test example with multiphase components****Information**

Test example with multiphase components:

Star-connected voltage source feeds via a line reactor a diode bridge rectifier with a DC burden.

Using  $f=5\text{ Hz}$ , simulate for 1 second (2 periods) and compare voltages and currents of source and DC burden, neglecting initial transient.

**Parameters**

Name	Default	Description
m	3	Number of phases
V	1	Amplitude of Star-Voltage [V]
f	5	Frequency [Hz]
L	0.001	Line Inductance [H]
RL	2	Load Resistance [Ohm]
C	0.05	Total DC-Capacitance [F]
RE	1E6	Earthing Resistance [Ohm]

**Modelica.Electrical.MultiPhase.Ideal****Multiphase components with idealized behaviour**

## Information

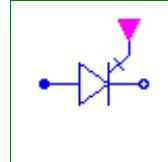
This package contains analog electrical multiphase components with idealized behaviour, like thyristor, diode, switch, transformer.

## Package Content

Name	Description
 IdealThyristor	Multiphase ideal thyristor
 IdealGTOThyristor	Multiphase ideal GTO thyristor
 IdealCommutingSwitch	Multiphase ideal commuting switch
 IdealIntermediateSwitch	Multiphase ideal intermediate switch
 IdealDiode	Multiphase ideal diode
 IdealTransformer	Multiphase ideal transformer
 Idle	Multiphase idle branch
 Short	Multiphase short cut branch
 IdealOpeningSwitch	Multiphase ideal opener
 IdealClosingSwitch	Multiphase ideal closer

## Modelica.Electrical.MultiPhase.Ideal.IdealThyristor

### Multiphase ideal thyristor



## Information

Contains m ideal thyristors (Modelica.Electrical.Analog.Ideal.IdealThyristor).

## Parameters

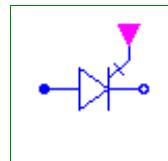
Name	Default	Description
m	3	number of phases
Ron[m]	fill(1.E-5, m)	Closed thyristor resistance [Ohm]
Goff[m]	fill(1.E-5, m)	Opened thyristor conductance [S]
Vknee[m]	zeros(m)	Threshold voltage [V]

## Connectors

Name	Description
plug_p	
plug_n	
fire[m]	

## Modelica.Electrical.MultiPhase.Ideal.IdealGTOThyristor

### Multiphase ideal GTO thyristor



## 362 Modelica.Electrical.MultiPhase.Ideal.IdealGTOThyristor

---

### Information

Contains m ideal GTO thyristors (Modelica.Electrical.Analog.Ideal.IdealGTOThyristor).

### Parameters

Name	Default	Description
m	3	number of phases
Ron[m]	fill(1.E-5, m)	Closed thyristor resistance [Ohm]
Goff[m]	fill(1.E-5, m)	Opened thyristor conductance [S]
Vknee[m]	zeros(m)	Treshold voltage [V]

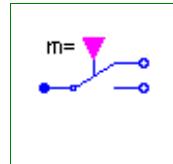
### Connectors

Name	Description
plug_p	
plug_n	
fire[m]	

---

## Modelica.Electrical.MultiPhase.Ideal.IdealCommutingSwitch

Multiphase ideal commuting switch



### Information

Contains m ideal commuting switches (Modelica.Electrical.Analog.Ideal.IdealCommutingSwitch).

### Parameters

Name	Default	Description
m	3	number of phases
Ron[m]	fill(1.E-5, m)	Closed switch resistance [Ohm]
Goff[m]	fill(1.E-5, m)	Opened switch conductance [S]

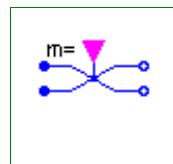
### Connectors

Name	Description
control[m]	true => p--n2 connected, false => p--n1 connected
plug_p	
plug_n2	
plug_n1	

---

## Modelica.Electrical.MultiPhase.Ideal.IdealIntermediateSwitch

Multiphase ideal intermediate switch



### Information

Contains m ideal intermediate switches (Modelica.Electrical.Analog.Ideal.IdealIntermediateSwitch).

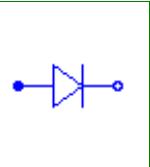
## Parameters

Name	Default	Description
m	3	number of phases
Ron[m]	fill(1.E-5, m)	Closed switch resistance [Ohm]
Goff[m]	fill(1.E-5, m)	Opened switch conductance [S]

## Connectors

Name	Description
control[m]	true => p1--n2, p2--n1 connected, otherwise p1--n1, p2--n2 connected
plug_p1	
plug_p2	
plug_n2	
plug_n1	

## Modelica.Electrical.MultiPhase.Ideal.IdealDiode



Multiphase ideal diode

## Information

Contains m ideal diodes (Modelica.Electrical.Analog.Ideal.IdealDiode).

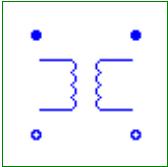
## Parameters

Name	Default	Description
m	3	number of phases
Ron[m]	fill(1.E-5, m)	Closed diode resistance [Ohm]
Goff[m]	fill(1.E-5, m)	Opened diode conductance [S]
Vknee[m]	zeros(m)	Threshold voltage [V]

## Connectors

Name	Description
plug_p	
plug_n	

## Modelica.Electrical.MultiPhase.Ideal.IdealTransformer



Multiphase ideal transformer

## Information

Contains m ideal transformers (Modelica.Electrical.Analog.Ideal.IdealTransformer).

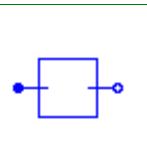
## Parameters

Name	Default	Description
m	3	number of phases

n[m]	fill(1, m)	Turns ratio
------	------------	-------------

**Connectors**

Name	Description
plug_p1	
plug_p2	
plug_n1	
plug_n2	

**Modelica.Electrical.MultiPhase.Ideal.Idle****Multiphase idle branch****Information**

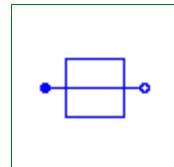
Contains m idles (Modelica.Electrical.Analog.Ideal.Idle)

**Parameters**

Name	Default	Description
m	3	number of phases

**Connectors**

Name	Description
plug_p	
plug_n	

**Modelica.Electrical.MultiPhase.Ideal.Short****Multiphase short cut branch****Information**

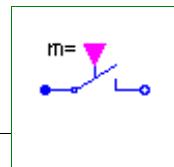
Contains m short cuts (Modelica.Electrical.Analog.Ideal.Short)

**Parameters**

Name	Default	Description
m	3	number of phases

**Connectors**

Name	Description
plug_p	
plug_n	

**Modelica.Electrical.MultiPhase.Ideal.IdealOpeningSwitch****Multiphase ideal opener**

## Information

Contains m ideal opening switches (Modelica.Electrical.Analog.Ideal.IdealOpeningSwitch).

## Parameters

Name	Default	Description
m	3	number of phases
Ron[m]	fill(1.E-5, m)	Closed switch resistance [Ohm]
Goff[m]	fill(1.E-5, m)	Opened switch conductance [S]

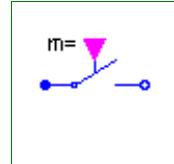
## Connectors

Name	Description
plug_p	
plug_n	
control[m]	true => switch open, false => p--n connected

---

## Modelica.Electrical.MultiPhase.Ideal.IdealClosingSwitch

Multiphase ideal closer



## Information

Contains m ideal closing switches (Modelica.Electrical.Analog.Ideal.IdealClosingSwitch).

</HTML>*Error: Found no end-tag in HTML-documentation*

## Parameters

Name	Default	Description
m	3	number of phases
Ron[m]	fill(1.E-5, m)	Closed switch resistance [Ohm]
Goff[m]	fill(1.E-5, m)	Opened switch conductance [S]

## Connectors

Name	Description
plug_p	
plug_n	
control[m]	true => p--n connected, false => switch open

---

## Modelica.Electrical.MultiPhase.Interfaces

Interfaces for electrical multiphase models

## Information

This package contains connectors and interfaces (partial models) for electrical multiphase components, based on Modelica.Electrical.Analog.

## Package Content

Name	Description
	Plug with m pins for an electric component
	Positive plug with m pins
	Negative plug with m pins
	Component with one m-phase electric port
	Component with two electrical plugs and currents from plug_p to plug_n
	Component with two m-phase electric ports
	Component with two m-phase electric ports, including currents

---

## Modelica.Electrical.MultiPhase.Interfaces.Plug

### Plug with m pins for an electric component

#### Information

Connectors PositivePlug and NegativePlug are nearly identical. The only difference is that the icons are different in order to identify more easily the plugs of a component. Usually, connector PositivePlug is used for the positive and connector NegativePlug for the negative plug of an electrical component.  
Connector Plug is a composite connector containing m Pins (Modelica.Electrical.Analog.Interfaces.Pin).

#### Parameters

Name	Default	Description
m	3	number of phases

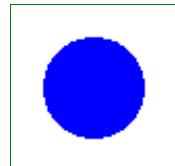
#### Contents

Name	Description
m	number of phases
pin[m]	

---

## Modelica.Electrical.MultiPhase.Interfaces.PositivePlug

### Positive plug with m pins



#### Information

Connectors PositivePlug and NegativePlug are nearly identical. The only difference is that the icons are different in order to identify more easily the plugs of a component. Usually, connector PositivePlug is used for the positive and connector NegativePlug for the negative plug of an electrical component.  
Connector Plug is a composite connector containing m Pins (Modelica.Electrical.Analog.Interfaces.Pin).

#### Parameters

Name	Default	Description
m	3	number of phases

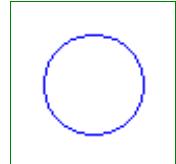
## Contents

Name	Description
m	number of phases
pin[m]	

---

## Modelica.Electrical.MultiPhase.Interfaces.NegativePlug

Negative plug with m pins



## Information

Connectors PositivePlug and NegativePlug are nearly identical. The only difference is that the icons are different in order to identify more easily the plugs of a component. Usually, connector PositivePlug is used for the positive and connector NegativePlug for the negative plug of an electrical component.  
Connector Plug is a composite connector containing m Pins (Modelica.Electrical.Analog.Interfaces.Pin).

## Parameters

Name	Default	Description
m	3	number of phases

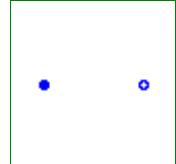
## Contents

Name	Description
m	number of phases
pin[m]	

---

## Modelica.Electrical.MultiPhase.Interfaces.TwoPlug

Component with one m-phase electric port



## Information

Superclass of elements which have **two** electrical plugs: the positive plug connector *plug\_p*, and the negative plug connector *plug\_n*. The currents flowing into *plug\_p* are provided explicitly as currents *i[m]*.

## Parameters

Name	Default	Description
m	3	number of phases

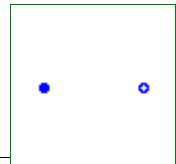
## Connectors

Name	Description
plug_p	
plug_n	

---

## Modelica.Electrical.MultiPhase.Interfaces.OnePort

Component with two electrical plugs and currents from *plug\_p* to *plug\_n*



## Information

Superclass of elements which have **two** electrical plugs: the positive plug connector *plug\_p*, and the negative plug connector *plug\_n*. The currents flowing into *plug\_p* are provided explicitly as currents *i[m]*. It is assumed that the currents flowing into *plug\_p* are identical to the currents flowing out of *plug\_n*.

## Parameters

Name	Default	Description
m	3	number of phases

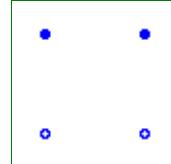
## Connectors

Name	Description
plug_p	
plug_n	

---

## Modelica.Electrical.MultiPhase.Interfaces.FourPlug

Component with two m-phase electric ports



## Information

Superclass of elements which have **four** electrical plugs.

## Parameters

Name	Default	Description
m	3	number of phases

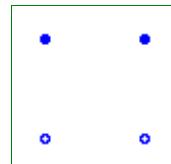
## Connectors

Name	Description
plug_p1	
plug_p2	
plug_n1	
plug_n2	

---

## Modelica.Electrical.MultiPhase.Interfaces.TwoPort

Component with two m-phase electric ports, including currents



## Information

Superclass of elements which have **four** electrical plugs. It is assumed that the currents flowing into *plug\_p1* are identical to the currents flowing out of *plug\_n1*, and that the currents flowing into *plug\_p2* are identical to the currents flowing out of *plug\_n2*.

## Parameters

Name	Default	Description
m	3	number of phases

## Connectors

Name	Description
plug_p1	
plug_p2	
plug_n1	
plug_n2	

---

## Modelica.Electrical.MultiPhase.Sensors

### Multiphase potential, voltage and current Sensors

## Information

This package contains multiphase potential, voltage, and current sensors.

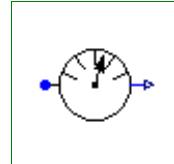
## Package Content

Name	Description
PotentialSensor	Multiphase potential sensor
VoltageSensor	Multiphase voltage sensor
CurrentSensor	Multiphase current sensor
PowerSensor	Multiphase instantaneous power sensor

---

## Modelica.Electrical.MultiPhase.Sensors.PotentialSensor

### Multiphase potential sensor



## Information

Contains m potential sensors (Modelica.Electrical.Analog.Sensors.PotentialSensor), thus measuring the m potentials  $\phi_i[m]$  of the m pins of plug\_p.

## Parameters

Name	Default	Description
m	3	number of phases

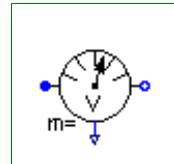
## Connectors

Name	Description
plug_p	
phi[m]	Absolute voltage potential as output signal

---

## Modelica.Electrical.MultiPhase.Sensors.VoltageSensor

### Multiphase voltage sensor



## 370 Modelica.Electrical.MultiPhase.Sensors.VoltageSensor

---

### Information

Contains m voltage sensors (Modelica.Electrical.Analog.Sensors.VoltageSensor), thus measuring the m potential differences  $v[m]$  between the m pins of plug\_p and plug\_n.

### Parameters

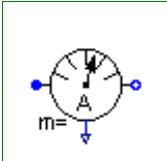
Name	Default	Description
m	3	number of phases

### Connectors

Name	Description
plug_p	
plug_n	
v[m]	Voltage between pin p and n (= p.v - n.v) as output signal

---

## Modelica.Electrical.MultiPhase.Sensors.CurrentSensor



Multiphase current sensor

### Information

Contains m current sensors (Modelica.Electrical.Analog.Sensors.CurrentSensor), thus measuring the m currents  $i[m]$  flowing from the m pins of plug\_p to the m pins of plug\_n.

### Parameters

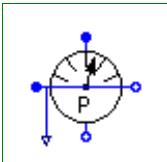
Name	Default	Description
m	3	number of phases

### Connectors

Name	Description
plug_p	
plug_n	
i[m]	current in the branch from p to n as output signal

---

## Modelica.Electrical.MultiPhase.Sensors.PowerSensor



Multiphase instantaneous power sensor

### Information

This power sensor measures instantaneous electrical power of a multiphase system and has a separated voltage and current path. The plugs of the voltage path are  $_{pv}$  and  $_{nv}$ , the plugs of the current path are  $_{pc}$  and  $_{nc}$ . The internal resistance of each current path is zero, the internal resistance of each voltage path is infinite.

### Parameters

Name	Default	Description

m	3	number of phases
---	---	------------------

## Connectors

Name	Description
pc	Positive plug, current path
nc	Negative plug, current path
pv	Positive plug, voltage path
nv	Negative plug, voltage path
power	

---

## Modelica.Electrical.MultiPhase.Sources

### Multiphase voltage and current sources

## Information

This package contains time-dependend and controlled multiphase voltage and current sources:

- SignalVoltage: fed by Modelica.Blocks.Sources arbitrary waveforms of voltages are possible
- SineVoltage : phase shift between consecutive voltages by default =  $\pi/m$
- SignalCurrent: fed by Modelica.Blocks.Sources arbitrary waveforms of currents are possible
- SineCurrent : phase shift between consecutive currents by default =  $\pi/m$

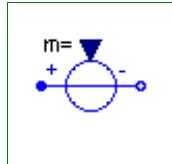
## Package Content

Name	Description
 SignalVoltage	Multiphase signal voltage source
 ConstantVoltage	Multiphase constant voltage source
 SineVoltage	Multiphase sine voltage source
 SignalCurrent	Multiphase sine current source
 ConstantCurrent	Multiphase constant current source
 SineCurrent	Multiphase sine current source

---

## Modelica.Electrical.MultiPhase.Sources.SignalVoltage

### Multiphase signal voltage source



## Information

Contains m signal controlled voltage sources (Modelica.Electrical.Analog.Sources.SignalVoltage)

## Parameters

Name	Default	Description
m	3	number of phases

## Connectors

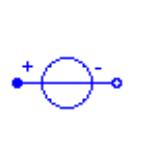
Name	Description
------	-------------

## 372 Modelica.Electrical.MultiPhase.Sources.SignalVoltage

plug_p	
plug_n	
v[m]	Voltage between pin p and n (= p.v - n.v) as input signal

## Modelica.Electrical.MultiPhase.Sources.ConstantVoltage

Multiphase constant voltage source



### Information

Contains m constant voltage sources (Modelica.Electrical.Analog.Sources.ConstantVoltage)

### Parameters

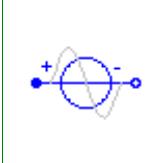
Name	Default	Description
m	3	number of phases
V[m]	fill(1, m)	Value of constant voltage [V]

### Connectors

Name	Description
plug_p	
plug_n	

## Modelica.Electrical.MultiPhase.Sources.SineVoltage

Multiphase sine voltage source



### Information

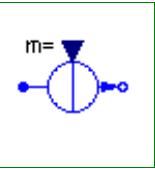
Contains m sine voltage sources (Modelica.Electrical.Analog.Sources.SineVoltage) with a default phase shift of  $-(j-1)/m * 2\pi i$  for  $j$  in 1:m.

### Parameters

Name	Default	Description
m	3	number of phases
V[m]	fill(1, m)	Amplitudes of sine waves [V]
phase[m]	$-(j-1)/m * 2\pi i$	Phases of sine waves [rad]
freqHz[m]	fill(1, m)	Frequencies of sine waves [Hz]
offset[m]	zeros(m)	Voltage offsets [V]
startTime[m]	zeros(m)	Time offsets [s]

### Connectors

Name	Description
plug_p	
plug_n	

**Modelica.Electrical.MultiPhase.Sources.SignalCurrent****Multiphase sine current source****Information**

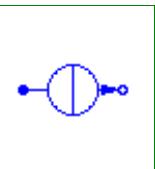
Contains m signal controlled current sources (Modelica.Electrical.Analog.Sources.SignalCurrent)

**Parameters**

Name	Default	Description
m	3	number of phases

**Connectors**

Name	Description
plug_p	
plug_n	
i[m]	Current flowing from pin p to pin n as input signal

**Modelica.Electrical.MultiPhase.Sources.ConstantCurrent****Multiphase constant current source****Information**

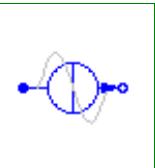
Contains m constant current sources (Modelica.Electrical.Analog.Sources.ConstantCurrent)

**Parameters**

Name	Default	Description
m	3	number of phases
I[m]	fill(1, m)	Value of constant current [A]

**Connectors**

Name	Description
plug_p	
plug_n	

**Modelica.Electrical.MultiPhase.Sources.SineCurrent****Multiphase sine current source****Information**Contains m sine current sources (Modelica.Electrical.Analog.Sources.SineCurrent) with a default phase shift of  $-(j-1)/m * 2\pi$  for  $j$  in  $1:m$ .**Parameters**

Name	Default	Description

## 374 Modelica.Electrical.MultiPhase.Sources.SineCurrent

---

m	3	number of phases
I[m]	fill(1, m)	Amplitudes of sine waves [A]
phase[m]	-{(j - 1)/m*2*Modelica.Const...	Phases of sine waves [rad]
freqHz[m]	fill(1, m)	Frequencies of sine waves [Hz]
offset[m]	zeros(m)	Current offsets [A]
startTime[m]	zeros(m)	Time offsets [s]

## Connectors

Name	Description
plug_p	
plug_n	

---

## Modelica.Icons

### Library of icons

## Information

This package contains definitions for the graphical layout of components which may be used in different libraries. The icons can be utilized by inheriting them in the desired class using "extends" or by directly copying the "icon" layer.

### Main Author:

Martin Otter

Deutsches Zentrum fuer Luft und Raumfahrt e.V. (DLR)  
Oberpfaffenhofen  
Postfach 1116  
D-82230 Wessling  
email: [Martin.Otter@dlr.de](mailto:Martin.Otter@dlr.de)

Copyright © 1998-2007, Modelica Association and DLR.

*This Modelica package is free software; it can be redistributed and/or modified under the terms of the Modelica license, see the license conditions and the accompanying disclaimer [here](#).*

## Package Content

Name	Description
 Info	Icon for an information class
 Library	Icon for library
 Library2	Icon for library where additional icon elements shall be added
 Example	Icon for an example model
 Function	Icon for a function
 Record	Icon for a record
 Enumeration	Icon for an enumeration (emulated by a package)
 TypeReal	Icon for a Real type
 TypeInteger	Icon for an Integer type

 TypeBoolean	Icon for a Boolean type
 TypeString	Icon for a String type
 TranslationalSensor	Icon representing translational measurement device
 RotationalSensor	Icon representing rotational measurement device
 GearIcon	Icon for gearbox
 MotorIcon	Icon for electrical motor
 SignalBus	Icon for signal bus
 SignalSubBus	Icon for signal sub-bus

## Types and constants

```

type TypeReal "Icon for a Real type"
  extends Real;
end TypeReal;

type TypeInteger "Icon for an Integer type"
  extends Integer;
end TypeInteger;

type TypeBoolean "Icon for a Boolean type"
  extends Boolean;
end TypeBoolean;

type TypeString "Icon for a String type"
  extends String;
end TypeString;

```

## Modelica(Icons.Info

### Icon for an information class



### Information

This icon is designed for an **information** class.

## Modelica(Icons.Library

### Icon for library

### Information

This icon is designed for a **library**.

## Modelica(Icons.Library2

### Icon for library where additional icon elements shall be added

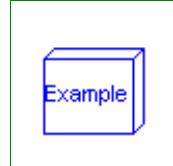
## Information

This icon is designed for a **package** where a package specific graphic is additionally included in the icon.

---

### Modelica(Icons.Example)

Icon for an example model



## Information

This icon is designed for an **Example package**, i.e. a package containing executable demo models.

---

### Modelica(Icons.Function)

Icon for a function



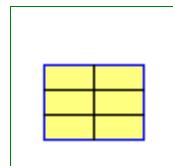
## Information

This icon is designed for a **function**

---

### Modelica(Icons.Record)

Icon for a record



## Information

This icon is designed for a **record**

---

### Modelica(Icons.Enumeration)

Icon for an enumeration (emulated by a package)



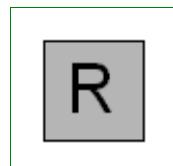
## Information

This icon is designed for an **enumeration** (that is emulated by a package).

---

### Modelica(Icons.TypeReal)

Icon for a Real type



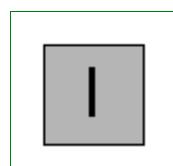
## Information

This icon is designed for a **Real type**.

---

### Modelica(Icons.TypeInteger)

Icon for an Integer type

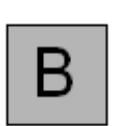


**Information**

This icon is designed for an **Integer** type.

**Modelica.Icons.TypeBoolean**

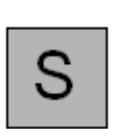
Icon for a Boolean type

**Information**

This icon is designed for a **Boolean** type.

**Modelica.Icons.TypeString**

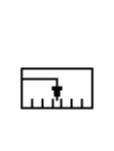
Icon for a String type

**Information**

This icon is designed for a **String** type.

**Modelica.Icons.TranslationalSensor**

Icon representing translational measurement device

**Information**

This icon is designed for a **translational sensor** model.

**Modelica.Icons.RotationalSensor**

Icon representing rotational measurement device

**Information**

This icon is designed for a **rotational sensor** model.

**Modelica.Icons.GearIcon**

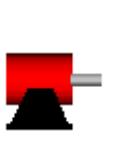
Icon for gearbox

**Information**

This icon is designed for a **gearbox** model.

**Modelica.Icons.MotorIcon**

Icon for electrical motor

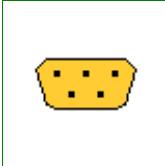


## Information

This icon is designed for an **electrical motor** model.

## Modelica(Icons.SignalBus

Icon for signal bus

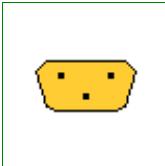


## Information

This icon is designed for a **signal bus** connector.

## Modelica(Icons.SignalSubBus

Icon for signal sub-bus



## Information

This icon is designed for a **sub-bus** in a signal connector.

## Modelica.Math

**Library of mathematical functions (e.g., sin, cos) and of functions operating on vectors and matrices**

## Information

This package contains **basic mathematical functions** (such as  $\sin(\dots)$ ), as well as functions operating on **vectors** and **matrices**.

### Main Author:

[Martin Otter](#)

Deutsches Zentrum für Luft und Raumfahrt e.V. (DLR)  
Institut für Robotik und Mechatronik  
Postfach 1116  
D-82230 Wessling  
Germany  
email: [Martin.Otter@dlr.de](mailto:Martin.Otter@dlr.de)

Copyright © 1998-2007, Modelica Association and DLR.

*This Modelica package is free software; it can be redistributed and/or modified under the terms of the Modelica license, see the license conditions and the accompanying disclaimer [here](#).*

## Package Content

Name	Description
Vectors	Library of functions operating on vectors
Matrices	Library of functions operating on matrices
sin	Sine

 cos	Cosine
 tan	Tangent (u shall not be -pi/2, pi/2, 3*pi/2, ...)
 asin	Inverse sine (-1 <= u <= 1)
 acos	Inverse cosine (-1 <= u <= 1)
 atan	Inverse tangent
 atan2	Four quadrant inverse tangent
 atan3	Four quadrant inverse tangens (select solution that is closest to given angle y0)
 sinh	Hyperbolic sine
 cosh	Hyperbolic cosine
 tanh	Hyperbolic tangent
 asinh	Inverse of sinh (area hyperbolic sine)
 acosh	Inverse of cosh (area hyperbolic cosine)
 exp	Exponential, base e
 log	Natural (base e) logarithm (u shall be > 0)
 log10	Base 10 logarithm (u shall be > 0)
 baselcon1	Basic icon for mathematical function with y-axis on left side
 baselcon2	Basic icon for mathematical function with y-axis in middle
 templInterpol1	Temporary function for linear interpolation (will be removed)
 templInterpol2	Temporary function for vectorized linear interpolation (will be removed)

## Modelica.Math.Vectors

### Library of functions operating on vectors

#### Information

##### Library content

This library provides functions operating on vectors:

Function	Description
isEqual(v1, v2)	Determines whether two vectors have the same size and elements
norm(v,p)	p-norm of vector v
length(v)	Length of vector v (= norm(v,2), but inlined and therefore usable in symbolic manipulations)
normalize(v)	Return normalized vector such that length = 1 and prevent zero-division for zero vector
reverse(v)	Reverse vector elements
sort(v)	Sort elements of vector in ascending or descending order

#### See also

Matrices

## Package Content

Name	Description
 <code>isEqual</code>	Determine if two Real vectors are numerically identical
 <code>norm</code>	Return the p-norm of a vector
 <code>length</code>	Return length of a vector (better as norm(), if further symbolic processing is performed)
 <code>normalize</code>	Return normalized vector such that length = 1 and prevent zero-division for zero vector
 <code>reverse</code>	Reverse vector elements (e.g. v[1] becomes last element)
 <code>sort</code>	Sort elements of vector in ascending or descending order

### Modelica.Math.Vectors.isEqual

Determine if two Real vectors are numerically identical



#### Information

#### Syntax

```
Vectors isEqual(v1, v2);
Vectors isEqual(v1, v2, eps=0);
```

#### Description

The function call "Vectors isEqual(v1, v2)" returns **true**, if the two Real vectors v1 and v2 have the same dimensions and the same elements. Otherwise the function returns **false**. Two elements e1 and e2 of the two vectors are checked on equality by the test "abs(e1-e2) ≤ eps", where "eps" can be provided as third argument of the function. Default is "eps = 0".

Modelica.Utilities.Strings.isEqual

#### Example

```
Real v1[3] = {1, 2, 3};
Real v2[3] = {1, 2, 3, 4};
Real v3[3] = {1, 2, 3.0001};
Boolean result;

algorithm
  result := Vectors isEqual(v1,v2);      // = false
  result := Vectors isEqual(v1,v3);      // = false
  result := Vectors isEqual(v1,v1);      // = true
  result := Vectors isEqual(v1,v3,0.1); // = true
```

#### See also

Matrices.isEqual, Strings.isEqual

#### Inputs

Name	Default	Description
v1[:]		First vector
v2[:]		Second vector (may have different length as v1)

eps	0	Two elements e1 and e2 of the two vectors are identical if $\text{abs}(e1-e2) \leq \text{eps}$
-----	---	--

## Outputs

Name	Description
result	= true, if vectors have the same length and the same elements

## Modelica.Math.Vectors.norm

Return the p-norm of a vector



## Information

### Syntax

```
Vectors.norm(v);
Vectors.norm(v, p=2); // 1 ≤ p ≤ ∞
```

### Description

The function call "Vectors.norm(v)" returns the **Euclidean norm** " $\sqrt{v^*v}$ " of vector v. With the optional second argument "p", any other p-norm can be computed:

$$\|v\|_p = \left( \sum_{i=1}^n |v_i|^p \right)^{\frac{1}{p}}, \quad 1 \leq p \leq \infty$$

Besides the Euclidean norm (p=2), also the 1-norm and the infinity-norm are sometimes used:

<b>1-norm</b>	= sum(abs(v))	norm(v,1)
<b>2-norm</b>	= sqrt(v*v)	norm(v) or norm(v,2)
<b>infinity-norm</b>	= max(abs(v))	norm(v,Modelica.Constants.inf)

Note, for any vector norm the following inequality holds:

$$\text{norm}(v1+v2, p) \leq \text{norm}(v1, p) + \text{norm}(v2, p)$$

### Example

```
v = {2, -4, -2, -1};
norm(v,1); // = 9
norm(v,2); // = 5
norm(v); // = 5
norm(v,10.5); // = 4.00052597412635
norm(v,Modelica.Constants.inf); // = 4
```

### See also

Matrices.norm

## Inputs

Name	Default	Description
------	---------	-------------

## 382 Modelica.Math.Vectors.norm

---

v[:]		Vector
p	2	Type of p-norm (often used: 1, 2, or Modelica.Constants.inf)

### Outputs

Name	Description
result	p-norm of vector v

---

## Modelica.Math.Vectors.length

Return length of a vector (better as norm(), if further symbolic processing is performed)



### Information

### Syntax

```
Vectors.length(v);
```

### Description

The function call "Vectors.length(v)" returns the **Euclidean length** " $\sqrt{v \cdot v}$ " of vector v. The function call is equivalent to Vectors.norm(v). The advantage of length(v) over norm(v)"is that function length(..) is implemented in one statement and therefore the function is usually automatically inlined. Further symbolic processing is therefore possible, which is not the case with function norm(..).

### Example

```
v = {2, -4, -2, -1};  
length(v); // = 5
```

### See also

[Vectors.norm](#)

### Inputs

Name	Default	Description
v[:]		Vector

### Outputs

Name	Description
result	Length of vector v

---

## Modelica.Math.Vectors.normalize

Return normalized vector such that length = 1 and prevent zero-division for zero vector



## Information

### Syntax

```
Vectors.normalize(v);
Vectors.normalize(v,eps=100*Modelica.Constants.eps);
```

### Description

The function call "Vectors.normalize(v)" returns the **unit vector** " $v/\text{length}(v)$ " of vector v. If  $\text{length}(v)$  is close to zero (more precisely, if  $\text{length}(v) < \text{eps}$ ), v is returned in order to avoid a division by zero. For many applications this is useful, because often the unit vector  $e = v/\text{length}(v)$  is used to compute a vector  $x^*e$ , where the scalar x is in the order of  $\text{length}(v)$ , i.e.,  $x^*e$  is small, when  $\text{length}(v)$  is small and then it is fine to replace e by v to avoid a division by zero.

Since the function is implemented in one statement, it is usually inlined and therefore symbolic processing is possible.

### Example

```
normalize({1,2,3}); // = {0.267, 0.534, 0.802}
normalize({0,0,0}); // = {0,0,0}
```

### See also

[Vectors.length](#)

### Inputs

Name	Default	Description
v[:]		Vector
eps	100*Modelica.Constants.eps	if $ v  < \text{eps}$ then result = v

### Outputs

Name	Description
result[size(v, 1)]	Input vector v normalized to length=1

---

## Modelica.Math.Vectors.reverse

Reverse vector elements (e.g. v[1] becomes last element)



### Information

### Syntax

```
Vectors.reverse(v);
```

### Description

The function call "Vectors.reverse(v)" returns the vector elements in reverse order.

**Example**

```
reverse({1,2,3,4}); // = {4,3,2,1}
```

**Inputs**

Name	Default	Description
v[:]		Vector

**Outputs**

Name	Description
result[size(v, 1)]	Elements of vector v in reversed order

---

**Modelica.Math.Vectors.sort**

Sort elements of vector in ascending or descending order

**Information****Syntax**

```
sorted_v = Vectors.sort(v);
(sorted_v, indices) = Vectors.sort(v, ascending=true);
```

**Description**

Function **sort(..)** sorts a Real vector v in ascending order and returns the result in sorted\_v. If the optional argument "ascending" is **false**, the vector is sorted in descending order. In the optional second output argument the indices of the sorted vector with respect to the original vector are given, such that sorted\_v = v[indices].

**Example**

```
(v2, i2) := Vectors.sort({-1, 8, 3, 6, 2});
-> v2 = {-1, 2, 3, 6, 8}
    i2 = {1, 5, 3, 4, 2}
```

**Inputs**

Name	Default	Description
v[:]		Vector to be sorted
ascending	true	= true if ascending order, otherwise descending order

**Outputs**

Name	Description
sorted_v[size(v, 1)]	Sorted vector
indices[size(v, 1)]	sorted_v = v[indices]

---

## Modelica.Math.Matrices

### Library of functions operating on matrices

#### Information

#### Library content

This library provides functions operating on matrices:

<i>Function</i>	<i>Description</i>
<code>isEqual(M1, M2)</code>	Determines whether two matrices have the same size and elements
<code>norm(A)</code>	1-, 2- and infinity-norm of matrix A
<code>sort(M)</code>	Sort rows or columns of matrix in ascending or descending order
<code>solve(A,b)</code>	Solve real system of linear equations $A^*x=b$ with a b vector
<code>solve2(A,B)</code>	Solve real system of linear equations $A^*X=B$ with a B matrix
<code>leastSquares(A,b)</code>	Solve overdetermined or underdetermined real system of linear equations $A^*x=b$ in a least squares sense
<code>equalityLeastSquares(A,a,B,b)</code>	Solve a linear equality constrained least squares problem: $\min A^*x-a ^2$ subject to $B^*x=b$
<code>(LU,p,info) = LU(A)</code>	LU decomposition of square or rectangular matrix
<code>LU_solve(LU,p,b)</code>	Solve real system of linear equations $P^*L^*U^*x=b$ with a b vector and an LU decomposition from "LU(..)"
<code>LU_solve2(LU,p,B)</code>	Solve real system of linear equations $P^*L^*U^*X=B$ with a B matrix and an LU decomposition from "LU(..)"
<code>(Q,R,p) = QR(A)</code>	QR decomposition with column pivoting of rectangular matrix ( $Q^*R = A[: , p]$ )
<code>eval = eigenValues(A)</code> <code>(eval,evec) = eigenValues(A)</code>	Compute eigenvalues and optionally eigenvectors for a real, nonsymmetric matrix
<code>eigenValueMatrix(eigen)</code>	Return real valued block diagonal matrix J of eigenvalues of matrix A ( $A=V^*J^*V^{-1}$ )
<code>sigma = singularValues(A)</code> <code>(sigma,U,VT) = singularValues(A)</code>	Compute singular values and optionally left and right singular vectors
<code>det(A)</code>	Determinant of a matrix (do <b>not</b> use; use rank(..))
<code>inv(A)</code>	Inverse of a matrix
<code>rank(A)</code>	Rank of a matrix
<code>balance(A)</code>	Balance a square matrix to improve the condition
<code>exp(A)</code>	Compute the exponential of a matrix by adaptive Taylor series expansion with scaling and balancing
<code>(P, G) = integralExp(A,B)</code>	Compute the exponential of a matrix and its integral
<code>(P, G, GT) = integralExpT(A,B)</code>	Compute the exponential of a matrix and two integrals

Most functions are solely an interface to the external LAPACK library (<http://www.netlib.org/lapack>). The details of this library are described in:

Anderson E., Bai Z., Bischof C., Blackford S., Demmel J., Dongarra J., Du Croz J., Greenbaum A., Hammarling S., McKenney A., and Sorensen D.:

**Lapack Users' Guide.** Third Edition, SIAM, 1999.

#### See also

[Vectors](#)

## Package Content

Name	Description
 <code>isEqual</code>	Compare whether two Real matrices are identical
 <code>norm</code>	Returns the norm of a matrix
 <code>sort</code>	Sort rows or columns of matrix in ascending or descending order
 <code>solve</code>	Solve real system of linear equations $A*x=b$ with a b vector (Gaussian elimination with partial pivoting)
 <code>solve2</code>	Solve real system of linear equations $A*X=B$ with a B matrix (Gaussian elimination with partial pivoting)
 <code>leastSquares</code>	Solve overdetermined or underdetermined real system of linear equations $A*x=b$ in a least squares sense (A may be rank deficient)
 <code>equalityLeastSquares</code>	Solve a linear equality constrained least squares problem
 <code>LU</code>	LU decomposition of square or rectangular matrix
 <code>LU_solve</code>	Solve real system of linear equations $P*L*U*x=b$ with a b vector and an LU decomposition (from <code>LU(..)</code> )
 <code>LU_solve2</code>	Solve real system of linear equations $P*L*U*X=B$ with a B matrix and an LU decomposition (from <code>LU(..)</code> )
 <code>QR</code>	QR decomposition of a square matrix with column pivoting ( $A(:,p) = Q*R$ )
 <code>eigenValues</code>	Compute eigenvalues and eigenvectors for a real, nonsymmetric matrix
 <code>eigenValueMatrix</code>	Return real valued block diagonal matrix J of eigenvalues of matrix A ( $A=V^*J*V^{-1}$ )
 <code>singularValues</code>	Compute singular values and left and right singular vectors
 <code>det</code>	Determinant of a matrix (computed by LU decomposition)
 <code>inv</code>	Inverse of a matrix (try to avoid, use function <code>solve(..)</code> instead)
 <code>rank</code>	Rank of a matrix (computed with singular values)
 <code>balance</code>	Balancing of matrix A to improve the condition of A
 <code>exp</code>	Compute the exponential of a matrix by adaptive Taylor series expansion with scaling and balancing
 <code>integralExp</code>	Computation of the transition-matrix phi and its integral gamma
 <code>integralExpT</code>	Computation of the transition-matrix phi and the integral gamma and gamma1

### Modelica.Math.Matrices.isEqual

Compare whether two Real matrices are identical



#### Information

#### Syntax

```
Matrices.isEqual(M1, M2);
Matrices.isEqual(M1, M2, eps=0);
```

#### Description

The function call "Matrices.`isEqual` (M1, M2)" returns **true**, if the two Real matrices M1 and M2 have

the same dimensions and the same elements. Otherwise the function returns **false**. Two elements e1 and e2 of the two matrices are checked on equality by the test "abs(e1-e2) ≤ eps", where "eps" can be provided as third argument of the function. Default is "eps = 0".

### Example

```
Real A1[2,2] = [1,2; 3,4];
Real A2[3,2] = [1,2; 3,4; 5,6];
Real A3[2,2] = [1,2, 3,4.0001];
Boolean result;
algorithm
  result := Matrices.isEqual(M1,M2);           // = false
  result := Matrices.isEqual(M1,M3);           // = false
  result := Matrices.isEqual(M1,M1);           // = true
  result := Matrices.isEqual(M1,M3,0.1);        // = true
```

### See also

[Vectors.isEqual](#), [Strings.isEqual](#)

### Inputs

Name	Default	Description
M1[:, :]		First matrix
M2[:, :]		Second matrix (may have different size as M1)
eps	0	Two elements e1 and e2 of the two matrices are identical if $\text{abs}(e1-e2) \leq \text{eps}$

### Outputs

Name	Description
result	= true, if matrices have the same size and the same elements

---

## Modelica.Math.Matrices.norm

Returns the norm of a matrix



### Information

#### Syntax

```
Matrices.norm(A);
Matrices.norm(A, p=2);
```

#### Description

The function call "Matrices.norm(A)" returns the 2-norm of matrix A, i.e., the largest singular value of A. The function call "Matrices.norm(A, p)" returns the p-norm of matrix A. The only allowed values for p are

- "p=1": the largest column sum of A
- "p=2": the largest singular value of A
- "p=Modelica.Constants.inf": the largest row sum of A

Note, for any matrices A1, A2 the following inequality holds:

## 388 Modelica.Math.Matrices.norm

```
Matrices.norm(A1+A2,p) ≤ Matrices.norm(A1,p) + Matrices.norm(A2,p)
```

Note, for any matrix A and vector v the following inequality holds:

```
Vectors.norm(A*v,p) ≤ Matrices.norm(A,p)*Vectors.norm(A,p)
```

### Inputs

Name	Default	Description
A[:, :]		Input matrix
p	2	Type of p-norm (only allowed: 1, 2 or Modelica.Constants.inf)

### Outputs

Name	Description
result	p-norm of matrix A

---

## Modelica.Math.Matrices.sort

Sort rows or columns of matrix in ascending or descending order



### Information

#### Syntax

```
sorted_M = Matrices.sort(M);
(sorted_M, indices) = Matrices.sort(M, sortRows=true, ascending=true);
```

#### Description

Function **sort(..)** sorts the rows of a Real matrix M in ascending order and returns the result in sorted\_M. If the optional argument "sortRows" is **false**, the columns of the matrix are sorted. If the optional argument "ascending" is **false**, the rows or columns are sorted in descending order. In the optional second output argument, the indices of the sorted rows or columns with respect to the original matrix are given, such that

```
sorted_M = if sortedRow then M[indices,:] else M[:,indices];
```

#### Example

```
(M2, i2) := Matrices.sort([2, 1, 0;
                            2, 0, -1]);
-> M2 = [2, 0, -1;
          2, 1, 0 ];
i2 = {2,1};
```

### Inputs

Name	Default	Description
M[:, :]		Matrix to be sorted
sortRows	true	= true if rows are sorted, otherwise columns
ascending	true	= true if ascending order, otherwise descending order

## Outputs

Name	Description
sorted_M[size(M, 1), size(M, 2)]	Sorted matrix
indices[if sortRows then size(M, 1) else size(M, 2)]	sorted_M = if sortRows then M[indices,:] else M[:,indices]

## Modelica.Math.Matrices.solve

Solve real system of linear equations  $A^*x=b$  with a b vector (Gaussian elemination with partial pivoting)



## Information

### Syntax

```
Matrices.solve(A, b);
```

### Description

This function call returns the solution  $x$  of the linear system of equations

$$A^*x = b$$

If a unique solution  $x$  does not exist (since  $A$  is singular), an exception is raised.

Note, the solution is computed with the LAPACK function "dgesv", i.e., by Gaussian elemination with partial pivoting.

## Example

```
Real A[3,3] = [1,2,3;
               3,4,5;
               2,1,4];
Real b[3] = {10,22,12};
Real x[3];
algorithm
  x := Matrices.solve(A,b); // x = {3,2,1}
```

## See also

[Matrices.LU](#), [Matrices.LU\\_solve](#)

## Inputs

Name	Default	Description
A[:, size(A, 1)]		Matrix A of $A^*x = b$
b[size(A, 1)]		Vector b of $A^*x = b$

## Outputs

Name	Description
x[size(b, 1)]	Vector x such that $A^*x = b$

**Modelica.Math.Matrices.solve2**

Solve real system of linear equations  $A \cdot X = B$  with a  $B$  matrix (Gaussian elimination with partial pivoting)

**Information****Syntax**

```
Matrices.solve2(A, b);
```

**Description**

This function call returns the solution  $X$  of the linear system of equations

$$A \cdot X = B$$

If a unique solution  $X$  does not exist (since  $A$  is singular), an exception is raised.

Note, the solution is computed with the LAPACK function "dgesv", i.e., by Gaussian elimination with partial pivoting.

**Example**

```
Real A[3,3] = [1,2,3;
                3,4,5;
                2,1,4];
Real B[3,2] = [10, 20;
                22, 44;
                12, 24];
Real X[3,2];
algorithm
  (LU, pivots) := Matrices.LU(A);
  X := Matrices.solve2(A, B1); /* X = [3, 6;
                                2, 4;
                                1, 2] */
```

**See also**

[Matrices.LU](#), [Matrices.LU\\_solve2](#)

**Inputs**

Name	Default	Description
$A[:, \text{size}(A, 1)]$		Matrix $A$ of $A \cdot X = B$
$B[\text{size}(A, 1), :]$		Matrix $B$ of $A \cdot X = B$

**Outputs**

Name	Description
$X[\text{size}(B, 1), \text{size}(B, 2)]$	Matrix $X$ such that $A \cdot X = B$

## Modelica.Math.Matrices.leastSquares

Solve overdetermined or underdetermined real system of linear equations  $A^*x=b$  in a least squares sense (A may be rank deficient)



### Information

#### Syntax

```
x = Matrices.leastSquares(A, b);
```

#### Description

A linear system of equations  $A^*x = b$  has no solutions or infinitely many solutions if A is not square. Function "leastSquares" returns a solution in a least square sense:

<code>size(A,1) &gt; size(A,2):</code>	returns x such that $ A^*x - b ^2$ is a minimum
<code>size(A,1) = size(A,2):</code>	returns x such that $A^*x = b$
<code>size(A,1) &lt; size(A,2):</code>	returns x such that $ x ^2$ is a minimum for all vectors x that fulfill $A^*x = b$

Note, the solution is computed with the LAPACK function "dgelsx", i.e., QR or LQ factorization of A with column pivoting. If A does not have full rank, the solution is not unique and from the infinitely many solutions the one is selected that minimizes both  $|x|^2$  and  $|A^*x - b|^2$ .

#### Inputs

Name	Default	Description
<code>A[:, :]</code>		Matrix A
<code>b[size(A, 1)]</code>		Vector b

#### Outputs

Name	Description
<code>x[size(A, 2)]</code>	Vector x such that $\min A^*x-b ^2$ if $\text{size}(A,1) \geq \text{size}(A,2)$ or $\min x ^2$ and $A^*x=b$ , if $\text{size}(A,1) < \text{size}(A,2)$

---

## Modelica.Math.Matrices.equalityLeastSquares



Solve a linear equality constrained least squares problem

### Information

#### Syntax

```
x = Matrices.equalityLeastSquares(A, a, B, b);
```

#### Description

This function returns the solution  $x$  of the linear equality-constrained least squares problem:

$\min|A^*x - a|^2$  over  $x$ , subject to  $B^*x = b$

## 392 Modelica.Math.Matrices.equalityLeastSquares

---

It is required that the dimensions of A and B fulfill the following relationship:

$$\text{size}(B, 1) \leq \text{size}(A, 2) \leq \text{size}(A, 1) + \text{size}(B, 1)$$

Note, the solution is computed with the LAPACK function "dgglse" using the generalized RQ factorization under the assumptions that B has full row rank (= size(B,1)) and the matrix [A;B] has full column rank (= size(A,2)). In this case, the problem has a unique solution.

### Inputs

Name	Default	Description
A[:, :]		Minimize $ A^*x - a ^2$
a[size(A, 1)]		
B[:, size(A, 2)]		subject to $B^*x=b$
b[size(B, 1)]		

### Outputs

Name	Description
x[size(A, 2)]	solution vector

---

## Modelica.Math.Matrices.LU

LU decomposition of square or rectangular matrix



### Information

### Syntax

```
(LU, pivots)      = Matrices.LU(A);  
(LU, pivots, info) = Matrices.LU(A);
```

### Description

This function call returns the LU decomposition of a "Real[m,n]" matrix A, i.e.,

$$P^*L^*U = A$$

where **P** is a permutation matrix (implicitly defined by vector **pivots**), **L** is a lower triangular matrix with unit diagonal elements (lower trapezoidal if  $m > n$ ), and **U** is an upper triangular matrix (upper trapezoidal if  $m < n$ ). Matrices **L** and **U** are stored in the returned matrix **LU** (the diagonal of **L** is not stored). With the companion function [Matrices.LU\\_solve](#), this decomposition can be used to solve linear systems  $(P^*L^*U)^*x = b$  with different right hand side vectors **b**. If a linear system of equations with just one right hand side vector **b** shall be solved, it is more convenient to just use the function [Matrices.solve](#).

The optional third (Integer) output argument has the following meaning:

info = 0:successful exit

info > 0:  
if info = i,  $U[i,i]$  is exactly zero. The factorization has been completed,  
but the factor **U** is exactly singular, and division by zero will occur  
if it is used to solve a system of equations.

The LU factorization is computed with the LAPACK function "dgetrf", i.e., by Gaussian elimination using partial pivoting with row interchanges. Vector "pivots" are the pivot indices, i.e., for  $1 \leq i \leq \min(m,n)$ , row  $i$  of matrix A was interchanged with row **pivots**[ $i$ ].

## Example

```

Real A[3,3] = [1,2,3;
               3,4,5;
               2,1,4];
Real b1[3] = {10,22,12};
Real b2[3] = { 7,13,10};
Real LU[3,3];
Integer pivots[3];
Real x1[3];
Real x2[3];
algorithm
  (LU, pivots) := Matrices.LU(A);
  x1 := Matrices.LU_solve(LU, pivots, b1); // x1 = {3,2,1}
  x2 := Matrices.LU_solve(LU, pivots, b2); // x2 = {1,0,2}

```

## See also

[Matrices.LU\\_solve](#), [Matrices.solve](#),

## Inputs

Name	Default	Description
A[:, :]		Square or rectangular matrix

## Outputs

Name	Description
LU[size(A, 1), size(A, 2)]	L,U factors (used with LU_solve(..))
pivots[min(size(A, 1), size(A, 2))]	pivot indices (used with LU_solve(..))
info	Information

---

## Modelica.Math.Matrices.LU\_solve

Solve real system of linear equations  $P^*L^*U^*x=b$  with a  $b$  vector and an LU decomposition (from LU(..))



## Information

### Syntax

```
Matrices.LU_solve(LU, pivots, b);
```

### Description

This function call returns the solution  $x$  of the linear systems of equations

$$P^*L^*U^*x = b;$$

where  $P$  is a permutation matrix (implicitly defined by vector `pivots`),  $L$  is a lower triangular matrix with unit diagonal elements (lower trapezoidal if  $m > n$ ), and  $U$  is an upper triangular matrix (upper trapezoidal if  $m < n$ ). The matrices of this decomposition are computed with function [Matrices.LU](#) that returns arguments `LU` and `pivots` used as input arguments of [Matrices.LU\\_solve](#). With [Matrices.LU](#) and [Matrices.LU\\_solve](#) it is possible to efficiently solve linear systems with different right hand side vectors. If

## 394 Modelica.Math.Matrices.LU\_solve

---

a linear system of equations with just one right hand side vector shall be solved, it is more convenient to just use the function [Matrices.solve](#).

If a unique solution  $\mathbf{x}$  does not exist (since the LU decomposition is singular), an exception is raised.

The LU factorization is computed with the LAPACK function "dgetrf", i.e., by Gaussian elimination using partial pivoting with row interchanges. Vector "pivots" are the pivot indices, i.e., for  $1 \leq i \leq \min(m,n)$ , row  $i$  of matrix A was interchanged with row  $\text{pivots}[i]$ .

### Example

```
Real A[3,3] = [1,2,3;
                3,4,5;
                2,1,4];
Real b1[3] = {10,22,12};
Real b2[3] = { 7,13,10};
Real    LU[3,3];
Integer pivots[3];
Real    x1[3];
Real    x2[3];
algorithm
  (LU, pivots) := Matrices.LU(A);
  x1 := Matrices.LU_solve(LU, pivots, b1); // x1 = {3,2,1}
  x2 := Matrices.LU_solve(LU, pivots, b2); // x2 = {1,0,2}
```

### See also

[Matrices.LU](#), [Matrices.solve](#),

### Inputs

Name	Default	Description
LU[:, size(LU, 1)]		L,U factors of <a href="#">Matrices.LU(..)</a> for a square matrix
pivots[size(LU, 1)]		Pivots indices of <a href="#">Matrices.LU(..)</a>
b[size(LU, 1)]		Right hand side vector of $P^*L^*U^*x=b$

### Outputs

Name	Description
x[size(b, 1)]	Solution vector such that $P^*L^*U^*x = b$

---

## Modelica.Math.Matrices.LU\_solve2

Solve real system of linear equations  $P^*L^*U^*X=B$  with a B matrix and an LU decomposition (from [LU\(..\)](#))



### Information

### Syntax

```
Matrices.LU_solve(LU, pivots, B);
```

## Description

This function call returns the solution **X** of the linear systems of equations

$$\mathbf{P}^*\mathbf{L}^*\mathbf{U}^*\mathbf{X} = \mathbf{B};$$

where **P** is a permutation matrix (implicitly defined by vector `pivots`), **L** is a lower triangular matrix with unit diagonal elements (lower trapezoidal if  $m > n$ ), and **U** is an upper triangular matrix (upper trapezoidal if  $m < n$ ). The matrices of this decomposition are computed with function `Matrices.LU` that returns arguments `LU` and `pivots` used as input arguments of `Matrices.LU_solve2`. With `Matrices.LU` and `Matrices.LU_solve2` it is possible to efficiently solve linear systems with different right hand side **matrices**. If a linear system of equations with just one right hand side matrix shall be solved, it is more convenient to just use the function `Matrices.solve2`.

If a unique solution **X** does not exist (since the LU decomposition is singular), an exception is raised.

The LU factorization is computed with the LAPACK function "dgetrf", i.e., by Gaussian elemination using partial pivoting with row interchanges. Vector "pivots" are the pivot indices, i.e., for  $1 \leq i \leq \min(m,n)$ , row  $i$  of matrix A was interchanged with row `pivots[i]`.

## Example

```

Real A[3,3] = [1,2,3;
               3,4,5;
               2,1,4];
Real B1[3] = [10, 20;
               22, 44;
               12, 24];
Real B2[3] = [ 7, 14;
               13, 26;
               10, 20];
Real     LU[3,3];
Integer pivots[3];
Real     X1[3,2];
Real     X2[3,2];
algorithm
  (LU, pivots) := Matrices.LU(A);
  X1 := Matrices.LU_solve2(LU, pivots, B1); /* X1 = [3, 6;
                                                2, 4;
                                                1, 2] */
  X2 := Matrices.LU_solve2(LU, pivots, B2); /* X2 = [1, 2;
                                                0, 0;
                                                2, 4] */

```

## See also

`Matrices.LU`, `Matrices.solve2`,

## Inputs

Name	Default	Description
<code>LU[:, size(LU, 1)]</code>		L,U factors of <code>Matrices.LU(..)</code> for a square matrix
<code>pivots[size(LU, 1)]</code>		Pivots indices of <code>Matrices.LU(..)</code>
<code>B[size(LU, 1), :]</code>		Right hand side matrix of $\mathbf{P}^*\mathbf{L}^*\mathbf{U}^*\mathbf{X}=\mathbf{B}$

## Outputs

Name	Description

X[size(B, 1), size(B, 2)]	Solution matrix such that $P^*L^*U^*X = B$
---------------------------	--

## Modelica.Math.Matrices.QR

QR decomposition of a square matrix with column pivoting ( $A(:,p) = Q*R$ )



### Information

### Syntax

```
(Q, R, p) = Matrices.QR(A);
```

### Description

This function returns the QR decomposition of a rectangular matrix **A** (the number of columns of **A** must be less than or equal to the number of rows):

$$Q^*R = A[:,p]$$

where **Q** is a rectangular matrix that has orthonormal columns and has the same size as **A** ( $Q^T Q = I$ ), **R** is a square, upper triangular matrix and **p** is a permutation vector. Matrix **R** has the following important properties:

- The absolute value of a diagonal element of **R** is the largest value in this row, i.e.,  $\text{abs}(R[i,i]) \geq \text{abs}(R[i,j])$ .
- The diagonal elements of **R** are sorted according to size, such that the largest absolute value is  $\text{abs}(R[1,1])$  and  $\text{abs}(R[i,i]) \geq \text{abs}(R[j,j])$  with  $i < j$ .

This means that if  $\text{abs}(R[i,i]) \leq \varepsilon$  then  $\text{abs}(R[j,k]) \leq \varepsilon$  for  $j \geq i$ , i.e., the  $i$ -th row up to the last row of **R** have small elements and can be treated as being zero. This allows to, e.g., estimate the row-rank of **R** (which is the same row-rank as **A**). Furthermore, **R** can be partitioned in two parts

$$\begin{aligned} A[:,p] = Q * [R_1, R_2; \\ 0, 0] \end{aligned}$$

where **R**<sub>1</sub> is a regular, upper triangular matrix.

Note, the solution is computed with the LAPACK functions "dgeqpf" and "dorgqr", i.e., by Householder transformations with column pivoting. If **Q** is not needed, the function may be called as:  $(, R, p) = QR(A)$ .

### Example

```
Real A[3,3] = [1,2,3;
                3,4,5;
                2,1,4];
Real R[3,3];
algorithm
  (,R) := Matrices.QR(A); // R = [-7.07..., -4.24..., -3.67...;
                           0      , -1.73..., -0.23...;
                           0      ,  0       ,  0.65...];
```

### Inputs

Name	Default	Description
------	---------	-------------

A[:, :]	Rectangular matrix with size(A,1) >= size(A,2)
---------	--

## Outputs

Name	Description
Q[size(A, 1), size(A, 2)]	Rectangular matrix with orthonormal columns such that Q*R=A[:,p]
R[size(A, 2), size(A, 2)]	Square upper triangular matrix
p[size(A, 2)]	Column permutation vector

## Modelica.Math.Matrices.eigenValues

Compute eigenvalues and eigenvectors for a real, nonsymmetric matrix



## Information

### Syntax

```
eigenvalues = Matrices.eigenValues(A);
(eigenvalues, eigenvectors) = Matrices.eigenValues(A);
```

### Description

This function call returns the eigenvalues and optionally the (right) eigenvectors of a square matrix **A**. The first column of "eigenvalues" contains the real and the second column contains the imaginary part of the eigenvalues. If the i-th eigenvalue has no imaginary part, then eigenvectors[:,i] is the corresponding real eigenvector. If the i-th eigenvalue has an imaginary part, then eigenvalues[i+1,:] is the conjugate complex eigenvalue and eigenvectors[:,i] is the real and eigenvectors[:,i+1] is the imaginary part of the eigenvector of the i-th eigenvalue. With function [Matrices.eigenValueMatrix](#), a real block diagonal matrix is constructed from the eigenvalues such that

```
A = eigenvectors * eigenValueMatrix(eigenvalues) * inv(eigenvectors)
```

provided the eigenvector matrix "eigenvectors" can be inverted (an inversion is possible, if all eigenvalues are different and no eigenvalue is zero).

### Example

```
Real A[3,3] = [1,2,3;
                3,4,5;
                2,1,4];
Real eval;
algorithm
  eval := Matrices.eigenValues(A); // eval = [-0.618, 0;
                                    //           8.0 , 0;
                                    //           1.618, 0];
```

i.e., matrix A has the 3 real eigenvalues -0.618, 8, 1.618.

### See also

[Matrices.eigenValueMatrix](#), [Matrices.singularValues](#)

## 398 Modelica.Math.Matrices.eigenValues

---

### Inputs

Name	Default	Description
A[:, size(A, 1)]		Matrix

### Outputs

Name	Description
eigenvalues[size(A, 1), 2]	Eigenvalues of matrix A (Re: first column, Im: second column)
eigenvectors[size(A, 1), size(A, 2)]	Real-valued eigenvector matrix

---

## Modelica.Math.Matrices.eigenValueMatrix

Return real valued block diagonal matrix J of eigenvalues of matrix A ( $A=V^*J^*V^{-1}$ )



### Information

### Syntax

```
Matrices.eigenValueMatrix(eigenvalues);
```

### Description

The function call returns a block diagonal matrix  $J$  from the two-column matrix `eigenvalues` (computed by function `Matrices.eigenValues`). Matrix `eigenvalues` must have the real part of the eigenvalues in the first column and the imaginary part in the second column. If an eigenvalue  $i$  has a vanishing imaginary part, then  $J[i,i] = eigenvalues[i,1]$ , i.e., the diagonal element of  $J$  is the real eigenvalue. Otherwise, eigenvalue  $i$  and conjugate complex eigenvalue  $i+1$  are used to construct a 2 by 2 diagonal block of  $J$ :

```
J[i , i] := eigenvalues[i,1];
J[i , i+1] := eigenvalues[i,2];
J[i+1, i] := eigenvalues[i+1,2];
J[i+1, i+1] := eigenvalues[i+1,1];
```

### See also

`Matrices.eigenValues`

### Inputs

Name	Default	Description
eigenValues[:, 2]		Eigen values from function <code>eigenValues(..)</code> (Re: first column, Im: second column)

### Outputs

Name	Description
$J[\text{size}(\text{eigenValues}, 1), \text{size}(\text{eigenValues}, 1)]$	Real valued block diagonal matrix with eigen values (Re: 1x1 block, Im: 2x2 block)

---

## Modelica.Math.Matrices.singularValues

Compute singular values and left and right singular vectors



## Information

### Syntax

```
sigma = Matrices.singularValues(A);
(sigma, U, VT) = Matrices.singularValues(A);
```

### Description

This function computes the singular values and optionally the singular vectors of matrix A. Basically the singular value decomposition of A is computed, i.e.,

$$\begin{aligned} \mathbf{A} &= \mathbf{U} \Sigma \mathbf{V}^T \\ &= \mathbf{U} * \Sigma * \mathbf{V}^T \end{aligned}$$

where **U** and **V** are orthogonal matrices ( $\mathbf{U}\mathbf{U}^T=\mathbf{I}$ ,  $\mathbf{V}\mathbf{V}^T=\mathbf{I}$ ).  $\Sigma = \text{diag}(\sigma_i)$  has the same size as matrix A with nonnegative diagonal elements in decreasing order and with all other elements zero ( $\sigma_1$  is the largest element). The function returns the singular values  $\sigma_i$  in vector `sigma` and the orthogonal matrices in matrices `U` and `V`.

### Example

```
A = [1, 2, 3, 4;
      3, 4, 5, -2;
      -1, 2, -3, 5];
(sigma, U, VT) = singularValues(A);
results in:
sigma = {8.33, 6.94, 2.31};
i.e.
Sigma = [8.33, 0, 0, 0;
          0, 6.94, 0, 0;
          0, 0, 2.31, 0]
```

### See also

[Matrices.eigenValues](#)

### Inputs

Name	Default	Description
A[:, :]		Matrix

### Outputs

Name	Description
sigma[min(size(A, 1), size(A, 2))]	Singular values
U[size(A, 1), size(A, 1)]	Left orthogonal matrix
VT[size(A, 2), size(A, 2)]	Transposed right orthogonal matrix

## Modelica.Math.Matrices.det

Determinant of a matrix (computed by LU decomposition)



## 400 Modelica.Math.Matrices.det

---

### Information

#### Syntax

```
Matrices.det(A);
```

#### Description

This function call returns the determinant of matrix A computed by a LU decomposition. Usually, this function should never be used, because there are nearly always better numerical algorithms as by computing the determinant. E.g., use function [Matrices.rank](#) to compute the rank of a matrix.

#### See also

[Matrices.rank](#), [Matrices.solve](#)

### Inputs

Name	Default	Description
A[:, size(A, 1)]		

### Outputs

Name	Description
result	Determinant of matrix A

---

## Modelica.Math.Matrices.inv

Inverse of a matrix (try to avoid, use function [solve\(..\)](#) instead)



### Information

#### Inputs

Name	Default	Description
A[:, size(A, 1)]		

### Outputs

Name	Description
invA[size(A, 1), size(A, 2)]	Inverse of matrix A

---

## Modelica.Math.Matrices.rank

Rank of a matrix (computed with singular values)



### Information

#### Inputs

Name	Default	Description

A[:, :]	Matrix	
eps	0	If $\text{eps} > 0$ , the singular values are checked against $\text{eps}$ ; otherwise $\text{eps} = \max(\text{size}(A)) * \text{norm}(A) * \text{Modelica.Constants.eps}$ is used

## Outputs

Name	Description
result	Rank of matrix A

---

## Modelica.Math.Matrices.balance

Balancing of matrix A to improve the condition of A



## Information

The function transforms the matrix A, so that the norm of the i-th column is nearby the i-th row.  $(D, B) = \text{Matrices.balance}(A)$  returns a vector D, such that  $B = \text{inv}(\text{diagonal}(D)) * A * \text{diagonal}(D)$  has better condition. The elements of D are multiples of 2. Balancing attempts to make the norm of each row equal to the norm of the belonging column. Balancing is used to minimize roundoff errors induced through large matrix calculations like Taylor-series approximation or computation of eigenvalues.

### Example:

```

- A = [1, 10, 1000; .01, 0, 10; .005, .01, 10]
- Matrices.norm(A, 1);
  = 1020.0
- (T, B)=Matrices.balance(A)
- T
  = {256, 16, 0.5}
- B
  = [1,      0.625,    1.953125;
     0.16,    0,        0.3125;
     2.56,   0.32,     10.0]
- Matrices.norm(B, 1);
  = 12.265625

```

The Algorithm is taken from

H. D. Joos, G. Grbel:

**RASP'91 Regulator Analysis and Synthesis Programs**  
DLR - Control Systems Group 1991

which based on the balanc function from EISPACK.

## Inputs

Name	Default	Description
A[:, size(A, 1)]		

## Outputs

Name	Description
D[size(A, 1)]	diagonal(D)=T is transformation matrix, such that $T^*A^*\text{inv}(T)$ has smaller condition as A
B[size(A, 1), size(A, 1)]	Balanced matrix (= $\text{diagonal}(D)^*A^*\text{inv}(\text{diagonal}(D))$ )

1)]

**Modelica.Math.Matrices.exp**

**Compute the exponential of a matrix by adaptive Taylor series expansion with scaling and balancing**

**Information**

This function computes

$$\Phi = e^{(AT)} = I + AT + \frac{(AT)^2}{2!} + \frac{(AT)^3}{3!} + \dots$$

where  $e=2.71828\dots$ ,  $A$  is an  $n \times n$  matrix with real elements and  $T$  is a real number, e.g., the sampling time.  $A$  may be singular. With the exponential of a matrix it is, e.g., possible to compute the solution of a linear system of differential equations

$$\text{der}(\mathbf{x}) = A \cdot \mathbf{x} \quad -> \quad \mathbf{x}(t_0 + T) = e^{(AT)} \cdot \mathbf{x}(t_0)$$

The function is called as

```
Phi = Matrices.exp(A, T);
```

or

```
M = Matrices.exp(A);
```

what calculates  $M$  as the exponential of matrix  $A$ .

**Algorithmic details:**

The algorithm is taken from

H. D. Joos, G. Gruebel:

**RASP'91 Regulator Analysis and Synthesis Programs**  
DLR - Control Systems Group 1991

The following steps are performed to calculate the exponential of  $A$ :

1. Matrix  $A$  is balanced  
(= is transformed with a diagonal matrix  $D$ , such that  $\text{inv}(D) \cdot A \cdot D$  has a smaller condition as  $A$ ).
2. The scalar  $T$  is divided by a multiple of 2 such that  $\text{norm}(\text{inv}(D) \cdot A \cdot D \cdot T / 2^k) < 0.5$ . Note, that (1) and (2) are implemented such that no round-off errors are introduced.
3. The matrix from (2) is approximated by explicitly performing the Taylor series expansion with a variable number of terms. Truncation occurs if a new term does no longer contribute to the value of  $\Phi$  from the previous iteration.
4. The resulting matrix is transformed back, by reverting the steps of (2) and (1).

In several sources it is not recommended to use Taylor series expansion to calculate the exponential of a matrix, such as in 'C.B. Moler and C.F. Van Loan: Nineteen dubious ways to compute the exponential of a matrix. SIAM Review 20, pp. 801-836, 1979' or in the documentation of m-file `expm2` in Matlab version 6 (<http://www.MathWorks.com>) where it is stated that 'As a practical numerical method, this is often slow and inaccurate'. These statements are valid for a direct implementation of the Taylor series expansion, but *not* for the implementation variant used in this function.

## Inputs

Name	Default	Description
A[:, size(A, 1)]		
T	1	

## Outputs

Name	Description
phi[size(A, 1), size(A, 1)]	= exp(A*T)

## Modelica.Math.Matrices.integralExp



Computation of the transition-matrix phi and its integral gamma

## Information

The function uses a Taylor series expansion with Balancing and scaling/squaring to approximate the integral  $\Psi$  of the matrix exponential  $\Phi = e^{\lambda}(AT)$ :

$$\Psi = \int(e^{\lambda}(As))ds = IT + \frac{AT^2}{2!} + \frac{A^2 * T^3}{3!} + \dots + \frac{A^k * T^{(k+1)}}{(k+1)!}$$

$\Phi$  is calculated through  $\Phi = I + A*\Psi$ , so A may be singular.  $\Gamma$  is simple  $\Psi*B$ .

The algorithm runs in the following steps:

1. Balancing
2. Scaling
3. Taylor series expansion
4. Re-scaling
5. Re-Balancing

Balancing put the bad condition of a square matrix A into a diagonal transformation matrix D. This reduce the effort of following calculations. Afterwards the result have to be re-balanced by transformation  $D^{-1}Atransf *inv(D)$ .

Scaling halven T k-times, until the norm of  $A*T$  is less than 0.5. This guarantees minimum rounding errors in the following series expansion. The re-scaling based on the equation  $\exp(A^2T) = \exp(AT)^2$ . The needed re-scaling formula for psi thus becomes:

$$\begin{aligned}\Phi &= \Phi' * \Phi' \\ I + A*\Psi &= I + 2A*\Psi' + A^2*\Psi'^2 \\ \Psi &= A*\Psi'^2 + 2*\Psi'\end{aligned}$$

where  $\Psi'$  is the scaled result from the series expansion while  $\Psi$  is the re-scaled matrix.

The function is normally used to discretize a state-space system as the zero-order-hold equivalent:

$$\begin{aligned}x(k+1) &= \Phi*x(k) + \Gamma*u(k) \\ y(k) &= C*x(k) + D*u(k)\end{aligned}$$

The zero-order-hold sampling, also known as step-invariant method, gives exact values of the state variables, under the assumption that the control signal u is constant between the sampling instants. Zero-order-hold sampling is described in

K. J. Astrom, B. Wittenmark:

**Computer Controlled Systems - Theory and Design**  
Third Edition, p. 32

## 404 Modelica.Math.Matrices.integralExp

---

### Syntax:

```
(phi, gamma) = Matrices.expIntegral(A, B, T)
    A,phi: [n,n] square matrices
    B,gamma: [n,m] input matrix
    T: scalar, e.g. sampling time
```

The Algorithm to calculate psi is taken from

H. D. Joos, G. Gruebel:

**RASP'91 Regulator Analysis and Synthesis Programs**  
DLR - Control Systems Group 1991

### Inputs

Name	Default	Description
A[:, size(A, 1)]		
B[size(A, 1), :]		
T	1	

### Outputs

Name	Description
phi[size(A, 1), size(A, 1)]	= exp(A*T)
gamma[size(A, 1), size(B, 2)]	= integral(phi)*B

---

## Modelica.Math.Matrices.integralExpT

Computation of the transition-matrix phi and the integral gamma and gamma1



### Information

The function calculates the matrices phi,gamma,gamma1 through the equation:

$$[\phi \gamma \gamma_1] = [\mathbf{I} \ 0 \ 0] * \exp\left(\begin{bmatrix} A & B & 0 \\ 0 & 0 & I \\ 0 & 0 & 0 \end{bmatrix} * T\right)$$

### Syntax:

```
(phi, gamma, gamma1) = Matrices.ExpIntegral2(A, B, T)
    A,phi: [n,n] square matrices
    B,gamma,gamma1: [n,m] matrices
    T: scalar, e.g. sampling time
```

The matrices define the discretized first-order-hold equivalent of a state-space system:

$$x(k+1) = \phi * x(k) + \gamma * u(k) + \gamma_1 / T * (u(k+1) - u(k))$$

The first-order-hold sampling, also known as ramp-invariant method, gives more smooth control signals as the ZOH equivalent. First-order-hold sampling is described in

K. J. Astrom, B. Wittenmark:

**Computer Controlled Systems - Theory and Design**  
Third Edition, p. 256

## Inputs

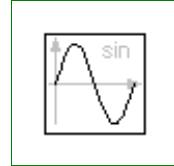
Name	Default	Description
A[:, size(A, 1)]		
B[size(A, 1), :]		
T	1	

## Outputs

Name	Description
phi[size(A, 1), size(A, 1)]	= exp(A*T)
gamma[size(A, 1), size(B, 2)]	= integral(phi)*B
gamma1[size(A, 1), size(B, 2)]	= integral((T-t)*exp(A*t))*B

## Modelica.Math.sin

Sine



## Information

This function returns  $y = \sin(u)$ , with  $-\infty < u < \infty$ :



## Inputs

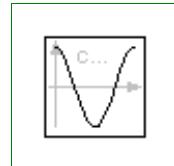
Name	Default	Description
u		[rad]

## Outputs

Name	Description
y	

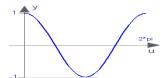
## Modelica.Math.cos

Cosine



## Information

This function returns  $y = \cos(u)$ , with  $-\infty < u < \infty$ :



## Inputs

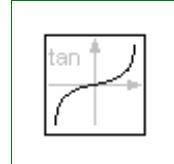
Name	Default	Description
u		[rad]

## Outputs

Name	Description
y	

## Modelica.Math.tan

Tangent ( $u$  shall not be  $-\pi/2, \pi/2, 3\pi/2, \dots$ )



## Information

This function returns  $y = \tan(u)$ , with  $-\infty < u < \infty$  (if  $u$  is a multiple of  $(2n-1)\pi/2$ ,  $y = \tan(u)$  is  $\pm\infty$ ).



## Inputs

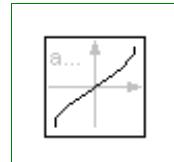
Name	Default	Description
u		[rad]

## Outputs

Name	Description
y	

## Modelica.Math.asin

Inverse sine ( $-1 \leq u \leq 1$ )



## Information

This function returns  $y = \text{asin}(u)$ , with  $-1 \leq u \leq +1$ :



## Inputs

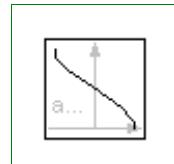
Name	Default	Description
u		

## Outputs

Name	Description
y	[rad]

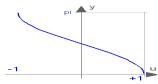
## Modelica.Math.acos

Inverse cosine ( $-1 \leq u \leq 1$ )



## Information

This function returns  $y = \text{acos}(u)$ , with  $-1 \leq u \leq +1$ :



## Inputs

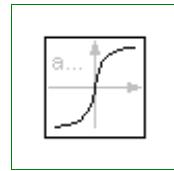
Name	Default	Description
u		

## Outputs

Name	Description
y	[rad]

## Modelica.Math.atan

Inverse tangent



## Information

This function returns  $y = \text{atan}(u)$ , with  $-\infty < u < \infty$ :



## Inputs

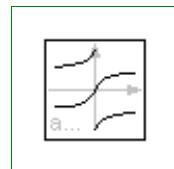
Name	Default	Description
u		

## Outputs

Name	Description
y	[rad]

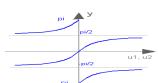
## Modelica.Math.atan2

Four quadrant inverse tangent



## Information

This function returns  $y = \text{atan2}(u_1, u_2)$  such that  $\tan(y) = u_1/u_2$  and  $y$  is in the range  $-\pi < y \leq \pi$ .  $u_2$  may be zero, provided  $u_1$  is not zero. Usually  $u_1, u_2$  is provided in such a form that  $u_1 = \sin(y)$  and  $u_2 = \cos(y)$ :



## Inputs

Name	Default	Description
$u_1$		
$u_2$		

## 408 Modelica.Math.atan2

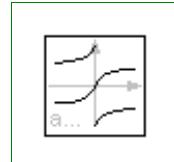
u1		
u2		

### Outputs

Name	Description
y	[rad]

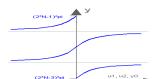
## Modelica.Math.atan3

Four quadrant inverse tangens (select solution that is closest to given angle y0)



### Information

This function returns  $y = \text{atan3}(u1, u2, y0)$  such that  $\tan(y) = u1/u2$  and  $y$  is in the range:  $-\pi < y - y_0 < \pi$ .  $u2$  may be zero, provided  $u1$  is not zero. The difference to Modelica.Math.atan2(..) is the optional third argument  $y0$  that allows to specify which of the infinite many solutions shall be returned:



### Inputs

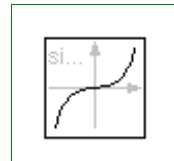
Name	Default	Description
u1		
u2		
y0	0	y shall be in the range: $-\pi < y - y_0 < \pi$ [rad]

### Outputs

Name	Description
y	[rad]

## Modelica.Math.sinh

Hyperbolic sine



### Information

This function returns  $y = \sinh(u)$ , with  $-\infty < u < \infty$ :



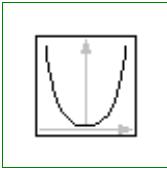
### Inputs

Name	Default	Description
u		

### Outputs

Name	Description

y	
---	--

**Modelica.Math.cosh****Hyperbolic cosine****Information**

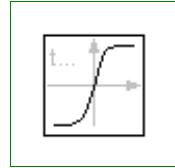
This function returns  $y = \cosh(u)$ , with  $-\infty < u < \infty$ :

**Inputs**

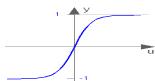
Name	Default	Description
u		

**Outputs**

Name	Description
y	

**Modelica.Math.tanh****Hyperbolic tangent****Information**

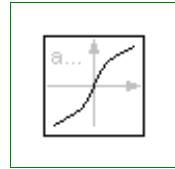
This function returns  $y = \tanh(u)$ , with  $-\infty < u < \infty$ :

**Inputs**

Name	Default	Description
u		

**Outputs**

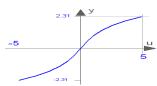
Name	Description
y	

**Modelica.Math.asinh****Inverse of sinh (area hyperbolic sine)****Information**

The function returns the area hyperbolic sine of its input argument  $u$ . This inverse of  $\sinh(..)$  is unique and there is no restriction on the input argument  $u$  of  $\text{asinh}(u)$  ( $-\infty < u < \infty$ ):

## 410 Modelica.Math.asinh

---



### Inputs

Name	Default	Description
u		

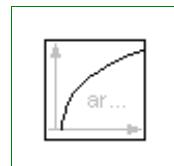
### Outputs

Name	Description
y	

---

## Modelica.Math.acosh

Inverse of cosh (area hyperbolic cosine)



### Information

This function returns the area hyperbolic cosine of its input argument u. The valid range of u is

$$+1 \leq u < +\infty$$

If the function is called with  $u < 1$ , an error occurs. The function  $\cosh(u)$  has two inverse functions (the curve looks similar to a  $\sqrt{..}$  function).  $\text{acosh}(..)$  returns the inverse that is positive. At  $u=1$ , the derivative  $dy/du$  is infinite. Therefore, this function should not be used in a model, if u can become close to 1:



### Inputs

Name	Default	Description
u		

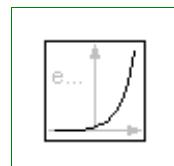
### Outputs

Name	Description
y	

---

## Modelica.Math.exp

Exponential, base e



### Information

This function returns  $y = \exp(u)$ , with  $-\infty < u < \infty$ :



## Inputs

Name	Default	Description
u		

## Outputs

Name	Description
y	

## Modelica.Math.log

Natural (base e) logarithm (u shall be > 0)



## Information

This function returns  $y = \ln(10)$  (the natural logarithm of u), with  $u > 0$ :



## Inputs

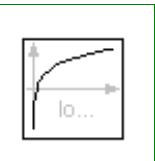
Name	Default	Description
u		

## Outputs

Name	Description
y	

## Modelica.Math.log10

Base 10 logarithm (u shall be > 0)



## Information

This function returns  $y = \log10(u)$ , with  $u > 0$ :

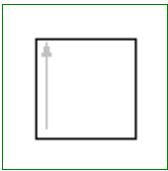


## Inputs

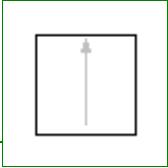
Name	Default	Description
u		

## Outputs

Name	Description
y	

**Modelica.Math.baselcon1****Basic icon for mathematical function with y-axis on left side****Information**

Icon for a mathematical function, consisting of an y-axis on the left side. It is expected, that an x-axis is added and a plot of the function.

**Modelica.Math.baselcon2****Basic icon for mathematical function with y-axis in middle****Modelica.Math.templInterpol1****Temporary function for linear interpolation (will be removed)****Information****Inputs**

Name	Default	Description
u		input value (first column of table)
table[:, :]		table to be interpolated
icol		column of table to be interpolated

**Outputs**

Name	Description
y	interpolated input value (icol column of table)

**Modelica.Math.templInterpol2****Temporary function for vectorized linear interpolation (will be removed)****Information****Inputs**

Name	Default	Description
u		input value (first column of table)
table[:, :]		table to be interpolated
icol[:]		column(s) of table to be interpolated

**Outputs**

Name	Description
y[1, size(icol, 1)]	interpolated input value(s) (column(s) icol of table)

## [Modelica.Mechanics](#)

Library of 1-dim. and 3-dim. mechanical components (multi-body, rotational, translational)

### Information

This package contains components to model the movement of 1-dim. rotational, 1-dim. translational, and 3-dim. **mechanical systems**.

### Package Content

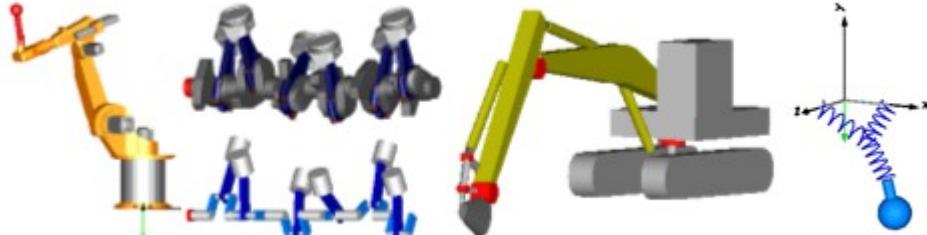
Name	Description
 <a href="#">MultiBody</a>	Library to model 3-dimensional mechanical systems
 <a href="#">Rotational</a>	Library to model 1-dimensional, rotational mechanical systems
 <a href="#">Translational</a>	Library to model 1-dimensional, translational mechanical systems

## [Modelica.Mechanics.MultiBody](#)

Library to model 3-dimensional mechanical systems

### Information

Library **MultiBody** is a **free** Modelica package providing 3-dimensional mechanical components to model in a convenient way **mechanical systems**, such as robots, mechanisms, vehicles. Typical animations generated with this library are shown in the next figure:



For an introduction, have especially a look at:

- [MultiBody.UsersGuide](#) discusses the most important aspects how to use this library.
- [MultiBody.Examples](#) contains examples that demonstrate the usage of this library.

Note, that the MultiBody library replaces the long used ModelicaAdditions.MultiBody library. In [MultiBody.UsersGuide.Upgrade](#) it is described how to upgrade.

Copyright © 1998-2007, Modelica Association and DLR.

*This Modelica package is **free** software; it can be redistributed and/or modified under the terms of the [Modelica license](#), see the license conditions and the accompanying [disclaimer](#) [here](#).*

### Package Content

Name	Description
 <a href="#">UsersGuide</a>	User's Guide of MultiBody Library
 <a href="#">World</a>	World coordinate system + gravity field + default animation definition

 Examples	Examples that demonstrate the usage of the MultiBody library
 Forces	Components that exert forces and/or torques between frames
 Frames	Functions to transform rotational frame quantities
 Interfaces	Connectors and partial models for 3-dim. mechanical components
 Joints	Components that constrain the motion between two frames
 Parts	Rigid components such as bodies with mass and inertia and massless rods
 Sensors	Sensors to measure variables
 Types	Constants and types with choices, especially to build menus
 Visualizers	3-dimensional visual objects used for animation

## Modelica.Mechanics.MultiBody.UsersGuide

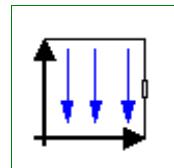
Library **MultiBody** is a **free** Modelica package providing 3-dimensional mechanical components to model in a convenient way **mechanical systems**, such as robots, mechanisms, vehicles. This package contains the user's guide for the MultiBody library.



1. [Tutorial](#) gives an introduction into the most important aspects of the library.
2. [Upgrade](#) describes how to upgrade from former versions, especially from the "old" ModelicaAdditions.MultiBody library.
3. [Release Notes](#) summarizes the differences between different versions of this library.
4. [Literature](#) provides references that have been used to design and implement this library.
5. [Contact](#) provides information about the author of the library as well as acknowledgments.

## Modelica.Mechanics.MultiBody.World

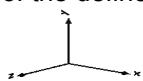
**World coordinate system + gravity field + default animation definition**



### Information

Model **World** represents a global coordinate system fixed in ground. This model serves several purposes:

- It is used as **inertial system** in which the equations of all elements of the MultiBody library are defined.
- It is the world frame of an **animation window** in which all elements of the MultiBody library are visualized.
- It is used to define the **gravity field** in which a multi-body model is present. Default is a uniform gravity field where the gravity acceleration vector  $\mathbf{g}$  is the same at every position. Additionally, a point gravity field can be selected.
- It is used to define **default settings** of animation properties (e.g. the diameter of a sphere representing by default the center of mass of a body, or the diameters of the cylinders representing a revolute joint).
- It is used to define a **visual representation** of the world model (= 3 coordinate axes with labels) and of the defined gravity field.



Since the gravity field function is required from all bodies with mass and the default settings of animation properties are required from nearly every component, exactly one instance of model World needs to be present in every model on the top level. The basic declaration needs to be:

```
inner Modelica.Mechanics.MultiBody.World world
```

Note, it must be an **inner** declaration with instance name **world** in order that this world object can be accessed from all objects in the model. When dragging the "World" object from the package browser into the diagram layer, this declaration is automatically generated (this is defined via annotations in model World).

All vectors and tensors of a mechanical system are resolved in a frame that is local to the corresponding component. Usually, if all relative joint coordinates vanish, the local frames of all components are parallel to each other, as well as to the world frame (this holds as long as a Parts.FixedRotation component is **not** used). In this "reference configuration" it is therefore alternatively possible to resolve all vectors in the world frame, since all frames are parallel to each other. This is often very convenient. In order to give some visual support in such a situation, in the icon of a World instance two axes of the world frame are shown and the labels of these axes can be set via parameters.

## Parameters

Name	Default	Description
enableAnimation	true	= true, if animation of all components is enabled
animateWorld	true	= true, if world coordinate system shall be visualized
animateGravity	true	= true, if gravity field shall be visualized (acceleration vector or field center)
label1	"x"	Label of horizontal axis in icon
label2	"y"	Label of vertical axis in icon
gravityType	GravityTypes.UniformGravity	Type of gravity field
g	9.81	Constant gravity acceleration [m/s <sup>2</sup> ]
n	{0,-1,0}	Direction of gravity resolved in world frame (gravity = g*n/length(n))
mue	3.986e14	Gravity field constant (default = field constant of earth) [m <sup>3</sup> /s <sup>2</sup> ]
driveTrainMechanics3D	false	= true, if 3-dim. mechanical effects of Parts.Mounting1D/Rotor1D/BevelGear1D shall be taken into account

### Animation

if animateWorld = true

axisLength	nominalLength/2	Length of world axes arrows [m]
axisDiameter	axisLength/defaultFrameDiamet..	Diameter of world axes arrows [m]
axisShowLabels	true	= true, if labels shall be shown
axisColor_x	Modelica.Mechanics.MultiBody..	Color of x-arrow
axisColor_y	axisColor_x	
axisColor_z	axisColor_x	Color of z-arrow

if animateGravity = true and gravityType = UniformGravity

gravityArrowTail[3]	{0,0,0}	Position vector from origin of world frame to arrow tail, resolved in world frame [m]
gravityArrowLength	axisLength/2	Length of gravity arrow [m]
gravityArrowDiameter	gravityArrowLength/defaultWi...	Diameter of gravity arrow [m]
gravityArrowColor	{0,230,0}	Color of gravity arrow

if animateGravity = true and gravityType = PointGravity

gravitySphereDiameter	12742000	Diameter of sphere representing gravity center (default = mean diameter of earth)
-----------------------	----------	---

		[m]
gravitySphereColor	{0,230,0}	Color of gravity sphere
<b>Defaults</b>		
nominalLength	1	"Nominal" length of multi-body system [m]
defaultAxisLength	nominalLength/5	Default for length of a frame axis (but not world frame) [m]
defaultJointLength	nominalLength/10	Default for the fixed length of a shape representing a joint [m]
defaultJointWidth	nominalLength/20	Default for the fixed width of a shape representing a joint [m]
defaultForceLength	nominalLength/10	Default for the fixed length of a shape representing a force (e.g. damper) [m]
defaultForceWidth	nominalLength/20	Default for the fixed width of a shape representing a force (e.g. spring, bushing) [m]
defaultBodyDiameter	nominalLength/9	Default for diameter of sphere representing the center of mass of a body [m]
defaultWidthFraction	20	Default for shape width as a fraction of shape length (e.g., for Parts.FixedTranslation)
defaultArrowDiameter	nominalLength/40	Default for arrow diameter (e.g., of forces, torques, sensors) [m]
defaultFrameDiameterFraction	40	Default for arrow diameter of a coordinate system as a fraction of axis length
defaultSpecularCoefficient	0.7	Default reflection of ambient light (= 0: light is completely absorbed)
defaultN_to_m	1000	Default scaling of force arrows (length = force/defaultN_to_m) [N/m]
defaultNm_to_m	1000	Default scaling of torque arrows (length = torque/defaultNm_to_m) [N.m/m]

## Connectors

Name	Description
frame_b	Coordinate system fixed in the origin of the world frame

## Modelica.Mechanics.MultiBody.Examples

Examples that demonstrate the usage of the MultiBody library

## Information

This package contains example models to demonstrate the usage of the MultiBody package. Open the models and simulate them according to the provided description in the models.

## Package Content

Name	Description
 Elementary	Elementary examples to demonstrate various features of the MultiBody library
 Loops	Examples with kinematic loops

 Systems	Examples of complete system models including 3-dimensional mechanics
---	--

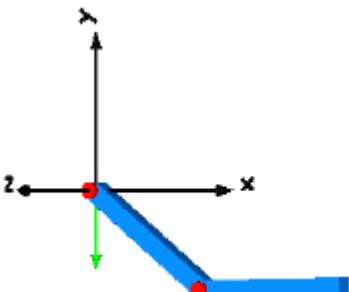
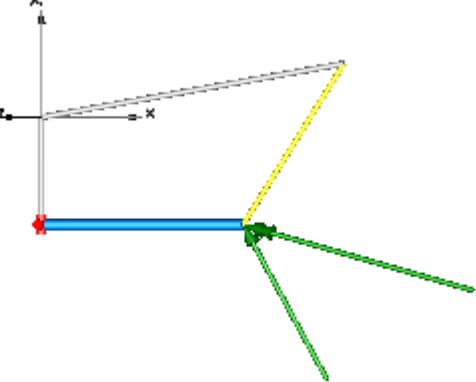
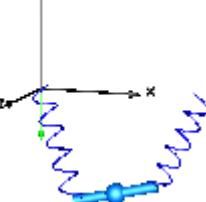
## Modelica.Mechanics.MultiBody.Examples.Elementary

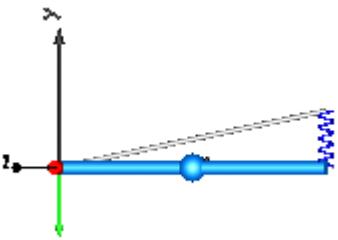
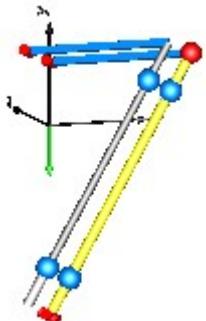
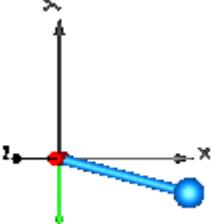
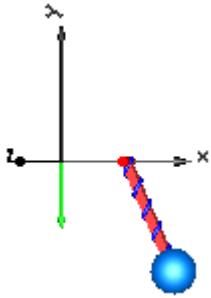
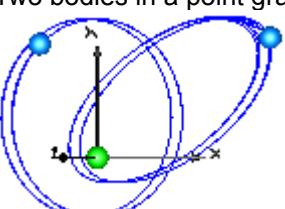
Elementary examples to demonstrate various features of the MultiBody library

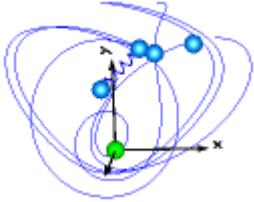
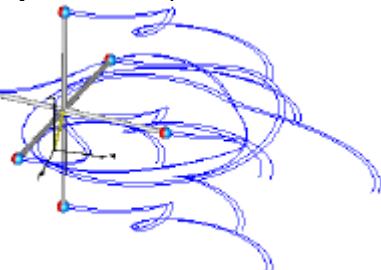
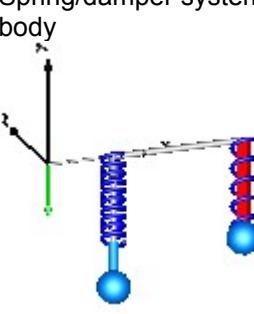
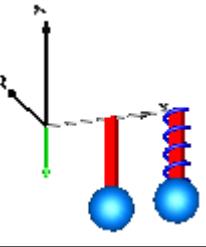
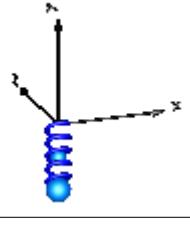
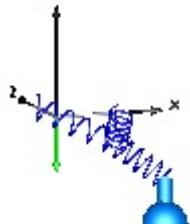
### Information

This package contains elementary example models to demonstrate the usage of the MultiBody library

### Content

<i>Model</i>	<i>Description</i>
<b>DoublePendulum</b>	Simple double pendulum with two revolute joints and two bodies. 
<b>ForceAndTorque</b>	Demonstrates usage of Forces.ForceAndTorque element. 
<b>FreeBody</b>	Free flying body attached by two springs to environment. 
<b>InitSpringConstant</b>	Determine spring constant such that system is in steady state at given position.

	
<b>LineForceWithTwoMasses</b>	Demonstrates a line force with two point masses using a Joints.Assemblies.JointUPS and alternatively a Forces.LineForceWithTwoMasses component. 
<b>Pendulum</b>	Simple pendulum with one revolute joint and one body. 
<b>PendulumWithSpringDamper</b>	Simple spring/damper/mass system 
<b>PointGravity</b>	Two bodies in a point gravity field 
<b>PointGravityWithPointMasses</b>	Two point masses in a point gravity field (rotation of bodies is neglected)

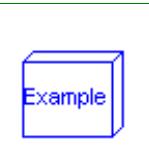
	
<b>PointGravityWithPointMasses2</b>	Rigidly connected point masses in a point gravity field 
<b>SpringDamperSystem</b>	Spring/damper system with a prismatic joint and attached on free flying body 
<b>SpringMassSystem</b>	Mass attached via a prismatic joint and a spring to the world frame 
<b>SpringWithMass</b>	Point mass hanging on a spring 
<b>ThreeSprings</b>	3-dimensional springs in series and parallel connection 

## Package Content

Name	Description
<a href="#">DoublePendulum</a>	Simple double pendulum with two revolute joints and two bodies
<a href="#">ForceAndTorque</a>	Demonstrate usage of ForceAndTorque element
<a href="#">FreeBody</a>	Free flying body attached by two springs to environment
<a href="#">InitSpringConstant</a>	Determine spring constant such that system is in steady state at given position
<a href="#">LineForceWithTwoMasses</a>	Demonstrate line force with two point masses using a JointUPS and alternatively a LineForceWithTwoMasses component
<a href="#">Pendulum</a>	Simple pendulum with one revolute joint and one body
<a href="#">PendulumWithSpringDamper</a>	Simple spring/damper/mass system
<a href="#">PointGravity</a>	Two point masses in a point gravity field
<a href="#">PointGravityWithPointMasses</a>	Two point masses in a point gravity field (rotation of bodies is neglected)
<a href="#">PointGravityWithPointMasses2</a>	Rigidly connected point masses in a point gravity field
<a href="#">SpringDamperSystem</a>	Simple spring/damper/mass system
<a href="#">SpringMassSystem</a>	Mass attached with a spring to the world frame
<a href="#">SpringWithMass</a>	Point mass hanging on a spring
<a href="#">ThreeSprings</a>	3-dim. springs in series and parallel connection

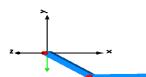
### Modelica.Mechanics.MultiBody.Examples.Elementary.DoublePendulum

Simple double pendulum with two revolute joints and two bodies



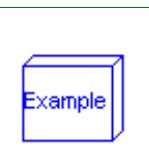
#### Information

This example demonstrates that by using joint and body elements animation is automatically available. Also the revolute joints are animated. Note, that animation of every component can be switched off by setting the first parameter **animation** to **false** or by setting **enableAnimation** in the **world** object to **false** to switch off animation of all components.



### Modelica.Mechanics.MultiBody.Examples.Elementary.ForceAndTorque

Demonstrate usage of ForceAndTorque element



#### Information

In this example the usage of the general force element "ForceAndTorque" is shown. A "ForceAndTorque" element is connected between a body and a fixed point in the world system. The force and torque is defined by the "Constant" block. The two vectors are resolved in the coordinate system defined by the "fixedRotation" component that is fixed in the world system:

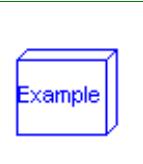
The animation view at time = 0 is shown in the figure below. The yellow line is directed from frame\_a to frame\_b of the forceAndTorque component. The green arrow characterizes the force acting at the body

whereas the green double arrow characterizes the torque acting at the body. The lengths of the two vectors are proportional to the lengths of the force and torque vectors (constant scaling factors are defined as parameters in the forceAndTorque component):



## Modelica.Mechanics.MultiBody.Examples.Elementary.FreeBody

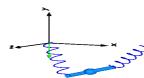
Free flying body attached by two springs to environment



### Information

This example demonstrates:

- The animation of spring and damper components
- A body can be freely moving without any connection to a joint. In this case body coordinates are used automatically as states (whenever joints are present, it is first tried to use the generalized coordinates of the joints as states).
- If a body is freely moving, the initial position and velocity of the body can be defined with the "Initialization" menu as shown with the body "body1" in the left part (click on "Initialization").

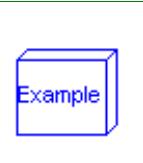


### Parameters

Name	Default	Description
animation	true	= true, if animation shall be enabled

## Modelica.Mechanics.MultiBody.Examples.Elementary.InitSpringConstant

Determine spring constant such that system is in steady state at given position



### Information

This example demonstrates a non-standard type of initialization by calculating a spring constant such that a simple pendulum is at a defined position in steady state.

The goal is that the pendulum should be in steady state when the rotation angle of the pendulum is zero. The spring constant of the spring shall be calculated during initialization such that this goal is reached.

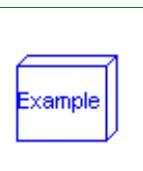
The pendulum has one degree of freedom, i.e., two states. Therefore, two additional equations have to be provided for initialization. However, parameter "c" of the spring component is defined with attribute "fixed = false", i.e., the value of this parameter is computed during initialization. Therefore, there is one additional equation required during initialization. The 3 initial equations are the rotational angle of the revolute joint and its first and second derivative. The latter one are zero, in order to initialize in steady state. By setting parameter initType of the revolute joint "rev" to "MultiBody.Types.Init.PositionVelocityAcceleration", the required 3 initial equations are defined.

After translation, this model is initialized in steady-state. The spring constant is computed as  $c = 49.05 \text{ N/m}$ . An animation of this simulation is shown in the figure below.



**Modelica.Mechanics.MultiBody.Examples.Elementary.LineForceWithTwoMasses**

Demonstrate line force with two point masses using a JointUPS and alternatively a LineForceWithTwoMasses component

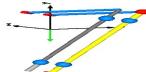
**Information**

It is demonstrated how to implement line force components that shall have mass properties. Two alternative implementations are given:

- With [JointUPS](#):  
Modelica.Mechanics.MultiBody.Joints.Assemblies.JointUPS is an aggregation of a universal, a prismatic and a spherical joint that approximates a real force component, such as a hydraulic cylinder. At the two frames of the prismatic joint (frame\_ia, frame\_ib of jointUPS) two bodies are attached. The parameters are selected such that the center of masses of the two bodies are located on the line connecting frame\_a and frame\_b of the jointUPS component. Both bodies have the same mass and the inertia tensor is set to zero, i.e., the two bodies are treated as point masses.
- With [LineForceWithTwoMasses](#):  
Modelica.Mechanics.MultiBody.Forces.LineForceWithTwoMasses is a line force component with the built-in property that two point masses are located on the line on which the line force is acting. The parameters are selected in such a way that the same system as with the jointUPS component is described.

In both cases, a linear 1-dimensional translational damper from the Modelica.Mechanics.Translational library is used as line force between the two attachment points. Simulate this system and plot the differences of the cut forces at both sides of the line force component ("rod\_f\_diff" and "body\_f\_diff"). Both vectors should be zero (depending on the chosen relative tolerance of the integration, the difference is in the order of 1.e-10 ... 1.e-15).

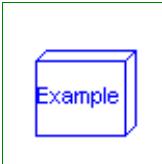
Note, that the implementation with the LineForceWithTwoMasses component is simpler and more convenient. An animation of this simulation is shown in the figure below. The system on the left side in the front is the animation with the LineForceWithTwoMasses component whereas the system on the right side in the back is the animation with the JointUPS component.

**Parameters**

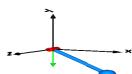
Name	Default	Description
m	1	Mass of point masses [kg]

**Modelica.Mechanics.MultiBody.Examples.Elementary.Pendulum**

Simple pendulum with one revolute joint and one body

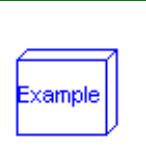
**Information**

This simple model demonstrates that by just dragging components default animation is defined that shows the structure of the assembled system.



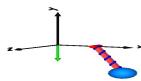
## **Modelica.Mechanics.MultiBody.Examples.Elementary.PendulumWithSpringDamper**

Simple spring/damper/mass system



### Information

A body is attached on a revolute and prismatic joint. A 3-dim. spring and a 3-dim. damper are connected between the body and a point fixed in the world frame:

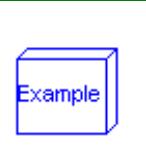


### Parameters

Name	Default	Description
animation	true	= true, if animation shall be enabled

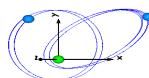
## **Modelica.Mechanics.MultiBody.Examples.Elementary.PointGravity**

Two point masses in a point gravity field



### Information

This model demonstrates a point gravity field. Two bodies are placed in the gravity field. The initial positions and velocities of these bodies are selected such that one body rotates on a circle and the other body rotates on an ellipse around the center of the point gravity field.



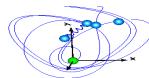
## **Modelica.Mechanics.MultiBody.Examples.Elementary.PointGravityWithPointMasses**

Two point masses in a point gravity field (rotation of bodies is neglected)



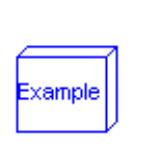
### Information

This model demonstrates the usage of model Parts.PointMass in a point gravity field. The PointMass model has the feature that rotation is not taken into account and can therefore also not be calculated. This example demonstrates two cases where this does not matter: If a PointMass is not connected (body1, body2), the orientation object in these point masses is set to a unit rotation. If a PointMass is connected by a line force element, such as the used Forces.LineForceWithMass component, then the orientation object is set to a unit rotation within the line force element. These are the two cases where the rotation is automatically set to a default value, when the physical system does not provide the equations.



## Modelica.Mechanics.MultiBody.Examples.Elementary.PointGravityWithPointMasses2

Rigidly connected point masses in a point gravity field

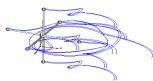


### Information

This model demonstrates the usage of model Parts.PointMass in a point gravity field. 6 point masses are connected rigidly together. Translating such a model results in an error, because point masses do not define an orientation object. The example demonstrates that in such a case (when the orientation object is not defined by an object that is connected to a point mass), a "MultiBody.Joints.FreeMotion" joint has to be used, to define the degrees of freedom of this structure.

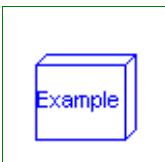
In order to demonstrate that this approach is correct, in model "referenceSystem", the same system is again provided, but this time modeled with a generic body (Parts.Body) where the inertia tensor is set to zero. In this case, no FreeMotion object is needed because every body provides its absolute translational and rotational position and velocity as potential states.

The two systems should move exactly in the same way. The system with the PointMasses object visualizes the point masses in "red", whereas the "referenceSystem" shows its bodies in "blue".



## Modelica.Mechanics.MultiBody.Examples.Elementary.SpringDamperSystem

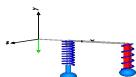
Simple spring/damper/mass system



### Information

This example demonstrates:

- The animation of spring and damper components
- A body can be freely moving without any connection to a joint. In this case body coordinates are used automatically as states (whenever joints are present, it is first tried to use the generalized coordinates of the joints as states).
- If a body is freely moving, the initial position and velocity of the body can be defined with the "Initialization" menu as shown with the body "body1" in the left part (click on "Initialization").

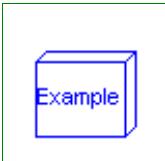


### Parameters

Name	Default	Description
animation	true	= true, if animation shall be enabled

## Modelica.Mechanics.MultiBody.Examples.Elementary.SpringMassSystem

Mass attached with a spring to the world frame



### Information

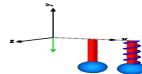
This example shows the two different ways how force laws can be utilized:

- In the left system a body is attached via a prismatic joint to the world frame. The prismatic joint has

two 1-dimensional translational flanges (called "bearing" and "axis") that allows to connect elements from the Modelica.Mechanics.Translational library between the bearing and the axis connector. The effect is that the force generated by the 1-dimensional elements acts as driving force in the axis of the prismatic joint. In the example a simple spring is used.

The advantage of this approach is that the many elements from the Translational library can be easily used here and that this implementation is usually more efficient as when using 3-dimensional springs.

- In the right system the same model is defined. The difference is that a 3-dimensional spring from the Modelica.Mechanics.MultiBody.Forces library is used. This has the advantage to get a nice animation of the force component.



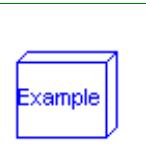
## Parameters

Name	Default	Description
animation	true	= true, if animation shall be enabled

---

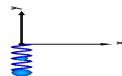
## Modelica.Mechanics.MultiBody.Examples.Elementary.SpringWithMass

Point mass hanging on a spring



## Information

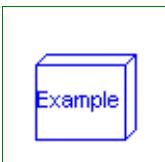
This example shows that a force component may have a mass. The 3-dimensional spring as used in this example, has an optional point mass between the two points where the spring is attached. In the animation, this point mass is represented by a small, light blue, sphere.




---

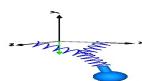
## Modelica.Mechanics.MultiBody.Examples.Elementary.ThreeSprings

3-dim. springs in series and parallel connection



## Information

This example demonstrates that **3-dimensional line force** elements (here: Modelica.Mechanics.MultiBody.Forces.Spring elements) can be connected together in **series** without having a body with mass at the connection point (as usually required by multi-body programs). This is advantageous since stiff systems can be avoided, say, due to a stiff spring and a small mass at the connection point.



## Parameters

Name	Default	Description
animation	true	= true, if animation shall be enabled

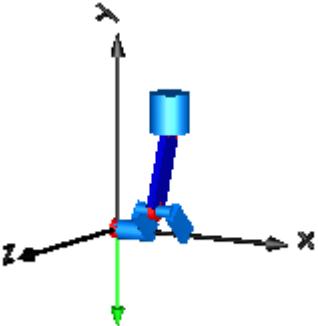
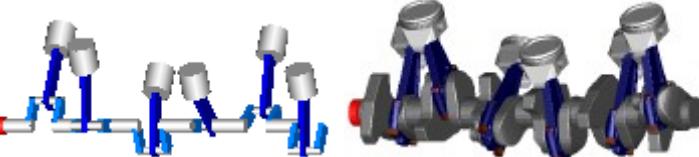
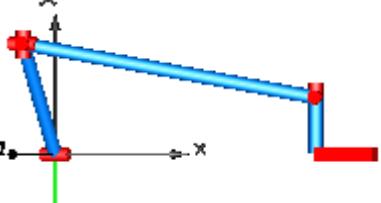
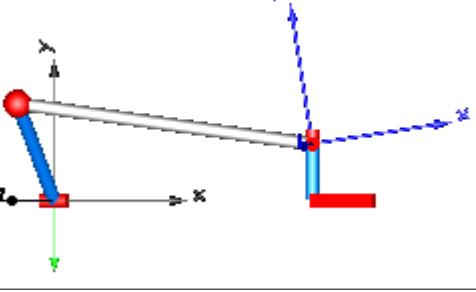
## Modelica.Mechanics.MultiBody.Examples.Loops

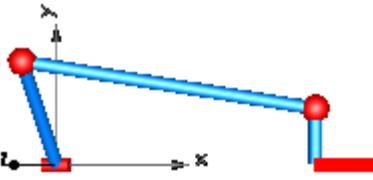
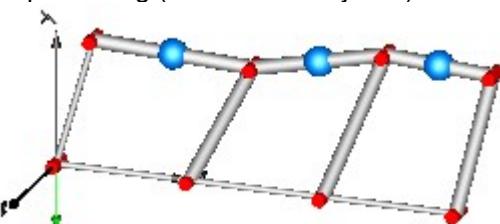
### Examples with kinematic loops

#### Information

This package contains different examples to show how mechanical systems with kinematic loops can be modeled.

#### Content

<i>Model</i>	<i>Description</i>
<b>Engine1a</b> <b>Engine1b</b> <b>Engine1b_analytic</b>	Model of one cylinder engine (Engine1a: simple, without combustion; Engine1b: with combustion; Engine1b_analytic: same as Engine1b but analytic loop handling)
	
<b>EngineV6</b> <b>EngineV6_analytic</b>	V6 engine with 6 cylinders, 6 planar loops and 1 degree-of-freedom. Second version with analytic handling of kinematic loops and CAD data animation.
	
<b>Fourbar1</b>	One kinematic loop with four bars (with only revolute joints; 5 non-linear equations)
	
<b>Fourbar2</b>	One kinematic loop with four bars (with UniversalSpherical joint; 1 non-linear equation)
	
<b>Fourbar_analytic</b>	One kinematic loop with four bars (with JointSSP joint; analytic solution of non-linear algebraic loop)

	
PlanarLoops_analytic	Mechanism with three planar kinematic loops and one degree-of-freedom with analytic loop handling (with JointRRR joints) 

## Package Content

Name	Description
<input type="checkbox"/> Engine1a	Model of one cylinder engine
<input type="checkbox"/> Engine1b	Model of one cylinder engine with gas force and preparation for assembly joint JointRRP
<input type="checkbox"/> Engine1b_analytic	Model of one cylinder engine with gas force and analytic loop handling
<input type="checkbox"/> EngineV6	V6 engine with 6 cylinders, 6 planar loops and 1 degree-of-freedom
<input type="checkbox"/> EngineV6_analytic	V6 engine with 6 cylinders, 6 planar loops, 1 degree-of-freedom and analytic handling of kinematic loops
<input type="checkbox"/> Fourbar1	One kinematic loop with four bars (with only revolute joints; 5 non-linear equations)
<input type="checkbox"/> Fourbar2	One kinematic loop with four bars (with UniversalSpherical joint; 1 non-linear equation)
<input type="checkbox"/> Fourbar_analytic	One kinematic loop with four bars (with JointSSP joint; analytic solution of non-linear algebraic loop)
<input type="checkbox"/> PlanarLoops_analytic	Mechanism with three planar kinematic loops and one degree-of-freedom with analytic loop handling (with JointRRR joints)
<input type="checkbox"/> Utilities	Utility models for Examples.Loops

## Modelica.Mechanics.MultiBody.Examples.Loops.Engine1a

### Model of one cylinder engine



#### Information

This is a model of the mechanical part of one cylinder of an engine. The combustion is not modelled. The "inertia" component at the lower left part is the output inertia of the engine driving the gearbox. The angular velocity of the output inertia has a start value of 10 rad/s in order to demonstrate the movement of the engine.

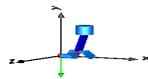
The engine is modeled solely by revolute and prismatic joints. Since this results in a **planar** loop there is the well known difficulty that the cut-forces perpendicular to the loop cannot be uniquely computed, as well as the cut-torques within the plane. This ambiguity is resolved by using the option **planarCutJoint** in the **Advanced** menu of one revolute joint in every planar loop (here: joint B1). This option sets the cut-force in direction of the axis of rotation, as well as the cut-torques perpendicular to the axis of rotation at this joint to

## 428 Modelica.Mechanics.MultiBody.Examples.Loops.Engine1a

---

zero and makes the problem mathematically well-formed.

An animation of this example is shown in the figure below.



---

## Modelica.Mechanics.MultiBody.Examples.Loops.Engine1b

**Model of one cylinder engine with gas force and preparation for assembly joint JointRRP**



### Information

This is a model of the mechanical part of one cylinder of an engine. It is similar to [Loops.Engine1a](#). The difference is that a simple model for the gas force in the cylinder is added and that the model is restructured in such a way, that the central part of the planar kinematic loop can be easily replaced by the assembly joint "Modelica.Mechanics.MultiBody.Joints.Assemblies.JointRRP". This exchange of the kinematic loop is shown in [Loops.Engine1b\\_analytic](#). The advantage of using JointRRP is, that the non-linear algebraic equation of this loop is solved analytically, and not numerically as in this model (Engine1b).

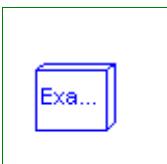
An animation of this example is shown in the figure below.



---

## Modelica.Mechanics.MultiBody.Examples.Loops.Engine1b\_analytic

**Model of one cylinder engine with gas force and analytic loop handling**



### Information

This is the same model as [Loops.Engine1b](#). The only difference is that the central part of the planar kinematic loop has been replaced by the assembly joint "Modelica.Mechanics.MultiBody.Joints.Assemblies.JointRRP". The advantage of using JointRRP is, that the non-linear algebraic equation of this loop is solved analytically, and not numerically as in [Loops.Engine1b](#).

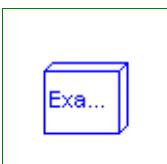
An animation of this example is shown in the figure below.



---

## Modelica.Mechanics.MultiBody.Examples.Loops.EngineV6

**V6 engine with 6 cylinders, 6 planar loops and 1 degree-of-freedom**



### Information

This is a V6 engine with 6 cylinders. It is hierarchically built up by using instances of one cylinder. For more details on the modeling of one cylinder, see example [Engine](#). An animation of the engine is shown in the figure below.



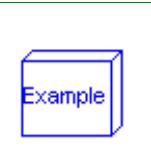
Simulate for 5 s, and plot the variables **engineSpeed\_rpm**, **engineTorque**, and **filteredEngineTorque**. Note, the result file has a size of about 50 Mbyte (for 5000 output intervals).

## Parameters

Name	Default	Description
animation	true	= true, if animation shall be enabled

## Modelica.Mechanics.MultiBody.Examples.Loops.EngineV6\_analytic

V6 engine with 6 cylinders, 6 planar loops, 1 degree-of-freedom and analytic handling of kinematic loops



## Information

This is a similar model as the example "EngineV6". However, the cylinders have been built up with component Modelica.Mechanics.MultiBody.Joints.Assemblies.JointRRR that solves the non-linear system of equations in an aggregation of 3 revolution joints **analytically** and only one body is used that holds the total mass of the crank shaft:



This model is about 20 times faster as the EngineV6 example and **no** linear or non-linear system of equations occur. In contrast, the "EngineV6" example leads to 6 systems of nonlinear equations (every system has dimension = 5, with Evaluate=false and dimension=1 with Evaluate=true) and a linear system of equations of about 40. This shows the power of the analytic loop handling.

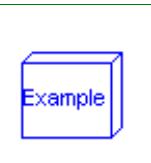
Simulate for 5 s, and plot the variables **engineSpeed\_rpm**, **engineTorque**, and **filteredEngineTorque**. Note, the result file has a size of about 50 Mbyte (for 5000 output intervals).

## Parameters

Name	Default	Description
animation	true	= true, if animation shall be enabled

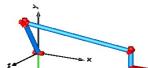
## Modelica.Mechanics.MultiBody.Examples.Loops.Fourbar1

One kinematic loop with four bars (with only revolute joints; 5 non-linear equations)



## Information

This is a simple kinematic loop consisting of 6 revolute joints, 1 prismatic joint and 4 bars that is often used as basic constructing unit in mechanisms. This example demonstrates that usually no particular knowledge of the user is needed to handle kinematic loops. Just connect the joints and bodies together according to the real system. In particular **no** cut-joints or a spanning tree has to be determined. In this case, the initial condition of the angular velocity of revolute joint j1 is set to 300 deg/s in order to drive this loop.

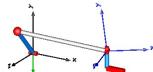


**Modelica.Mechanics.MultiBody.Examples.Loops.Fourbar2**

One kinematic loop with four bars (with UniversalSpherical joint; 1 non-linear equation)

**Information**

This is a second version of the "four-bar" mechanism, see figure:



In this case the three revolute joints on the left top-side and the two revolute joints on the right top side have been replaced by the joint **UniversalSpherical** that is a rod connecting a spherical and a universal joint. This joint is defined by **1 constraint** stating that the distance between the two spherical joints is constant. Using this joint in a kinematic loop reduces the sizes of non-linear algebraic equations. For this loop, only one non-linear algebraic system of equations of order 1 remains.

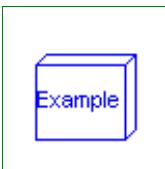
At the UniversalSpherical joint an additional frame\_ia fixed to the rod is present where components can be attached to the connecting rod. In this example just a coordinate system is attached to visualize frame\_ia (coordinate system on the right in blue color).

Another feature is that the length of the connecting rod can be automatically calculated during **initialization**. In order to do this, another initialization condition has to be given. In this example, the initial value of the distance of the prismatic joint j2 has been fixed (via the "Initialization" menu) and the rod length of joint "UniversalSpherical" is computed during initialization since parameter **computeLength = true** is set in the joint parameter menu. The main advantage is that during initialization no non-linear system of equation is solved and therefore initialization always works. To be precise, the following trivial non-linear equation is actually solved for rodLength:

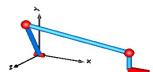
```
rodLength*rodLength = f(angle of revolute joint, distance of prismatic joint)
```

**Modelica.Mechanics.MultiBody.Examples.Loops.Fourbar\_analytic**

One kinematic loop with four bars (with JointSSP joint; analytic solution of non-linear algebraic loop)

**Information**

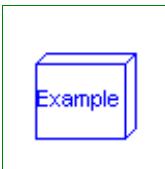
This is a third version of the "four-bar" mechanism, see figure:



In this case the three revolute joints on the left top-side and the two revolute joints on the right top side have been replaced by the assembly joint **Joints.Assemblies.JointSSP** which consists of two spherical joints and one prismatic joint. Since JointSSP solves the non-linear constraint equation internally analytically, no non-linear equation appears any more and a Modelica translator, such as Dymola, can transform the system into state space form without solving a system of equations. For more details, see [MultiBody.UsersGuide.Tutorial.LoopStructures.AnalyticLoopHandling](#).

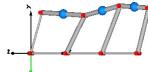
**Modelica.Mechanics.MultiBody.Examples.Loops.PlanarLoops\_analytic**

Mechanism with three planar kinematic loops and one degree-of-freedom with analytic loop handling (with JointRRR joints)



## Information

It is demonstrated how the Modelica.Mechanics.MultiBody.Joints.Assemblies.JointRRR joint can be used to solve the non-linear equations of coupled planar loops analytically. In the mechanism below no non-linear equation occurs any more from the tool view, since these equations are solved analytically in the JointRRR joints. For more details, see [MultiBody.UsersGuide.Tutorial.LoopStructures.AnalyticLoopHandling](#).



## Parameters

Name	Default	Description
rh[3]	{0.5,0,0}	Position vector from 'lower left' revolute to 'lower right' revolute joint for all the 3 loops [m]
rv[3]	{0,0.5,0}	Position vector from 'lower left' revolute to 'upper left' revolute joint [m]
r1b[3]	{0.1,0.5,0}	[m]
r1a[3]	r1b + rh - rv	[m]
r2b[3]	{0.1,0.6,0}	[m]
r2a[3]	r2b + rh - r1b	[m]
r3b[3]	{0,0.55,0}	[m]
r3a[3]	r3b + rh - r2b	[m]

## Modelica.Mechanics.MultiBody.Examples.Loops.Utilities

### Utility models for Examples.Loops

#### Package Content

Name	Description
Cylinder	
GasForce	
GasForce2	Rough approximation of gas force in a cylinder
CylinderBase	One cylinder with analytic handling of kinematic loop
Cylinder_analytic_CAD	
EngineV6_analytic	V6 engine with analytic loop handling
Engine1bBase	Model of one cylinder engine with gas force

## Modelica.Mechanics.MultiBody.Examples.Loops.Utilities.Cylinder



### Parameters

Name	Default	Description
animation	true	= true, if animation shall be enabled
cylinderTopPosition	0.42	Length from crank shaft to end of cylinder. [m]
pistonLength	0.1	Length of cylinder [m]

## 432 Modelica.Mechanics.MultiBody.Examples.Loops.Utilities.Cylinder

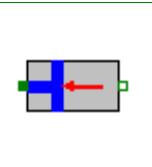
rodLength	0.2	Length of rod [m]
crankLength	0.2	Length of crank shaft in x direction [m]
crankPinOffset	0.1	Offset of crank pin from center axis [m]
crankPinLength	0.1	Offset of crank pin from center axis [m]
cylinderInclination	0	Inclination of cylinder [rad]
crankAngleOffset	0	Offset for crank angle [rad]
cylinderLength	cylinderTopPosition - (pisto...)	Maximum length of cylinder volume [m]

### Connectors

Name	Description
cylinder_a	
cylinder_b	
crank_a	
crank_b	

---

## Modelica.Mechanics.MultiBody.Examples.Loops.Utilities.GasForce



### Parameters

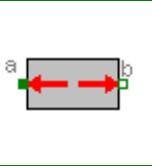
Name	Default	Description
L		Length of cylinder [m]
d		diameter of cylinder [m]
k0	0.01	
k1	1	
k	1	

### Connectors

Name	Description
flange_a	(left) driving flange (flange axis directed INTO cut plane, e. g. from left to right)
flange_b	(right) driven flange (flange axis directed OUT OF cut plane)

---

## Modelica.Mechanics.MultiBody.Examples.Loops.Utilities.GasForce2



Rough approximation of gas force in a cylinder

### Information

The gas force in a cylinder is computed as function of the relative distance of the two flanges. It is required that  $s_{\text{rel}} = \text{flange}_b.s - \text{flange}_a.s$  is in the range

$$0 \leq s_{\text{rel}} \leq L$$

where the parameter L is the length of the cylinder. If this assumption is not fulfilled, an error occurs.

### Parameters

Name	Default	Description

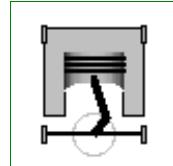
L		Length of cylinder [m]
d		diameter of cylinder [m]
k0	0.01	
k1	1	
k	1	

## Connectors

Name	Description
flange_a	(left) driving flange (flange axis directed INTO cut plane, e. g. from left to right)
flange_b	(right) driven flange (flange axis directed OUT OF cut plane)

## Modelica.Mechanics.MultiBody.Examples.Loops.Utilities.CylinderBase

One cylinder with analytic handling of kinematic loop



## Parameters

Name	Default	Description
animation	true	= true, if animation shall be enabled
cylinderTopPosition	0.42	Length from crank shaft to end of cylinder. [m]
crankLength	0.14	Length of crank shaft in x direction [m]
crankPinOffset	0.05	Offset of crank pin from center axis [m]
crankPinLength	0.1	Offset of crank pin from center axis [m]
cylinderInclination	0	Inclination of cylinder [deg]
crankAngleOffset	0	Offset for crank angle [deg]
<b>Piston</b>		
pistonLength	0.1	Length of cylinder [m]
pistonCenterOfMass	pistonLength/2	Distance from frame_a to center of mass of piston [m]
pistonMass	6	Mass of piston [kg]
pistonInertia_11	0.0088	Inertia 11 of piston with respect to center of mass frame, parallel to frame_a [kg.m2]
pistonInertia_22	0.0076	Inertia 22 of piston with respect to center of mass frame, parallel to frame_a [kg.m2]
pistonInertia_33	0.0088	Inertia 33 of piston with respect to center of mass frame, parallel to frame_a [kg.m2]
<b>Rod</b>		
rodLength	0.175	Length of rod [m]
rodCenterOfMass	rodLength/2	Distance from frame_a to center of mass of piston [m]
rodMass	1	Mass of rod [kg]
rodInertia_11	0.006	Inertia 11 of rod with respect to center of mass frame, parallel to frame_a [kg.m2]
rodInertia_22	0.0005	Inertia 22 of rod with respect to center of mass frame, parallel to frame_a [kg.m2]
rodInertia_33	0.006	Inertia 33 of rod with respect to center of mass frame, parallel to frame_a [kg.m2]

## Connectors

Name	Description
cylinder_a	
cylinder_b	
crank_a	
crank_b	

## Modelica.Mechanics.MultiBody.Examples.Loops.Utilities.Cylinder\_analytic\_CAD



## Parameters

Name	Default	Description
animation	true	= true, if animation shall be enabled
cylinderTopPosition	0.42	Length from crank shaft to end of cylinder. [m]
crankLength	0.14	Length of crank shaft in x direction [m]
crankPinOffset	0.05	Offset of crank pin from center axis [m]
crankPinLength	0.1	Offset of crank pin from center axis [m]
cylinderInclination	0	Inclination of cylinder [deg]
crankAngleOffset	0	Offset for crank angle [deg]
<b>Piston</b>		
pistonLength	0.1	Length of cylinder [m]
pistonCenterOfMass	pistonLength/2	Distance from frame_a to center of mass of piston [m]
pistonMass	6	Mass of piston [kg]
pistonInertia_11	0.0088	Inertia 11 of piston with respect to center of mass frame, parallel to frame_a [kg.m2]
pistonInertia_22	0.0076	Inertia 22 of piston with respect to center of mass frame, parallel to frame_a [kg.m2]
pistonInertia_33	0.0088	Inertia 33 of piston with respect to center of mass frame, parallel to frame_a [kg.m2]
<b>Rod</b>		
rodLength	0.175	Length of rod [m]
rodCenterOfMass	rodLength/2	Distance from frame_a to center of mass of piston [m]
rodMass	1	Mass of rod [kg]
rodInertia_11	0.006	Inertia 11 of rod with respect to center of mass frame, parallel to frame_a [kg.m2]
rodInertia_22	0.0005	Inertia 22 of rod with respect to center of mass frame, parallel to frame_a [kg.m2]
rodInertia_33	0.006	Inertia 33 of rod with respect to center of mass frame, parallel to frame_a [kg.m2]

## Connectors

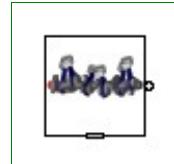
Name	Description
cylinder_a	
cylinder_b	
crank_a	

crank_b	
---------	--

---

## Modelica.Mechanics.MultiBody.Examples.Loops.Utilities.EngineV6\_analytic

V6 engine with analytic loop handling



### Parameters

Name	Default	Description
animation	true	= true, if animation shall be enabled

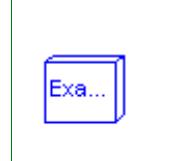
### Connectors

Name	Description
flange_b	
frame_a	

---

## Modelica.Mechanics.MultiBody.Examples.Loops.Utilities.Engine1bBase

Model of one cylinder engine with gas force



### Information

This is a model of the mechanical part of one cylinder of an engine. The combustion is not modelled. The "inertia" component at the lower left part is the output inertia of the engine driving the gearbox. The angular velocity of the output inertia has a start value of 10 rad/s in order to demonstrate the movement of the engine.

The engine is modeled solely by revolute and prismatic joints. Since this results in a **planar** loop there is the well known difficulty that the cut-forces perpendicular to the loop cannot be uniquely computed, as well as the cut-torques within the plane. This ambiguity is resolved by using the option **planarCutJoint** in the **Advanced** menu of one revolute joint in every planar loop (here: joint B1). This option sets the cut-force in direction of the axis of rotation, as well as the cut-torques perpendicular to the axis of rotation at this joint to zero and makes the problem mathematically well-formed.

An animation of this example is shown in the figure below.




---

## Modelica.Mechanics.MultiBody.Examples.Systems

Examples of complete system models including 3-dimensional mechanics

### Information

This package contains complete **system models** where components from different domains are used, including 3-dimensional mechanics.

### Content

Model	Description

<b>RobotR3</b> <b>RobotR3.oneAxis</b> <b>RobotR3.fullRobot</b>	<p>6 degree of freedom robot with path planning, controllers, motors, brakes, gears and mechanics. "oneAxis" models only one drive train. "fullRobot" is the complete, detailed robot model.</p> 
--	--

## Package Content

Name	Description
 <b>RobotR3</b>	Library to demonstrate robot system models based on the Manutec r3 robot

---

## Modelica.Mechanics.MultiBody.Examples.Systems.RobotR3

**Library to demonstrate robot system models based on the Manutec r3 robot**

### Information

This package contains models of the robot r3 of the company Manutec. These models are used to demonstrate in which way complex robot models might be built up by testing first the component models individually before composing them together. Furthermore, it is shown how CAD data can be used for animation.



The following models are available:

**oneAxis** Test one axis (controller, motor, gearbox).  
**fullRobot** Test complete robot model.

The r3 robot is no longer manufactured. In fact the company Manutec does no longer exist. The parameters of this robot have been determined by measurements in the laboratory of DLR. The measurement procedure is described in:

Tuerk S. (1990): Zur Modellierung der Dynamik von Robotern mit rotatorischen Gelenken. Fortschrittberichte VDI, Reihe 8, Nr. 211, VDI-Verlag 1990.

The robot model is described in detail in

Otter M. (1995): Objektorientierte Modellierung mechatronischer Systeme am Beispiel geregelter Roboter. Dissertation, Fortschrittberichte VDI, Reihe 20, Nr. 147, VDI-Verlag 1995.  
 This report can be downloaded as compressed postscript file from: <http://www.robotic.dlr.de/Martin.Otter/publications.html>.

The path planning is performed in a simple way by using essentially the Modelica.Mechanics.Rotational.KinematicPTP block. A user defines a path by start and end angle of every axis. A path is planned such that all axes are moving as fast as possible under the given restrictions of maximum joint speeds and maximum joint accelerations. The actual r3 robot from Manutec had a different

path planning strategy. Todays path planning algorithms from robot companies are much more involved.

In order to get a nice animation, CAD data from a KUKA robot is used, since CAD data of the original r3 robot was not available. The KUKA CAD data was derived from public data of KUKA available at:

[http://www.kuka-roboter.de/english/produkte/cad/low\\_payloads.html](http://www.kuka-roboter.de/english/produkte/cad/low_payloads.html). Since dimensions of the corresponding KUKA robot are similar but not identical to the r3 robot, the data of the r3 robot (such as arm lengths) have been modified, such that it matches the CAD data.

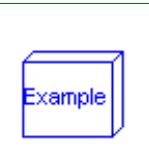
In this model, a simplified P-PI cascade controller for every axes is used. The parameters have been manually adjusted by simulations. The original r3 controllers are more complicated. The reason to use simplified controllers is to have a simpler demo.

## Package Content

Name	Description
 oneAxis	Model of one axis of robot (controller, motor, gearbox) with simple load
 fullRobot	6 degree of freedom robot with path planning, controllers, motors, brakes, gears and mechanics
 Components	Library of components of the robot

### Modelica.Mechanics.MultiBody.Examples.Systems.RobotR3.oneAxis

Model of one axis of robot (controller, motor, gearbox) with simple load



#### Information

With this model one axis of the r3 robot is checked. The mechanical structure is replaced by a simple load inertia.

#### Parameters

Name	Default	Description
mLoad	15	mass of load [kg]
kp	5	gain of position controller of axis 2
ks	0.5	gain of speed controller of axis 2
Ts	0.05	time constant of integrator of speed controller of axis 2 [s]
startAngle	0	start angle of axis 2 [deg]
endAngle	120	end angle of axis 2 [deg]
swingTime	0.5	additional time after reference motion is in rest before simulation is stopped [s]
refSpeedMax	3	maximum reference speed [rad/s]
refAccMax	10	maximum reference acceleration [rad/s <sup>2</sup> ]

### Modelica.Mechanics.MultiBody.Examples.Systems.RobotR3.fullRobot

6 degree of freedom robot with path planning, controllers, motors, brakes, gears and mechanics



#### Information

This is a detailed model of the robot. For animation CAD data is used. Translate and simulate with the default settings (default simulation time = 3 s). Use command script "Scripts\Examples\fullRobotPlot.mos" to plot variables.



## Parameters

Name	Default	Description
mLoad	15	mass of load [kg]
rLoad[3]	{0.1,0.25,0.1}	distance from last flange to load mass [m]
g	9.81	gravity acceleration [m/s <sup>2</sup> ]
refStartTime	0	start time of reference motion [s]
refSwingTime	0.7	additional time after reference motion is in rest before simulation is stopped [s]
<b>Reference</b>		
startAngles		
startAngle1	-60	start angle of axis 1 [deg]
startAngle2	20	start angle of axis 2 [deg]
startAngle3	90	start angle of axis 3 [deg]
startAngle4	0	start angle of axis 4 [deg]
startAngle5	-110	start angle of axis 5 [deg]
startAngle6	0	start angle of axis 6 [deg]
endAngles		
endAngle1	60	end angle of axis 1 [deg]
endAngle2	-70	end angle of axis 2 [deg]
endAngle3	-35	end angle of axis 3 [deg]
endAngle4	45	end angle of axis 4 [deg]
endAngle5	110	end angle of axis 5 [deg]
endAngle6	45	end angle of axis 6 [deg]
<b>Limits</b>		
refSpeedMax[6]	{3,1.5,5,3.1,3.1,4.1}	Maximum reference speeds of all joints [rad/s]
refAccMax[6]	{15,15,15,60,60,60}	Maximum reference accelerations of all joints [rad/s <sup>2</sup> ]
<b>Controller</b>		
Axis 1		
kp1	5	gain of position controller
ks1	0.5	gain of speed controller
Ts1	0.05	time constant of integrator of speed controller [s]
Axis 2		
kp2	5	gain of position controller
ks2	0.5	gain of speed controller
Ts2	0.05	time constant of integrator of speed controller [s]
Axis 3		
kp3	5	gain of position controller
ks3	0.5	gain of speed controller
Ts3	0.05	time constant of integrator of speed controller [s]
Axis 4		
kp4	5	gain of position controller
ks4	0.5	gain of speed controller

Ts4	0.05	time constant of integrator of speed controller [s]
<b>Axis 5</b>		
kp5	5	gain of position controller
ks5	0.5	gain of speed controller
Ts5	0.05	time constant of integrator of speed controller [s]
<b>Axis 6</b>		
kp6	5	gain of position controller
ks6	0.5	gain of speed controller
Ts6	0.05	time constant of integrator of speed controller [s]

## Modelica.Mechanics.MultiBody.Examples.Systems.RobotR3.Components

### Library of components of the robot

#### Information

This library contains the different components of the r3 robot. Usually, there is no need to use this library directly.

#### Package Content

Name	Description
 AxisControlBus	Data bus for one robot axis
 ControlBus	Data bus for all axes of robot
 PathPlanning1	Generate reference angles for fastest kinematic movement
 PathPlanning6	Generate reference angles for fastest kinematic movement
 PathToAxisControlBus	Map path planning to one axis control bus
 GearType1	Motor inertia and gearbox model for r3 joints 1,2,3
 GearType2	Motor inertia and gearbox model for r3 joints 4,5,6
 Motor	Motor model including current controller of r3 motors
 Controller	P-PI cascade controller for one axis
 AxisType1	Axis model of the r3 joints 1,2,3
 AxisType2	Axis model of the r3 joints 4,5,6
 MechanicalStructure	Model of the mechanical part of the r3 robot (without animation)

## Modelica.Mechanics.MultiBody.Examples.Systems.RobotR3.Components.AxisControlIBus

### Data bus for one robot axis

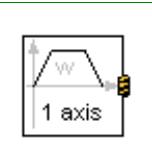


#### Information

Signal bus that is used to communicate all signals for **one** axis. This is an expandable connector which is "empty". The actual signal content is defined by connecting to an instance of this connector.

**Modelica.Mechanics.MultiBody.Examples.Systems.RobotR3.Components.ControlBus****Data bus for all axes of robot****Information**

Signal bus that is used to communicate **all signals** of the robot. This is an expandable connector which is "empty". The actual signal content is defined by connecting to an instance of this connector. It consists of one instance of the "AxisControlBus" for every axis.

**Modelica.Mechanics.MultiBody.Examples.Systems.RobotR3.Components.PathPlannng1****Generate reference angles for fastest kinematic movement****Information**

Given

- start and end angle of an axis
- maximum speed of the axis
- maximum acceleration of the axis

this component computes the fastest movement under the given constraints. This means, that:

1. The axis accelerates with the maximum acceleration until the maximum speed is reached.
2. Drives with the maximum speed as long as possible.
3. Decelerates with the negative of the maximum acceleration until rest.

The acceleration, constant velocity and deceleration phase are determined in such a way that the movement starts from the start angles and ends at the end angles. The output of this block are the computed angles, angular velocities and angular acceleration and this information is stored as reference motion on the controlBus of the r3 robot.

**Parameters**

Name	Default	Description
angleBegDeg	0	start angle [deg]
angleEndDeg	1	end angle [deg]
speedMax	3	maximum axis speed [rad/s]
accMax	2.5	maximum axis acceleration [rad/s <sup>2</sup> ]
startTime	0	start time of movement [s]
swingTime	0.5	additional time after reference motion is in rest before simulation is stopped [s]

**Connectors**

Name	Description
controlBus	

**Modelica.Mechanics.MultiBody.Examples.Systems.RobotR3.Components.PathPlannng6****Generate reference angles for fastest kinematic movement**

## Information

Given

- start and end angles of every axis
- maximum speed of every axis
- maximum acceleration of every axis

this component computes the fastest movement under the given constraints. This means, that:

1. Every axis accelerates with the maximum acceleration until the maximum speed is reached.
2. Drives with the maximum speed as long as possible.
3. Decelerates with the negative of the maximum acceleration until rest.

The acceleration, constant velocity and deceleration phase are determined in such a way that the movement starts from the start angles and ends at the end angles. The output of this block are the computed angles, angular velocities and angular acceleration and this information is stored as reference motion on the controlBus of the r3 robot.

## Parameters

Name	Default	Description
naxis	6	number of driven axis
angleBegDeg[naxis]	zeros(naxis)	start angles [deg]
angleEndDeg[naxis]	ones(naxis)	end angles [deg]
speedMax[naxis]	fill(3, naxis)	maximum axis speed [rad/s]
accMax[naxis]	fill(2.5, naxis)	maximum axis acceleration [rad/s <sup>2</sup> ]
startTime	0	start time of movement [s]
swingTime	0.5	additional time after reference motion is in rest before simulation is stopped [s]

## Connectors

Name	Description
controlBus	

## Modelica.Mechanics.MultiBody.Examples.Systems.RobotR3.Components.PathToAxis ControlBus

Map path planning to one axis control bus



## Parameters

Name	Default	Description
nAxis	6	Number of driven axis
axisUsed	1	Map path planning of axisUsed to axisControlBus

## Connectors

Name	Description
q[nAxis]	
qd[nAxis]	

qdd[nAxis]	
axisControlBus	
moving[nAxis]	

**Modelica.Mechanics.MultiBody.Examples.Systems.RobotR3.Components.GearType1**

Motor inertia and gearbox model for r3 joints 1,2,3

**Information**

Models the gearbox used in the first three joints with all its effects, like elasticity and friction. Coulomb friction is approximated by a friction element acting at the "motor"-side. In reality, bearing friction should be also incorporated at the driven side of the gearbox. However, this would require considerable more effort for the measurement of the friction parameters. Default values for all parameters are given for joint 1. Model relativeStates is used to define the relative angle and relative angular velocity across the spring (=gear elasticity) as state variables. The reason is, that a default initial value of zero of these states makes always sense. If the absolute angle and the absolute angular velocity of model Jmotor would be used as states, and the load angle (= joint angle of robot) is NOT zero, one has always to ensure that the initial values of the motor angle and of the joint angle are modified correspondingly. Otherwise, the spring has an unrealistic deflection at initial time. Since relative quantities are used as state variables, this simplifies the definition of initial values considerably.

**Parameters**

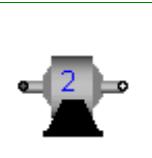
Name	Default	Description
i	-105	gear ratio
c	43	spring constant [N.m/rad]
d	0.005	damper constant [N.m.s/rad]
Rv0	0.4	viscous friction torque at zero velocity [N.m]
Rv1	(0.13/160)	viscous friction coefficient ( $R=Rv0+Rv1*abs(qd)$ ) [N.m.s/rad]
peak	1	peak*Rv0 = maximum static friction torque (peak >= 1)

**Connectors**

Name	Description
flange_a	
flange_b	

**Modelica.Mechanics.MultiBody.Examples.Systems.RobotR3.Components.GearType2**

Motor inertia and gearbox model for r3 joints 4,5,6

**Information**

The elasticity and damping in the gearboxes of the outermost three joints of the robot is neglected. Default values for all parameters are given for joint 4.

**Parameters**

Name	Default	Description
i	-99	gear ratio

Rv0	21.8	viscous friction torque at zero velocity [N.m]
Rv1	9.8	viscous friction coefficient in [Nms/rad] ( $R=Rv0+Rv1*abs(qd)$ )
peak	(26.7/21.8)	peak*Rv0 = maximum static friction torque (peak $\geq 1$ )

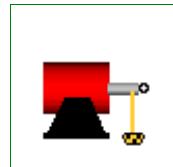
## Connectors

Name	Description
flange_a	
flange_b	

---

## **Modelica.Mechanics.MultiBody.Examples.Systems.RobotR3.Components.Motor**

**Motor model including current controller of r3 motors**



## Information

Default values are given for the motor of joint 1. The input of the motor is the desired current (the actual current is proportional to the torque produced by the motor).

## Parameters

Name	Default	Description
J	0.0013	moment of inertia of motor [kg.m <sup>2</sup> ]
k	1.1616	gain of motor
w	4590	time constant of motor
D	0.6	damping constant of motor
w_max	315	maximum speed of motor [rad/s]
i_max	9	maximum current of motor [A]

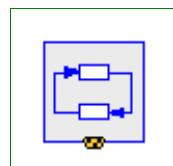
## Connectors

Name	Description
flange_motor	
axisControlBus	

---

## **Modelica.Mechanics.MultiBody.Examples.Systems.RobotR3.Components.Controller**

**P-PI cascade controller for one axis**



## Information

This controller has an inner PI-controller to control the motor speed, and an outer P-controller to control the motor position of one axis. The reference signals are with respect to the gear-output, and the gear ratio is used in the controller to determine the motor reference signals. All signals are communicated via the "axisControlBus".

## Parameters

Name	Default	Description
kp	10	gain of position controller
ks	1	gain of speed controller

Ts	0.01	time constant of integrator of speed controller [s]
ratio	1	gear ratio of gearbox

## Connectors

Name	Description
axisControlBus	

## Modelica.Mechanics.MultiBody.Examples.Systems.RobotR3.Components.AxisType1

Axis model of the r3 joints 1,2,3



## Parameters

Name	Default	Description
<b>Controller</b>		
kp	10	gain of position controller
ks	1	gain of speed controller
Ts	0.01	time constant of integrator of speed controller [s]
<b>Motor</b>		
k	1.1616	gain of motor
w	4590	time constant of motor
D	0.6	damping constant of motor
J	0.0013	moment of inertia of motor [kg.m <sup>2</sup> ]
<b>Gear</b>		
ratio	-105	gear ratio
Rv0	0.4	viscous friction torque at zero velocity in [Nm] [N.m]
Rv1	(0.13/160)	viscous friction coefficient in [Nms/rad] [N.m.s/rad]
peak	1	peak*Rv0 = maximum static friction torque (peak >= 1)
c	43	spring constant [N.m/rad]
cd	0.005	damper constant [N.m.s/rad]

## Connectors

Name	Description
flange	
axisControlBus	

## Modelica.Mechanics.MultiBody.Examples.Systems.RobotR3.Components.AxisType2

Axis model of the r3 joints 4,5,6



## Information

The axis model consists of the **controller**, the **motor** including current controller and the **gearbox** including gear elasticity and bearing friction. The only difference to the axis model of joints 4,5,6 (= model axisType2) is that elasticity and damping in the gear boxes are not neglected.

The input signals of this component are the desired angle and desired angular velocity of the joint. The reference signals have to be "smooth" (position has to be differentiable at least 2 times). Otherwise, the gear

elasticity leads to significant oscillations.

Default values of the parameters are given for the axis of joint 1.

## Parameters

Name	Default	Description
gear	redeclare GearType2 gear(Rv0...)	
Controller		
kp	10	gain of position controller
ks	1	gain of speed controller
Ts	0.01	time constant of integrator of speed controller [s]
Motor		
k	1.1616	gain of motor
w	4590	time constant of motor
D	0.6	damping constant of motor
J	0.0013	moment of inertia of motor [kg.m <sup>2</sup> ]
Gear		
ratio	-105	gear ratio
Rv0	0.4	viscous friction torque at zero velocity in [Nm] [N.m]
Rv1	(0.13/160)	viscous friction coefficient in [Nms/rad] [N.m.s/rad]
peak	1	peak*Rv0 = maximum static friction torque (peak >= 1)

## Connectors

Name	Description
flange	
axisControlBus	

---

## Modelica.Mechanics.MultiBody.Examples.Systems.RobotR3.Components.MechanicalStructure

Model of the mechanical part of the r3 robot (without animation)



## Information

This model contains the mechanical components of the r3 robot (multibody system).

## Parameters

Name	Default	Description
animation	true	= true, if animation shall be enabled
mLoad	15	mass of load [kg]
rLoad[3]	{0,0.25,0}	distance from last flange to load mass> [m]
g	9.81	gravity acceleration [m/s <sup>2</sup> ]

## Connectors

Name	Description
axis1	

axis2	
axis3	
axis4	
axis5	
axis6	

## Modelica.Mechanics.MultiBody.Force

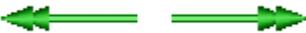
Components that exert forces and/or torques between frames

### Information

This package contains components that exert forces and torques between two frame connectors, e.g., between two parts.

### Content

Model	Description
<b>WorldForce</b>	External force acting at the frame to which this component is connected and defined by 3 input signals, that are interpreted as one vector resolved in the world frame. 
<b>WorldTorque</b>	External torque acting at the frame to which this component is connected and defined by 3 input signals, that are interpreted as one vector resolved in the world frame. 
<b>WorldForceAndTorque</b>	External force and external torque acting at the frame to which this component is connected and defined by 6 input signals, that are interpreted as a force and as a torque vector resolved in the world frame.  
<b>FrameForce</b>	External force acting at the frame to which this component is connected and defined by 3 input signals, that are interpreted as one vector resolved in the local frame or in "frame_resolve", if connected. 
<b>FrameTorque</b>	External torque acting at the frame to which this component is connected and defined by 3 input signals, that are interpreted as one vector resolved in the local frame or in "frame_resolve", if connected. 
<b>FrameForceAndTorque</b>	External force and torque acting at the frame to which this component is connected and defined by 6 input signals, that are interpreted as one force and one torque vector resolved in the local frame or in "frame_resolve", if connected.  
<b>Force</b>	Force acting between two frames defined by 3 input signals resolved in the local frame or in "frame_resolve", if connected.  

<b>Torque</b>	Torque acting between two frames defined by 3 input signals resolved in the local frame or in "frame_resolve", if connected. 
<b>ForceAndTorque</b>	Force and torque acting between two frames defined by 6 input signals resolved in the local frame or in "frame_resolve", if connected. 
<b>LineForceWithMass</b>	General line force component with an optional point mass on the connection line. The force law can be defined by a component of Modelica.Mechanics.Translational 
<b>LineForceWithTwoMasses</b>	General line force component with two optional point masses on the connection line. The force law can be defined by a component of Modelica.Mechanics.Translational 
<b>Spring</b>	Linear translational spring with optional mass 
<b>Damper</b>	Linear (velocity dependent) damper 
<b>SpringDamperParallel</b>	Linear spring and damper in parallel connection
<b>SpringDamperSeries</b>	Linear spring and damper in series connection

## Package Content

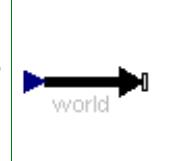
Name	Description
 <b>WorldForce</b>	External force acting at frame_b, defined by 3 input signals and resolved in world frame
 <b>WorldTorque</b>	External torque acting at frame_b, defined by 3 input signals and resolved in world frame
 <b>WorldForceAndTorque</b>	External force and torque acting at frame_b, defined by 6 input signals and resolved in world frame
 <b>FrameForce</b>	External force acting at frame_b, defined by 3 input signals and resolved in frame_b or in frame_resolve
 <b>FrameTorque</b>	External torque acting at frame_b, defined by 3 input signals and resolved in frame_b or in frame_resolve
 <b>FrameForceAndTorque</b>	External force and torque acting at frame_b, defined by 6 input signals and resolved in frame_b or in frame_resolve
 <b>Force</b>	Force acting between two frames, defined by 3 input signals and resolved in frame_b or in frame_resolve
 <b>Torque</b>	Torque acting between two frames, defined by 3 input signals and resolved in frame_b or in frame_resolve
 <b>ForceAndTorque</b>	Force and torque acting between two frames, defined by 6 input signals and resolved in frame_b or in frame_resolve
 <b>LineForceWithMass</b>	General line force component with an optional point mass on the connection line
 <b>LineForceWithTwoMasses</b>	General line force component with two optional point masses on the connection line

## 448 Modelica.Mechanics.MultiBody.Forces

 Spring	Linear translational spring with optional mass
 Damper	Linear (velocity dependent) damper
 SpringDamperParallel	Linear spring and linear damper in parallel
 SpringDamperSeries	Linear spring and linear damper in series connection

## Modelica.Mechanics.MultiBody.Forces.WorldForce

External force acting at frame\_b, defined by 3 input signals and resolved in world frame

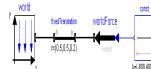


### Information

The 3 signals of the **force** connector are interpreted as the x-, y- and z-coordinates of a **force** resolved in the **world frame** and acting at the frame connector to which this component is attached.

This force component is by default visualized as an arrow acting at the connector to which it is connected. The diameter and color of the arrow are fixed and can be defined via parameters **diameter** and **color**. The arrow points in the direction defined by the inPort.signal signals. The length of the arrow is proportional to the length of the force vector using parameter **N\_to\_m** as scaling factor. For example, if **N\_to\_m** = 100 N/m, then a force of 350 N is displayed as an arrow of length 3.5 m.

An example how to use this model is given in the following figure:



This leads to the following animation



### Parameters

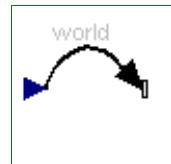
Name	Default	Description
animation	true	= true, if animation shall be enabled
if animation = true		
N_to_m	world.defaultN_to_m	Force arrow scaling (length = force/N_to_m) [N/m]
diameter	world.defaultArrowDiameter	Diameter of force arrow [m]
color	Modelica.Mechanics.MultiBody...	Color of arrow
specularCoefficient	world.defaultSpecularCoeffic...	Reflection of ambient light (= 0: light is completely absorbed)

### Connectors

Name	Description
frame_b	Coordinate system fixed to the component with one cut-force and cut-torque
force[3]	x-, y-, z-coordinates of force resolved in world frame

## Modelica.Mechanics.MultiBody.Forces.WorldTorque

External torque acting at frame\_b, defined by 3 input signals and resolved in world frame

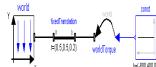


## Information

The **3** signals of the **torque** connector are interpreted as the x-, y- and z-coordinates of a **torque** resolved in the **world frame** and acting at the frame connector to which this component is attached.

This torque component is by default visualized as a **double arrow** acting at the connector to which it is connected. The diameter and color of the arrow are fixed and can be defined via parameters **diameter** and **color**. The double arrow points in the direction defined by the inPort.signal signals. The length of the double arrow is proportional to the length of the torque vector using parameter **Nm\_to\_m** as scaling factor. For example, if **Nm\_to\_m** = 100 Nm/m, then a torque of 350 Nm is displayed as an arrow of length 3.5 m.

An example how to use this model is given in the following figure:



This leads to the following animation



## Parameters

Name	Default	Description
animation	true	= true, if animation shall be enabled
if animation = true		
Nm_to_m	world.defaultNm_to_m	Torque arrow scaling (length = torque/Nm_to_m) [N.m/m]
diameter	world.defaultArrowDiameter	Diameter of torque arrow [m]
color	Modelica.Mechanics.MultiBody...	Color of arrow
specularCoefficient	world.defaultSpecularCoeffic...	Reflection of ambient light (= 0: light is completely absorbed)

## Connectors

Name	Description
frame_b	Coordinate system fixed to the component with one cut-force and cut-torque
torque[3]	x-, y-, z-coordinates of torque resolved in world frame

## Modelica.Mechanics.MultiBody.Forces.WorldForceAndTorque

External force and torque acting at frame\_b, defined by 6 input signals and resolved in world frame



## Information

The **6** signals of the **load** connector are interpreted as the x-, y- and z-coordinates of a **force** and as the x-, y-, and z-coordinates of a **torque** resolved in the **world frame** and acting at the frame connector to which this component is attached. The input signals are mapped to the force and torque in the following way:

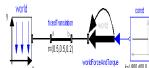
```
force  = load[1:3]
torque = load[4:6]
```

The force and torque are by default visualized as an arrow (force) and as a double arrow (torque) acting at the connector to which they are connected. The diameters and colors of the arrows are fixed and can be defined via parameters **forceDiameter**, **torqueDiameter**, **forceColor** and **torqueColor**. The arrows point in

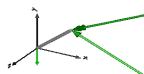
## 450 Modelica.Mechanics.MultiBody.Forces.WorldForceAndTorque

the directions defined by the inPort.signal signals. The lengths of the arrows are proportional to the length of the force and torque vectors, respectively, using parameters **N\_to\_m** and **Nm\_to\_m** as scaling factors. For example, if  $N_{to\_m} = 100 \text{ N/m}$ , then a force of 350 N is displayed as an arrow of length 3.5 m.

An example how to use this model is given in the following figure:



This leads to the following animation



### Parameters

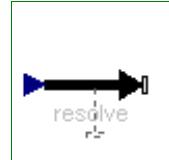
Name	Default	Description
animation	true	= true, if animation shall be enabled
if animation = true		
N_to_m	world.defaultN_to_m	Force arrow scaling (length = force/N_to_m) [N/m]
Nm_to_m	world.defaultNm_to_m	Torque arrow scaling (length = torque/Nm_to_m) [N.m/m]
forceDiameter	world.defaultArrowDiameter	Diameter of force arrow [m]
torqueDiameter	forceDiameter	Diameter of torque arrow [m]
forceColor	Modelica.Mechanics.MultiBody...	Color of force arrow
torqueColor	Modelica.Mechanics.MultiBody...	Color of torque arrow
specularCoefficient	world.defaultSpecularCoeffic...	Reflection of ambient light (= 0: light is completely absorbed)

### Connectors

Name	Description
frame_b	Coordinate system fixed to the component with one cut-force and cut-torque
load[6]	[1:6] = x-, y-, z-coordinates of force and x-, y-, z-coordinates of torque resolved in world frame

## Modelica.Mechanics.MultiBody.Forces.FrameForce

External force acting at **frame\_b**, defined by 3 input signals and resolved in **frame\_b** or in **frame\_resolve**

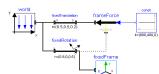


### Information

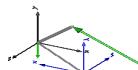
The 3 signals of the **force** connector are interpreted as the x-, y- and z-coordinates of a **force** acting at the frame connector to which this component is attached. If connector **frame\_resolve** is **not** connected, the force coordinates are with respect to **frame\_b**. If connector **frame\_resolve** is connected, the force coordinates are with respect to **frame\_resolve**. In this case the force and torque in connector **frame\_resolve** are set to zero, i.e., this connector is solely used to provide the information of the coordinate system, in which the force coordinates are defined.

This force component is by default visualized as an arrow acting at the connector to which it is connected. The diameter and color of the arrow are fixed and can be defined via parameters **diameter** and **color**. The arrow points in the direction defined by the inPort.signal signals. The length of the arrow is proportional to the length of the force vector using parameter **N\_to\_m** as scaling factor. For example, if  $N_{to\_m} = 100 \text{ N/m}$ , then a force of 350 N is displayed as an arrow of length 3.5 m.

An example how to use this model is given in the following figure:



This leads to the following animation



## Parameters

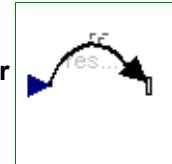
Name	Default	Description
animation	true	= true, if animation shall be enabled
if animation = true		
N_to_m	world.defaultN_to_m	Force arrow scaling (length = force/N_to_m) [N/m]
diameter	world.defaultArrowDiameter	Diameter of force arrow [m]
color	Modelica.Mechanics.MultiBody...	Color of arrow
specularCoefficient	world.defaultSpecularCoeffic...	Reflection of ambient light (= 0: light is completely absorbed)

## Connectors

Name	Description
frame_b	Coordinate system fixed to the component with one cut-force and cut-torque
frame_resolve	If connected, the input signals are resolved in this frame
force[3]	x-, y-, z-coordinates of force resolved in frame_b or frame_resolve (if connected)

## Modelica.Mechanics.MultiBody.Forces.FrameTorque

External torque acting at frame\_b, defined by 3 input signals and resolved in frame\_b or in frame\_resolve

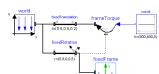


## Information

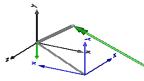
The 3 signals of the **torque** connector are interpreted as the x-, y- and z-coordinates of a **torque** acting at the frame connector to which this component is attached. If connector **frame\_resolve** is not connected, the torque coordinates are with respect to **frame\_b**. If connector **frame\_resolve** is connected, the torque coordinates are with respect to **frame\_resolve**. In this case the force and torque in connector **frame\_resolve** are set to zero, i.e., this connector is solely used to provide the information of the coordinate system, in which the force coordinates are defined.

This torque component is by default visualized as an arrow acting at the connector to which it is connected. The diameter and color of the arrow are fixed and can be defined via parameters **diameter** and **color**. The arrow points in the direction defined by the **inPort.signal** signals. The length of the arrow is proportional to the length of the torque vector using parameter **Nm\_to\_m** as scaling factor. For example, if **Nm\_to\_m** = 100 N/m, then a torque of 350 Nm is displayed as an arrow of length 3.5 m.

An example how to use this model is given in the following figure:



This leads to the following animation



## Parameters

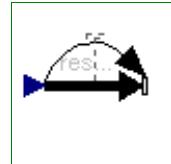
Name	Default	Description
animation	true	= true, if animation shall be enabled
if animation = true		
Nm_to_m	world.defaultNm_to_m	Torque arrow scaling (length = torque/Nm_to_m) [N.m/m]
diameter	world.defaultArrowDiameter	Diameter of torque arrow [m]
color	Modelica.Mechanics.MultiBody...	Color of arrow
specularCoefficient	world.defaultSpecularCoeffic...	Reflection of ambient light (= 0: light is completely absorbed)

## Connectors

Name	Description
frame_b	Coordinate system fixed to the component with one cut-force and cut-torque
frame_resolve	If connected, the input signals are resolved in this frame
torque[3]	x-, y-, z-coordinates of torque resolved in frame_b or frame_resolve (if connected)

## Modelica.Mechanics.MultiBody.Forces.FrameForceAndTorque

External force and torque acting at frame\_b, defined by 6 input signals and resolved in frame\_b or in frame\_resolve



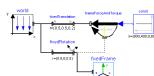
## Information

The **6** signals of the **load** connector are interpreted as the x-, y- and z-coordinates of a **force** and as the x-, y-, and z-coordinates of a **torque** acting at the frame connector to which this component is attached. If connector **frame\_resolve** is **not** connected, the force and torque coordinates are with respect to **frame\_b**. If connector **frame\_resolve** is connected, the force and torque coordinates are with respect to **frame\_resolve**. In this case the force and torque in connector **frame\_resolve** are set to zero, i.e., this connector is solely used to provide the information of the coordinate system, in which the force coordinates are defined. The input signals are mapped to the force and torque in the following way:

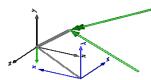
```
force  = load[1:3]
torque = load[4:6]
```

The force and torque are by default visualized as an arrow (force) and as a double arrow (torque) acting at the connector to which they are connected. The diameters and colors of the arrows are fixed and can be defined via parameters **forceDiameter**, **torqueDiameter**, **forceColor** and **torqueColor**. The arrows point in the directions defined by the inPort.signal signals. The lengths of the arrows are proportional to the length of the force and torque vectors, respectively, using parameters **N\_to\_m** and **Nm\_to\_m** as scaling factors. For example, if **N\_to\_m** = 100 N/m, then a force of 350 N is displayed as an arrow of length 3.5 m.

An example how to use this model is given in the following figure:



This leads to the following animation



## Parameters

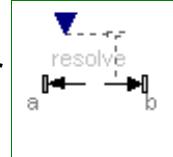
Name	Default	Description
animation	true	= true, if animation shall be enabled
if animation = true		
N_to_m	world.defaultN_to_m	Force arrow scaling (length = force/N_to_m) [N/m]
Nm_to_m	world.defaultNm_to_m	Torque arrow scaling (length = torque/Nm_to_m) [N.m/m]
forceDiameter	world.defaultArrowDiameter	Diameter of force arrow [m]
torqueDiameter	forceDiameter	Diameter of torque arrow [m]
forceColor	Modelica.Mechanics.MultiBody...	Color of force arrow
torqueColor	Modelica.Mechanics.MultiBody...	Color of torque arrow
specularCoefficient	world.defaultSpecularCoeffic...	Reflection of ambient light (= 0: light is completely absorbed)

## Connectors

Name	Description
frame_b	Coordinate system fixed to the component with one cut-force and cut-torque
frame_resolve	If connected, the input signals are resolved in this frame
load[6]	[1:6] = x-, y-, z-coordinates of force and x-, y-, z-coordinates of torque resolved in frame_b or frame_resolved (if connected)

## Modelica.Mechanics.MultiBody.Forces.Force

Force acting between two frames, defined by 3 input signals and resolved in frame\_b or in frame\_resolve

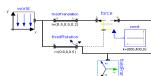


## Information

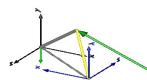
The 3 signals of the **force** connector are interpreted as the x-, y- and z-coordinates of a **force** acting at the frame connector to which frame\_b of this component is attached. If connector **frame\_resolve** is not connected, the force coordinates of the inPort connector are with respect to **frame\_b**. If connector **frame\_resolve** is connected, the force coordinates of the inPort connector are with respect to **frame\_resolve**. In this case the force in connector **frame\_resolve** is set to zero, i.e., this connector is solely used to provide the information of the coordinate system, in which the force coordinates are defined.

Note, the cut-torque in frame\_b (frame\_b.t) is set to zero. Additionally, a force and torque acts on frame\_a in such a way that the force and torque balance between frame\_a and frame\_b is fulfilled.

An example how to use this model is given in the following figure:



This leads to the following animation (the yellow cylinder characterizes the line between frame\_a and frame\_b of the Force component, i.e., the force acts with negative sign also on the opposite side of this cylinder, but for clarity this is not shown in the animation):



## Parameters

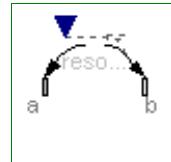
Name	Default	Description
animation	true	= true, if animation shall be enabled
if animation = true		
N_to_m	world.defaultN_to_m	Force arrow scaling (length = force/N_to_m) [N/m]
forceDiameter	world.defaultArrowDiameter	Diameter of force arrow [m]
connectionLineDiameter	forceDiameter	Diameter of line connecting frame_a and frame_b [m]
forceColor	Modelica.Mechanics.MultiBody...	Color of force arrow
connectionLineColor	Modelica.Mechanics.MultiBody...	Color of line connecting frame_a and frame_b
specularCoefficient	world.defaultSpecularCoeffic...	Reflection of ambient light (= 0: light is completely absorbed)

## Connectors

Name	Description
frame_a	Coordinate system fixed to the component with one cut-force and cut-torque
frame_b	Coordinate system fixed to the component with one cut-force and cut-torque
frame_resolve	If connected, the input signals are resolved in this frame
force[3]	x-, y-, z-coordinates of force resolved in frame_b or frame_resolved (if connected)

## Modelica.Mechanics.MultiBody.Forces.Torque

Torque acting between two frames, defined by 3 input signals and resolved in frame\_b or in frame\_resolve

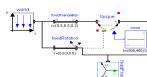


## Information

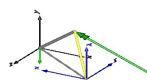
The 3 signals of the **torque** connector are interpreted as the x-, y- and z-coordinates of a **torque** acting at the frame connector to which frame\_b of this component is attached. If connector **frame\_resolve** is not connected, the torque coordinates of the inPort connector are with respect to **frame\_b**. If connector **frame\_resolve** is connected, the torque coordinates of the inPort connector are with respect to **frame\_resolve**. In this case the torque in connector **frame\_resolve** is set to zero, i.e., this connector is solely used to provide the information of the coordinate system, in which the torque coordinates are defined.

Note, the cut-forces in frame\_a and frame\_b (frame\_a.f, frame\_b.f) are set to zero and the cut-torque at frame\_a (frame\_a.t) is the same as the cut-torque at frame\_b (frame\_b.t) but with opposite sign.

An example how to use this model is given in the following figure:



This leads to the following animation (the yellow cylinder characterizes the line between frame\_a and frame\_b of the Torque component, i.e., the torque acts with negative sign also on the opposite side of this cylinder, but for clarity this is not shown in the animation):



## Parameters

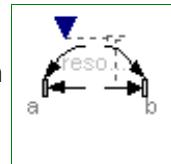
Name	Default	Description
animation	true	= true, if animation shall be enabled
if animation = true		
Nm_to_m	world.defaultNm_to_m	Torque arrow scaling (length = torque/Nm_to_m) [N.m/m]
torqueDiameter	world.defaultArrowDiameter	Diameter of torque arrow [m]
connectionLineDiameter	torqueDiameter	Diameter of line connecting frame_a and frame_b [m]
torqueColor	Modelica.Mechanics.MultiBody...	Color of torque arrow
connectionLineColor	Modelica.Mechanics.MultiBody...	Color of line connecting frame_a and frame_b
specularCoefficient	world.defaultSpecularCoeffic...	Reflection of ambient light (= 0: light is completely absorbed)

## Connectors

Name	Description
frame_a	Coordinate system fixed to the component with one cut-force and cut-torque
frame_b	Coordinate system fixed to the component with one cut-force and cut-torque
frame_resolve	If connected, the input signals are resolved in this frame
torque[3]	x-, y-, z-coordinates of torque resolved in frame_b or frame_resolved (if connected)

## Modelica.Mechanics.MultiBody.Forces.ForceAndTorque

Force and torque acting between two frames, defined by 6 input signals and resolved in frame\_b or in frame\_resolve



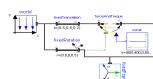
## Information

The **6** signals of the **load** connector are interpreted as the x-, y- and z-coordinates of a **force** and as the x-, y-, and z-coordinates of a **torque** acting at the frame connector to which frame\_b of this component is attached. If connector **frame\_resolve** is **not** connected, the force and torque coordinates are with respect to **frame\_b**. If connector **frame\_resolve** is connected, the force and torque coordinates are with respect to **frame\_resolve**. In this case the force and torque in connector **frame\_resolve** are set to zero, i.e., this connector is solely used to provide the information of the coordinate system, in which the force/torque coordinates are defined. The input signals are mapped to the force and torque in the following way:

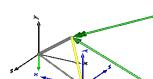
```
force  = load[1:3]
torque = load[4:6]
```

Additionally, a force and torque acts on frame\_a in such a way that the force and torque balance between frame\_a and frame\_b is fulfilled.

An example how to use this model is given in the following figure:



This leads to the following animation (the yellow cylinder characterizes the line between frame\_a and frame\_b of the ForceAndTorque component, i.e., the force and torque acts with negative sign also on the opposite side of this cylinder, but for clarity this is not shown in the animation):



## Parameters

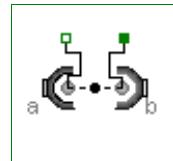
Name	Default	Description
animation	true	= true, if animation shall be enabled
if animation = true		
N_to_m	world.defaultN_to_m	Force arrow scaling (length = force/N_to_m) [N/m]
Nm_to_m	world.defaultNm_to_m	Torque arrow scaling (length = torque/Nm_to_m) [N.m/m]
forceDiameter	world.defaultArrowDiameter	Diameter of force arrow [m]
torqueDiameter	forceDiameter	Diameter of torque arrow [m]
connectionLineDiameter	forceDiameter	Diameter of line connecting frame_a and frame_b [m]
forceColor	Modelica.Mechanics.MultiBody...	Color of force arrow
torqueColor	Modelica.Mechanics.MultiBody...	Color of torque arrow
connectionLineColor	Modelica.Mechanics.MultiBody...	Color of line connecting frame_a and frame_b
specularCoefficient	world.defaultSpecularCoeffic...	Reflection of ambient light (= 0: light is completely absorbed)

## Connectors

Name	Description
frame_a	Coordinate system fixed to the component with one cut-force and cut-torque
frame_b	Coordinate system fixed to the component with one cut-force and cut-torque
frame_resolve	If connected, the input signals are resolved in this frame
load[6]	[1:6] = x-, y-, z-coordinates of force and x-, y-, z-coordinates of torque resolved in frame_b or frame_resolved (if connected)

## Modelica.Mechanics.MultiBody.Forces.LineForceWithMass

General line force component with an optional point mass on the connection line



## Information

This component is used to exert a **line force** between the origin of frame\_a and the origin of frame\_b by attaching components of the **1-dimensional translational** mechanical library of Modelica (Modelica.Mechanics.Translational) between the two flange connectors **flange\_a** and **flange\_b**. Optionally, there is a **point mass** on the line connecting the origin of frame\_a and the origin of frame\_b. This point mass approximates the **mass** of the **force element**. The distance of the point mass from frame\_a as a fraction of the distance between frame\_a and frame\_b is defined via parameter **lengthFraction** (default is 0.5, i.e., the point mass is in the middle of the line).

In the translational library there is the implicit assumption that forces of components that have only one flange connector act with opposite sign on the bearings of the component. This assumption is also used in the LineForceWithMass component: If a connection is present to only one of the flange connectors, then the force in this flange connector acts implicitly with opposite sign also in the other flange connector.

## Parameters

Name	Default	Description
animateLine	true	= true, if a line shape between frame_a and frame_b shall be visualized

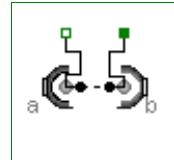
animateMass	true	= true, if point mass shall be visualized as sphere provided $m > 0$
m	0	Mass of point mass on the connection line between the origin of frame_a and the origin of frame_b [kg]
lengthFraction	0.5	Location of point mass with respect to frame_a as a fraction of the distance from frame_a to frame_b
<b>Animation</b>		
specularCoefficient	world.defaultSpecularCoeffic...	Reflection of ambient light (= 0: light is completely absorbed)
if animateLine = true		
lineShapeType	"cylinder"	Type of shape visualizing the line from frame_a to frame_b
lineShapeWidth	world.defaultArrowDiameter	Width of shape [m]
lineShapeHeight	lineShapeWidth	Height of shape [m]
lineShapeExtra	0.0	Extra parameter for shape
lineShapeColor	Modelica.Mechanics.MultiBody.. .	Color of line shape
if animateMass = true		
massDiameter	world.defaultBodyDiameter	Diameter of point mass sphere
massColor	Modelica.Mechanics.MultiBody.. .	Color of point mass
<b>Advanced</b>		
s_small	1.E-10	Prevent zero-division if distance between frame_a and frame_b is zero [m]

## Connectors

Name	Description
frame_a	Coordinate system fixed to the component with one cut-force and cut-torque
frame_b	Coordinate system fixed to the component with one cut-force and cut-torque
flange_b	1-dim. translational flange (connect force of Translational library between flange_a and flange_b)
flange_a	1-dim. translational flange (connect force of Translational library between flange_a and flange_b)

## Modelica.Mechanics.MultiBody.Forces.LineForceWithTwoMasses

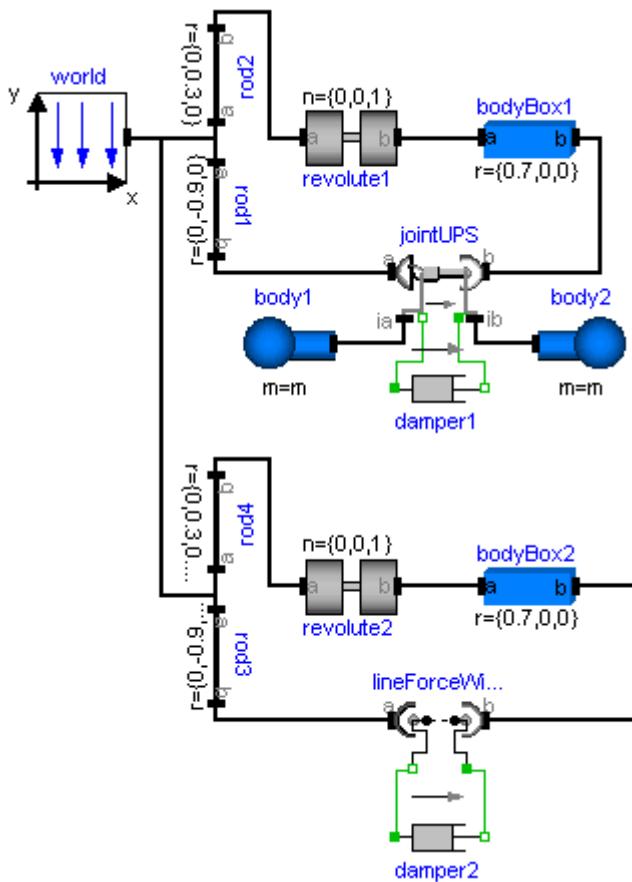
General line force component with two optional point masses on the connection line



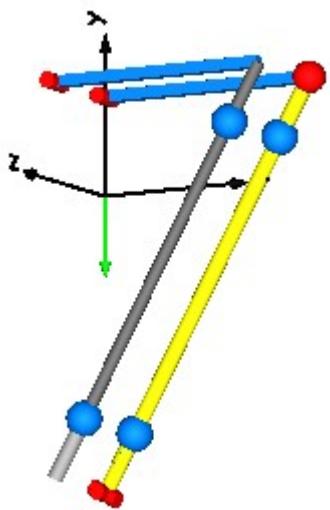
## Information

This component is used to exert a **line force** between the origin of frame\_a and the origin of frame\_b by attaching components of the **1-dimensional translational** mechanical library of Modelica (Modelica.Mechanics.Translational) between the two flange connectors **flange\_a** and **flange\_b**. Optionally, there are **two point masses** on the line connecting the origin of frame\_a and the origin of frame\_b. These point masses approximate the **masses** of the **force element**. The locations of the two point masses are defined by their (fixed) distances of L\_a relative to frame\_a and of L\_b relative to frame\_b, respectively.

In example [MultiBody.Examples.Elementary.LineForceWithTwoMasses](#) the usage of this line force element is shown and is compared with an alternative implementation using a [MultiBody.Joints.Assemblies.JointUPS](#) component. The composition diagram of this example is displayed in the figure below.



The animation view at time = 0 is shown in the next figure. The system on the left side in the front is the animation with the LineForceWithTwoMasses component whereas the system on the right side in the back is the animation with the JointUPS component. Both implementations yield the same result. However, the implementation with the LineForceWithTwoMasses component is simpler.



In the translational library there is the implicit assumption that forces of components that have only one flange connector act with opposite sign on the bearings of the component. This assumption is also used in the LineForceWithTwoMasses component: If a connection is present to only one of the flange connectors, then the force in this flange connector acts implicitly with opposite sign also in the other flange connector.

## Parameters

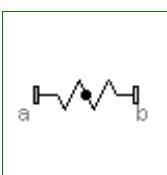
Name	Default	Description
animate	true	= true, if animation shall be enabled
animateMasses	true	= true, if point masses shall be visualized provided animate=true and m_a, m_b > 0
m_a	0	Mass of point mass a on the connection line between the origin of frame_a and the origin of frame_b [kg]
m_b	0	Mass of point mass b on the connection line between the origin of frame_a and the origin of frame_b [kg]
L_a	0	Distance between point mass a and frame_a (positive, if in direction of frame_b) [m]
L_b	L_a	Distance between point mass b and frame_b (positive, if in direction of frame_a) [m]
<b>Animation</b>		
Cylinder at frame_a if animation = true		
cylinderDiameter_a	world.defaultForceWidth	Diameter of cylinder at frame_a [m]
cylinderLength_a	2*L_a	Length of cylinder at frame_a [m]
color_a	{155,155,155}	Color of cylinder at frame_a
specularCoefficient	world.defaultSpecularCoeffic...	Reflection of ambient light (= 0: light is completely absorbed)
Cylinder at frame_b if animation = true		
diameterFraction	0.8	Diameter of cylinder at frame_b with respect to diameter of cylinder at frame_a
cylinderLength_b	2*L_b	Length of cylinder at frame_b [m]
color_b	{100,100,100}	Color of cylinder at frame_b
if animation = true and animateMasses = true		
massDiameterFaction	1.7	Diameter of point mass spheres with respect to cylinderDiameter_a
massColor	Modelica.Mechanics.MultiBody.. .	Color of point masses
<b>Advanced</b>		
s_small	1.E-10	Prevent zero-division if distance between frame_a and frame_b is zero [m]

## Connectors

Name	Description
frame_a	Coordinate system fixed to the component with one cut-force and cut-torque
frame_b	Coordinate system fixed to the component with one cut-force and cut-torque
flange_b	1-dim. translational flange (connect force of Translational library between flange_a and flange_b)
flange_a	1-dim. translational flange (connect force of Translational library between flange_a and flange_b)

## Modelica.Mechanics.MultiBody.Forces.Spring

Linear translational spring with optional mass



## Information

Linear spring acting as line force between frame\_a and frame\_b. A **force f** is exerted on the origin of

frame\_b and with opposite sign on the origin of frame\_a along the line from the origin of frame\_a to the origin of frame\_b according to the equation:

$$f = c * (s - s_{unstretched});$$

where "c" and "s\_unstretched" are parameters and "s" is the distance between the origin of frame\_a and the origin of frame\_b.

Optionally, the mass of the spring is taken into account by a point mass located on the line between frame\_a and frame\_b (default: middle of the line). If the spring mass is zero, the additional equations to handle the mass are removed.

In the following figure a typical animation of the spring is shown. The blue sphere in the middle of the spring characterizes the location of the point mass.



## Parameters

Name	Default	Description
animation	true	= true, if animation shall be enabled
showMass	true	= true, if point mass shall be visualized as sphere if animation=true and m>0
c		Spring constant [N/m]
s_unstretched	0	Unstretched spring length [m]
m	0	Spring mass located on the connection line between the origin of frame_a and the origin of frame_b [kg]
lengthFraction	0.5	Location of spring mass with respect to frame_a as a fraction of the distance from frame_a to frame_b (=0: at frame_a; =1: at frame_b)

### Animation

if animation = true

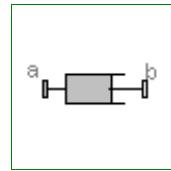
width	world.defaultForceWidth	Width of spring [m]
coilWidth	width/10	Width of spring coil [m]
numberOfWindings	5	Number of spring windings
color	Modelica.Mechanics.MultiBody..	Color of spring
specularCoefficient	world.defaultSpecularCoeffic...	Reflection of ambient light (= 0: light is completely absorbed)
if animation = true and showMass = true		
massDiameter	max(0, (width - 2*coilWidth)...)	Diameter of mass point sphere
massColor	Modelica.Mechanics.MultiBody..	Color of mass point

## Connectors

Name	Description
frame_a	Coordinate system fixed to the component with one cut-force and cut-torque
frame_b	Coordinate system fixed to the component with one cut-force and cut-torque

## Modelica.Mechanics.MultiBody.Forces.Damper

Linear (velocity dependent) damper



### Information

**Linear damper** acting as line force between frame\_a and frame\_b. A **force f** is exerted on the origin of frame\_b and with opposite sign on the origin of frame\_a along the line from the origin of frame\_a to the origin of frame\_b according to the equation:

$$f = d * \text{der}(s);$$

where "d" is a parameter, "s" is the distance between the origin of frame\_a and the origin of frame\_b and  $\text{der}(s)$  is the time derivative of "s".

In the following figure a typical animation is shown where a mass is hanging on a damper.



### Parameters

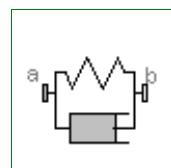
Name	Default	Description
animation	true	= true, if animation shall be enabled
d	0	Damping constant [N.s/m]
<b>Animation</b>		
if animation = true		
length_a	world.defaultForceLength	Length of cylinder at frame_a side [m]
diameter_a	world.defaultForceWidth	Diameter of cylinder at frame_a side [m]
diameter_b	0.6*diameter_a	Diameter of cylinder at frame_b side
color_a	{100,100,100}	Color at frame_a
color_b	{155,155,155}	Color at frame_b
specularCoefficient	world.defaultSpecularCoeffic...	Reflection of ambient light (= 0: light is completely absorbed)
<b>Advanced</b>		
s_small	1.E-6	Prevent zero-division if relative distance s=0 [m]

### Connectors

Name	Description
frame_a	Coordinate system fixed to the force element with one cut-force and cut-torque
frame_b	Coordinate system fixed to the force element with one cut-force and cut-torque

## Modelica.Mechanics.MultiBody.Forces.SpringDamperParallel

Linear spring and linear damper in parallel



### Information

**Linear spring** and **linear damper** in parallel acting as line force between frame\_a and frame\_b. A **force f** is exerted on the origin of frame\_b and with opposite sign on the origin of frame\_a along the line from the origin of frame\_a to the origin of frame\_b according to the equation:

$$f = c * (s - s_{\text{unstretched}}) + d * \text{der}(s);$$

## 462 Modelica.Mechanics.MultiBody.Forces.SpringDamperParallel

where "c", "s\_unstretched" and "d" are parameters, "s" is the distance between the origin of frame\_a and the origin of frame\_b and der(s) is the time derivative of s.

### Parameters

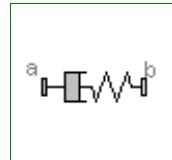
Name	Default	Description
animation	true	= true, if animation shall be enabled
c		Spring constant [N/m]
s_unstretched	0	Unstretched spring length [m]
d	0	Damping constant [N.s/m]
<b>Animation</b>		
if animation = true		
width	world.defaultForceWidth	Width of spring [m]
coilWidth	width/10	Width of spring coil [m]
numberOfWindings	5	Number of spring windings
color	Modelica.Mechanics.MultiBody...	Color of spring
specularCoefficient	world.defaultSpecularCoeffic...	Reflection of ambient light (= 0: light is completely absorbed)
<b>Advanced</b>		
s_small	1.E-6	Prevent zero-division if relative distance s=0 [m]

### Connectors

Name	Description
frame_a	Coordinate system fixed to the force element with one cut-force and cut-torque
frame_b	Coordinate system fixed to the force element with one cut-force and cut-torque

## Modelica.Mechanics.MultiBody.Forces.SpringDamperSeries

Linear spring and linear damper in series connection



### Information

Linear spring and linear damper in series connection acting as line force between frame\_a and frame\_b:

```
frame_a --> damper ----> spring --> frame_b
          |           |
          |-- s_damper --| (s_damper is the state variable of this system)
```

A **force f** is exerted on the origin of frame\_b and with opposite sign on the origin of frame\_a along the line from the origin of frame\_a to the origin of frame\_b according to the equations:

$$\begin{aligned} f &= c * (s - s_{\text{unstretched}} - s_{\text{damper}}); \\ f &= d * \text{der}(s_{\text{damper}}); \end{aligned}$$

where "c", "s\_unstretched" and "d" are parameters, "s" is the distance between the origin of frame\_a and the origin of frame\_b. "s\_damper" is the length of the damper (= an internal state of this force element) and der(s\_damper) is the time derivative of s\_damper.

### Parameters

Name	Default	Description

c		Spring constant [N/m]
s_unstretched	0	Unstretched spring length [m]
d	0	Damping constant [N.s/m]
s_damper_start	0	Initial length of damper [m]
<b>Advanced</b>		
s_small	1.E-6	Prevent zero-division if relative distance s=0 [m]

## Connectors

Name	Description
frame_a	Coordinate system fixed to the force element with one cut-force and cut-torque
frame_b	Coordinate system fixed to the force element with one cut-force and cut-torque

## Modelica.Mechanics.MultiBody.Frames

### Functions to transform rotational frame quantities

#### Information

Package **Frames** contains type definitions and functions to transform rotational frame quantities. The basic idea is to hide the actual definition of an **orientation** in this package by providing essentially type **Orientation** together with **functions** operating on instances of this type.

#### Content

In the table below an example is given for every function definition. The used variables have the following declaration:

```
Frames.Orientation R, R1, R2, R_rel, R_inv;
Real[3,3] T, T_inv;
Real[3] v1, v2, w1, w2, n_x, n_y, n_z, e, e_x, res_ori, phi;
Real[6] res_equal;
Real L, angle;
```

Function/type	Description
<b>Orientation R;</b>	New type defining an orientation object that describes the rotation of frame 1 into frame 2.
<b>res_ori = orientationConstraint(R);</b>	Return the constraints between the variables of an orientation object (shall be zero).
<b>w1 = angularVelocity1(R);</b>	Return angular velocity resolved in frame 1 from orientation object R.
<b>w2 = angularVelocity2(R);</b>	Return angular velocity resolved in frame 2 from orientation object R.
<b>v1 = resolve1(R,v2);</b>	Transform vector v2 from frame 2 to frame 1.
<b>v2 = resolve2(R,v1);</b>	Transform vector v1 from frame 1 to frame 2.
<b>v2 = resolveRelative(v1,R1,R2);</b>	Transform vector v1 from frame 1 to frame 2 using absolute orientation objects R1 of frame 1 and R2 of frame 2.
<b>D1 = resolveDyade1(R,D2);</b>	Transform second order tensor D2 from frame 2 to frame 1.
<b>D2 = resolveDyade2(R,D1);</b>	Transform second order tensor D1 from frame 1 to frame 2.
<b>R = nullRotation()</b>	Return orientation object R that does not rotate a frame.
<b>R_inv = inverseRotation(R);</b>	Return inverse orientation object.
<b>R_rel = relativeRotation(R1,R2);</b>	Return relative orientation object from two absolute orientation objects.

<code>R2 = absoluteRotation(R1,R_rel);</code>	Return absolute orientation object from another absolute and a relative orientation object.
<code>R = planarRotation(e, angle, der_angle);</code>	Return orientation object of a planar rotation.
<code>angle = planarRotationAngle(e, v1, v2);</code>	Return angle of a planar rotation, given the rotation axis and the representations of a vector in frame 1 and frame 2.
<code>R = axisRotation(axis, angle, der_angle);</code>	Return orientation object R to rotate around angle along axis of frame 1.
<code>R = axesRotations(sequence, angles, der_angles);</code>	Return rotation object to rotate in sequence around 3 axes. Example: <code>R = axesRotations({1,2,3},{pi/2,pi/4,-pi}, zeros(3));</code>
<code>angles = axesRotationsAngles(R, sequence);</code>	Return the 3 angles to rotate in sequence around 3 axes to construct the given orientation object.
<code>phi = smallRotation(R);</code>	Return rotation angles phi valid for a small rotation R.
<code>R = from_nxy(n_x, n_y);</code>	Return orientation object from n_x and n_y vectors.
<code>R = from_nxz(n_x, n_z);</code>	Return orientation object from n_x and n_z vectors.
<code>R = from_T(T,w);</code>	Return orientation object R from transformation matrix T and its angular velocity w.
<code>R = from_T2(T,der(T));</code>	Return orientation object R from transformation matrix T and its derivative der(T).
<code>R = from_T_inv(T_inv,w);</code>	Return orientation object R from inverse transformation matrix T_inv and its angular velocity w.
<code>R = from_Q(Q,w);</code>	Return orientation object R from quaternion orientation object Q and its angular velocity w.
<code>T = to_T(R);</code>	Return transformation matrix T from orientation object R.
<code>T_inv = to_T_inv(R);</code>	Return inverse transformation matrix T_inv from orientation object R.
<code>Q = to_Q(R);</code>	Return quaternion orientation object Q from orientation object R.
<code>exy = to_exy(R);</code>	Return [e_x, e_y] matrix of an orientation object R, with e_x and e_y vectors of frame 2, resolved in frame 1.
<code>L = length(n_x);</code>	Return length L of a vector n_x.
<code>e_x = normalize(n_x);</code>	Return normalized vector e_x of n_x such that length of e_x is one.
<code>e = axis(i);</code>	Return unit vector e directed along axis i
<code>Quaternions</code>	<b>Package</b> with functions to transform rotational frame quantities based on quaternions (also called Euler parameters).
<code>TransformationMatrices</code>	<b>Package</b> with functions to transform rotational frame quantities based on transformation matrices.

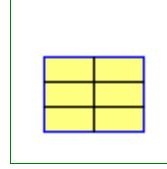
## Package Content

Name	Description
 <code>Orientation</code>	Orientation object defining rotation from a frame 1 into a frame 2
 <code>orientationConstraint</code>	Return residues of orientation constraints (shall be zero)
 <code>angularVelocity1</code>	Return angular velocity resolved in frame 1 from orientation object
 <code>angularVelocity2</code>	Return angular velocity resolved in frame 2 from orientation object
 <code>resolve1</code>	Transform vector from frame 2 to frame 1
 <code>resolve2</code>	Transform vector from frame 1 to frame 2
 <code>resolveRelative</code>	Transform vector from frame 1 to frame 2 using absolute orientation objects of frame 1 and of frame 2
 <code>resolveDyade1</code>	Transform second order tensor from frame 2 to frame 1

 <a href="#">resolveDyade2</a>	Transform second order tensor from frame 1 to frame 2
 <a href="#">nullRotation</a>	Return orientation object that does not rotate a frame
 <a href="#">inverseRotation</a>	Return inverse orientation object
 <a href="#">relativeRotation</a>	Return relative orientation object
 <a href="#">absoluteRotation</a>	Return absolute orientation object from another absolute and a relative orientation object
 <a href="#">planarRotation</a>	Return orientation object of a planar rotation
 <a href="#">planarRotationAngle</a>	Return angle of a planar rotation, given the rotation axis and the representations of a vector in frame 1 and frame 2
 <a href="#">axisRotation</a>	Return rotation object to rotate around an angle along one frame axis
 <a href="#">axesRotations</a>	Return fixed rotation object to rotate in sequence around fixed angles along 3 axes
 <a href="#">axesRotationsAngles</a>	Return the 3 angles to rotate in sequence around 3 axes to construct the given orientation object
 <a href="#">smallRotation</a>	Return rotation angles valid for a small rotation and optionally residues that should be zero
 <a href="#">from_nxy</a>	Return fixed orientation object from n_x and n_y vectors
 <a href="#">from_nxz</a>	Return fixed orientation object from n_x and n_z vectors
 <a href="#">from_T</a>	Return orientation object R from transformation matrix T
 <a href="#">from_T2</a>	Return orientation object R from transformation matrix T and its derivative der(T)
 <a href="#">from_T_inv</a>	Return orientation object R from inverse transformation matrix T_inv
 <a href="#">from_Q</a>	Return orientation object R from quaternion orientation object Q
 <a href="#">to_T</a>	Return transformation matrix T from orientation object R
 <a href="#">to_T_inv</a>	Return inverse transformation matrix T_inv from orientation object R
 <a href="#">to_Q</a>	Return quaternion orientation object Q from orientation object R
 <a href="#">to_vector</a>	Map rotation object into vector
 <a href="#">to_exy</a>	Map rotation object into e_x and e_y vectors of frame 2, resolved in frame 1
 <a href="#">length</a>	Return length of a vector
 <a href="#">normalize</a>	Return normalized vector such that length = 1
 <a href="#">axis</a>	Return unit vector for x-, y-, or z-axis
 <a href="#">Quaternions</a>	Functions to transform rotational frame quantities based on quaternions (also called Euler parameters)
 <a href="#">TransformationMatrices</a>	Functions for transformation matrices

## Modelica.Mechanics.MultiBody.Frames.Orientation

Orientation object defining rotation from a frame 1 into a frame 2



### Information

This object describes the **rotation** from a **frame 1** into a **frame 2**. An instance of this type should never be directly accessed but only with the access functions provided in package

## 466 Modelica.Mechanics.MultiBody.Frames.Orientation

---

Modelica.Mechanics.MultiBody.Frames. As a consequence, it is not necessary to know the internal representation of this object as described in the next paragraphs.

"Orientation" is defined to be a record consisting of two elements: "Real T[3,3]", the transformation matrix to rotate frame 1 into frame 2 and "Real w[3]", the angular velocity of frame 2 with respect to frame 1, resolved in frame 2. Element "T" has the following interpretation:

```
Orientation R;  
R.T = [ex, ey, ez];  
e.g., R.T = [1,0,0; 0,1,0; 0,0,1]
```

where  $e_x, e_y, e_z$  are unit vectors in the direction of the x-axis, y-axis, and z-axis of frame 1, resolved in frame 2, respectively. Therefore, if  $\mathbf{v}_1$  is vector  $\mathbf{v}$  resolved in frame 1 and  $\mathbf{v}_2$  is vector  $\mathbf{v}$  resolved in frame 2, the following relationship holds:

$$\mathbf{v}_2 = \mathbf{R} \cdot \mathbf{T} * \mathbf{v}_1$$

The **inverse** orientation  $\mathbf{R}_{\text{inv}} \cdot \mathbf{T} = \mathbf{R} \cdot \mathbf{T}^T$  describes the rotation from frame 2 into frame 1.

Since the orientation is described by 9 variables, there are 6 constraints between these variables. These constraints are defined in function **Frames.orientationConstraint**.

R.w is the angular velocity of frame 2 with respect to frame 1, resolved in frame 2. Formally, R.w is defined as:

**skew(R.w) = R.T\*der(transpose(R.T))** with

$$\text{skew}(w) = \begin{vmatrix} 0 & -w[3] & w[2] \\ w[3] & 0 & -w[1] \\ -w[2] & w[1] & 0 \end{vmatrix}$$

---

## Modelica.Mechanics.MultiBody.Frames.OrientationConstraint



Return residues of orientation constraints (shall be zero)

### Inputs

Name	Default	Description
R		Orientation object to rotate frame 1 into frame 2

### Outputs

Name	Description
residue[6]	Residues of constraints between elements of orientation object (shall be zero)

---

## Modelica.Mechanics.MultiBody.Frames.angularVelocity1



Return angular velocity resolved in frame 1 from orientation object

### Inputs

Name	Default	Description
R		Orientation object to rotate frame 1 into frame 2

## Outputs

Name	Description
w[3]	Angular velocity of frame 2 with respect to frame 1 resolved in frame 1 [rad/s]

---

## Modelica.Mechanics.MultiBody.Frames.angularVelocity2

Return angular velocity resolved in frame 2 from orientation object



## Inputs

Name	Default	Description
R		Orientation object to rotate frame 1 into frame 2

## Outputs

Name	Description
w[3]	Angular velocity of frame 2 with respect to frame 1 resolved in frame 2 [rad/s]

---

## Modelica.Mechanics.MultiBody.Frames.resolve1

Transform vector from frame 2 to frame 1



## Inputs

Name	Default	Description
R		Orientation object to rotate frame 1 into frame 2
v2[3]		Vector in frame 2

## Outputs

Name	Description
v1[3]	Vector in frame 1

---

## Modelica.Mechanics.MultiBody.Frames.resolve2

Transform vector from frame 1 to frame 2



## Inputs

Name	Default	Description
R		Orientation object to rotate frame 1 into frame 2
v1[3]		Vector in frame 1

## Outputs

Name	Description
v2[3]	Vector in frame 2

**Modelica.Mechanics.MultiBody.Frames.resolveRelative**

Transform vector from frame 1 to frame 2 using absolute orientation objects of frame 1 and of frame 2

**Inputs**

Name	Default	Description
v1[3]		Vector in frame 1
R1		Orientation object to rotate frame 0 into frame 1
R2		Orientation object to rotate frame 0 into frame 2

**Outputs**

Name	Description
v2[3]	Vector in frame 2

**Modelica.Mechanics.MultiBody.Frames.resolveDyade1**

Transform second order tensor from frame 2 to frame 1

**Inputs**

Name	Default	Description
R		Orientation object to rotate frame 1 into frame 2
D2[3, 3]		Second order tensor resolved in frame 2

**Outputs**

Name	Description
D1[3, 3]	Second order tensor resolved in frame 1

**Modelica.Mechanics.MultiBody.Frames.resolveDyade2**

Transform second order tensor from frame 1 to frame 2

**Inputs**

Name	Default	Description
R		Orientation object to rotate frame 1 into frame 2
D1[3, 3]		Second order tensor resolved in frame 1

**Outputs**

Name	Description
D2[3, 3]	Second order tensor resolved in frame 2

**Modelica.Mechanics.MultiBody.Frames.nullRotation**

Return orientation object that does not rotate a frame



## Outputs

Name	Description
R	Orientation object such that frame 1 and frame 2 are identical

---

## Modelica.Mechanics.MultiBody.Frames.inverseRotation

Return inverse orientation object



## Inputs

Name	Default	Description
R		Orientation object to rotate frame 1 into frame 2

## Outputs

Name	Description
R_inv	Orientation object to rotate frame 2 into frame 1

---

## Modelica.Mechanics.MultiBody.Frames.relativeRotation

Return relative orientation object



## Inputs

Name	Default	Description
R1		Orientation object to rotate frame 0 into frame 1
R2		Orientation object to rotate frame 0 into frame 2

## Outputs

Name	Description
R_rel	Orientation object to rotate frame 1 into frame 2

---

## Modelica.Mechanics.MultiBody.Frames.absoluteRotation

Return absolute orientation object from another absolute and a relative orientation object



## Inputs

Name	Default	Description
R1		Orientation object to rotate frame 0 into frame 1
R_rel		Orientation object to rotate frame 1 into frame 2

## Outputs

Name	Description
R2	Orientation object to rotate frame 0 into frame 2

**Modelica.Mechanics.MultiBody.Frames.planarRotation**

Return orientation object of a planar rotation

**Inputs**

Name	Default	Description
e[3]		Normalized axis of rotation (must have length=1)
angle		Rotation angle to rotate frame 1 into frame 2 along axis e [rad]
der_angle		= der(angle) [rad/s]

**Outputs**

Name	Description
R	Orientation object to rotate frame 1 into frame 2

**Modelica.Mechanics.MultiBody.Frames.planarRotationAngle**

Return angle of a planar rotation, given the rotation axis and the representations of a vector in frame 1 and frame 2

**Information**

A call to this function of the form

```
Real[3]           e, v1, v2;
Modelica.SIunits.Angle angle;
equation
  angle = planarRotationAngle(e, v1, v2);
```

computes the rotation angle "**angle**" of a planar rotation along unit vector **e**, rotating frame 1 into frame 2, given the coordinate representations of a vector "v" in frame 1 (**v1**) and in frame 2 (**v2**). Therefore, the result of this function fulfills the following equation:

```
v2 = resolve2(planarRotation(e,angle), v1)
```

The rotation angle is returned in the range

```
-π <= angle <= π
```

This function makes the following assumptions on the input arguments

- Vector **e** has length 1, i.e.,  $\text{length}(e) = 1$
- Vector "v" is not parallel to **e**, i.e.,  $\text{length}(\text{cross}(e,v)) \neq 0$

The function does not check the above assumptions. If these assumptions are violated, a wrong result will be returned and/or a division by zero will occur.

**Inputs**

Name	Default	Description
e[3]		Normalized axis of rotation to rotate frame 1 around e into frame 2 (must have length=1)
v1[3]		A vector v resolved in frame 1 (shall not be parallel to e)
v2[3]		Vector v resolved in frame 2, i.e., $v2 = \text{resolve2}(\text{planarRotation}(e, \text{angle}), v1)$

## Outputs

Name	Description
angle	Rotation angle to rotate frame 1 into frame 2 along axis e in the range: -pi <= angle <= pi [rad]

---

## Modelica.Mechanics.MultiBody.Frames.axisRotation

Return rotation object to rotate around an angle along one frame axis



## Inputs

Name	Default	Description
axis		Rotate around 'axis' of frame 1
angle		Rotation angle to rotate frame 1 into frame 2 along 'axis' of frame 1 [rad]
der_angle		= der(angle) [rad/s]

## Outputs

Name	Description
R	Orientation object to rotate frame 1 into frame 2

---

## Modelica.Mechanics.MultiBody.Frames.axesRotations

Return fixed rotation object to rotate in sequence around fixed angles along 3 axes



## Inputs

Name	Default	Description
sequence[3]	{1,2,3}	Sequence of rotations from frame 1 to frame 2 along axis sequence[i]
angles[3]		Rotation angles around the axes defined in 'sequence' [rad]
der_angles[3]		= der(angles) [rad/s]

## Outputs

Name	Description
R	Orientation object to rotate frame 1 into frame 2

---

## Modelica.Mechanics.MultiBody.Frames.axesRotationsAngles

Return the 3 angles to rotate in sequence around 3 axes to construct the given orientation object



## Information

A call to this function of the form

```
Frames.Orientation      R;
parameter Integer      sequence[3] = {1,2,3};
Modelica.SIunits.Angle angles[3];
equation
  angle = axesRotationAngles(R, sequence);
```

## 472 Modelica.Mechanics.MultiBody.Frames.axesRotationsAngles

computes the rotation angles "**angles[1:3]**" to rotate frame 1 into frame 2 along axes **sequence[1:3]**, given the orientation object **R** from frame 1 to frame 2. Therefore, the result of this function fulfills the following equation:

```
R = axesRotation(sequence, angles)
```

The rotation angles are returned in the range

```
-π <= angles[i] <= π
```

There are **two solutions** for "angles[1]" in this range. Via the third argument **guessAngle1** (default = 0) the returned solution is selected such that |angles[1] - guessAngle1| is minimal. The orientation object R may be in a singular configuration, i.e., there is an infinite number of angle values leading to the same R. The returned solution is selected by setting angles[1] = guessAngle1. Then angles[2] and angles[3] can be uniquely determined in the above range.

Note, that input argument **sequence** has the restriction that only values 1,2,3 can be used and that sequence[1] ≠ sequence[2] and sequence[2] ≠ sequence[3]. Often used values are:

```
sequence = {1,2,3} // Cardan angle sequence  
= {3,1,3} // Euler angle sequence  
= {3,2,1} // Tait-Bryan angle sequence
```

### Inputs

Name	Default	Description
R		Orientation object to rotate frame 1 into frame 2
sequence[3]	{1,2,3}	Sequence of rotations from frame 1 to frame 2 along axis sequence[i]
guessAngle1	0	Select angles[1] such that  angles[1] - guessAngle1  is a minimum [rad]

### Outputs

Name	Description
angles[3]	Rotation angles around the axes defined in 'sequence' such that R=Frames.axesRotation(sequence,angles); -pi < angles[i] <= pi [rad]

---

## Modelica.Mechanics.MultiBody.Frames.smallRotation

Return rotation angles valid for a small rotation and optionally residues that should be zero



### Inputs

Name	Default	Description
R		Orientation object to rotate frame 1 into frame 2
withResidues	false	= false/true, if 'angles'/'angles and residues' are returned in phi

### Outputs

Name	Description
phi[if withResidues then 6 else 3]	The rotation angles around x-, y-, and z-axis of frame 1 to rotate frame 1 into frame 2 for a small rotation + optionally 3 residues that should be zero [rad]

**Modelica.Mechanics.MultiBody.Frames.from\_nxy**

Return fixed orientation object from n\_x and n\_y vectors

**Information**

It is assumed that the two input vectors n\_x and n\_y are resolved in frame 1 and are directed along the x and y axis of frame 2 (i.e., n\_x and n\_y are orthogonal to each other) The function returns the orientation object R to rotate from frame 1 to frame 2.

The function is robust in the sense that it returns always an orientation object R, even if n\_y is not orthogonal to n\_x. This is performed in the following way:

If n\_x and n\_y are not orthogonal to each other, first a unit vector e\_y is determined that is orthogonal to n\_x and is lying in the plane spanned by n\_x and n\_y. If n\_x and n\_y are parallel or nearly parallel to each other, a vector e\_y is selected arbitrarily such that e\_x and e\_y are orthogonal to each other.

**Inputs**

Name	Default	Description
n_x[3]		Vector in direction of x-axis of frame 2, resolved in frame 1
n_y[3]		Vector in direction of y-axis of frame 2, resolved in frame 1

**Outputs**

Name	Description
R	Orientation object to rotate frame 1 into frame 2

**Modelica.Mechanics.MultiBody.Frames.from\_nxz**

Return fixed orientation object from n\_x and n\_z vectors

**Information**

It is assumed that the two input vectors n\_x and n\_z are resolved in frame 1 and are directed along the x and z axis of frame 2 (i.e., n\_x and n\_z are orthogonal to each other) The function returns the orientation object R to rotate from frame 1 to frame 2.

The function is robust in the sense that it returns always an orientation object R, even if n\_z is not orthogonal to n\_x. This is performed in the following way:

If n\_x and n\_z are not orthogonal to each other, first a unit vector e\_z is determined that is orthogonal to n\_x and is lying in the plane spanned by n\_x and n\_z. If n\_x and n\_z are parallel or nearly parallel to each other, a vector e\_z is selected arbitrarily such that n\_x and e\_z are orthogonal to each other.

**Inputs**

Name	Default	Description
n_x[3]		Vector in direction of x-axis of frame 2, resolved in frame 1
n_z[3]		Vector in direction of z-axis of frame 2, resolved in frame 1

**Outputs**

Name	Description
R	Orientation object to rotate frame 1 into frame 2

**Modelica.Mechanics.MultiBody.Frames.from\_T**

Return orientation object R from transformation matrix T

**Inputs**

Name	Default	Description
T[3, 3]		Transformation matrix to transform vector from frame 1 to frame 2 ( $v2=T*v1$ )
w[3]		Angular velocity from frame 2 with respect to frame 1, resolved in frame 2 ( $\text{skew}(w)=T*\text{der}(\text{transpose}(T))$ ) [rad/s]

**Outputs**

Name	Description
R	Orientation object to rotate frame 1 into frame 2

**Modelica.Mechanics.MultiBody.Frames.from\_T2**

Return orientation object R from transformation matrix T and its derivative der(T)

**Information**

Computes the orientation object from a transformation matrix T and the derivative  $\text{der}(T)$  of the transformation matrix. Usually, it is more efficient to use function "from\_T" instead, where the angular velocity has to be given as input argument. Only if this is not possible or too difficult to compute, use function from\_T2(..).

**Inputs**

Name	Default	Description
T[3, 3]		Transformation matrix to transform vector from frame 1 to frame 2 ( $v2=T*v1$ )
der_T[3, 3]		= $\text{der}(T)$

**Outputs**

Name	Description
R	Orientation object to rotate frame 1 into frame 2

**Modelica.Mechanics.MultiBody.Frames.from\_T\_inv**

Return orientation object R from inverse transformation matrix T\_inv

**Inputs**

Name	Default	Description
T_inv[3, 3]		Inverse transformation matrix to transform vector from frame 2 to frame 1 ( $v1=T_{\text{inv}}*v2$ )
w[3]		Angular velocity from frame 1 with respect to frame 2, resolved in frame 1 ( $\text{skew}(w)=T_{\text{inv}}*\text{der}(\text{transpose}(T_{\text{inv}}))$ ) [rad/s]

**Outputs**

Name	Description
R	Orientation object to rotate frame 1 into frame 2

**Modelica.Mechanics.MultiBody.Frames.from\_Q**

Return orientation object R from quaternion orientation object Q

**Inputs**

Name	Default	Description
Q		Quaternions orientation object to rotate frame 1 into frame 2
w[3]		Angular velocity from frame 2 with respect to frame 1, resolved in frame 2 [rad/s]

**Outputs**

Name	Description
R	Orientation object to rotate frame 1 into frame 2

**Modelica.Mechanics.MultiBody.Frames.to\_T**

Return transformation matrix T from orientation object R

**Inputs**

Name	Default	Description
R		Orientation object to rotate frame 1 into frame 2

**Outputs**

Name	Description
T[3, 3]	Transformation matrix to transform vector from frame 1 to frame 2 ( $v2=T*v1$ )

**Modelica.Mechanics.MultiBody.Frames.to\_T\_inv**

Return inverse transformation matrix T\_inv from orientation object R

**Inputs**

Name	Default	Description
R		Orientation object to rotate frame 1 into frame 2

**Outputs**

Name	Description
T_inv[3, 3]	Inverse transformation matrix to transform vector from frame 2 into frame 1 ( $v1=T\_inv*v2$ )

## 476 Modelica.Mechanics.MultiBody.Frames.to\_Q

---

### Modelica.Mechanics.MultiBody.Frames.to\_Q

Return quaternion orientation object Q from orientation object R



#### Inputs

Name	Default	Description
R		Orientation object to rotate frame 1 into frame 2
Q_guess	Quaternions.nullRotation()	Guess value for output Q (there are 2 solutions; the one closer to Q_guess is used)

#### Outputs

Name	Description
Q	Quaternions orientation object to rotate frame 1 into frame 2

---

### Modelica.Mechanics.MultiBody.Frames.to\_vector

Map rotation object into vector



#### Inputs

Name	Default	Description
R		Orientation object to rotate frame 1 into frame 2

#### Outputs

Name	Description
vec[9]	Elements of R in one vector

---

### Modelica.Mechanics.MultiBody.Frames.to\_exy

Map rotation object into e\_x and e\_y vectors of frame 2, resolved in frame 1



#### Inputs

Name	Default	Description
R		Orientation object to rotate frame 1 into frame 2

#### Outputs

Name	Description
exy[3, 2]	= [e_x, e_y] where e_x and e_y are axes unit vectors of frame 2, resolved in frame 1

---

### Modelica.Mechanics.MultiBody.Frames.length

Return length of a vector



#### Inputs

Name	Default	Description

r[:]		Vector
------	--	--------

## Outputs

Name	Description
r_length	Length of vector r

## Modelica.Mechanics.MultiBody.Frames.normalize

Return normalized vector such that length = 1



## Inputs

Name	Default	Description
r[:]		Vector

## Outputs

Name	Description
r_unitLength[size(r, 1)]	Input vector r normalized to length=1

## Modelica.Mechanics.MultiBody.Frames.axis

Return unit vector for x-, y-, or z-axis



## Inputs

Name	Default	Description
axis		Axis vector to be returned

## Outputs

Name	Description
e[3]	Unit axis vector

## Modelica.Mechanics.MultiBody.Frames.Quaternions

Functions to transform rotational frame quantities based on quaternions (also called Euler parameters)

## Information

Package **Frames.Quaternions** contains type definitions and functions to transform rotational frame quantities with quaternions. Functions of this package are currently only utilized in MultiBody.Parts.Body components, when quaternions shall be used as parts of the body states. Some functions are also used in a new Modelica package for B-Spline interpolation that is able to interpolate paths consisting of position vectors and orientation objects.

## Content

In the table below an example is given for every function definition. The used variables have the following declaration:

## 478 Modelica.Mechanics.MultiBody.Frames.Quaternions

```

Quaternions.Orientation Q, Q1, Q2, Q_rel, Q_inv;
Real[3,3] T, T_inv;
Real[3] v1, v2, w1, w2, n_x, n_y, n_z, res_ori, phi;
Real[6] res_equal;
Real L, angle;

```

<b>Function/type</b>	<b>Description</b>
<b>Orientation Q;</b>	New type defining a quaternion object that describes the rotation of frame 1 into frame 2.
<b>der_Orientation der_Q;</b>	New type defining the first time derivative of Frames.Quaternions.Orientation.
<b>res_ori = orientationConstraint(Q);</b>	Return the constraints between the variables of a quaternion object (shall be zero).
<b>w1 = angularVelocity1(Q, der_Q);</b>	Return angular velocity resolved in frame 1 from quaternion object Q and its derivative der_Q.
<b>w2 = angularVelocity2(Q, der_Q);</b>	Return angular velocity resolved in frame 2 from quaternion object Q and its derivative der_Q.
<b>v1 = resolve1(Q,v2);</b>	Transform vector v2 from frame 2 to frame 1.
<b>v2 = resolve2(Q,v1);</b>	Transform vector v1 from frame 1 to frame 2.
<b>[v1,w1] = multipleResolve1(Q, [v2,w2]);</b>	Transform several vectors from frame 2 to frame 1.
<b>[v2,w2] = multipleResolve2(Q, [v1,w1]);</b>	Transform several vectors from frame 1 to frame 2.
<b>Q = nullRotation()</b>	Return quaternion object R that does not rotate a frame.
<b>Q_inv = inverseRotation(Q);</b>	Return inverse quaternion object.
<b>Q_rel = relativeRotation(Q1,Q2);</b>	Return relative quaternion object from two absolute quaternion objects.
<b>Q2 = absoluteRotation(Q1,Q_rel);</b>	Return absolute quaternion object from another absolute and a relative quaternion object.
<b>Q = planarRotation(e, angle);</b>	Return quaternion object of a planar rotation.
<b>phi = smallRotation(Q);</b>	Return rotation angles phi valid for a small rotation.
<b>Q = from_T(T);</b>	Return quaternion object Q from transformation matrix T.
<b>Q = from_T_inv(T_inv);</b>	Return quaternion object Q from inverse transformation matrix T_inv.
<b>T = to_T(Q);</b>	Return transformation matrix T from quaternion object Q.
<b>T_inv = to_T_inv(Q);</b>	Return inverse transformation matrix T_inv from quaternion object Q.

### Package Content

<b>Name</b>	<b>Description</b>
<b>Orientation</b>	Orientation type defining rotation from a frame 1 into a frame 2 with quaternions {p1,p2,p3,p0}
<b>der_Orientation</b>	First time derivative of Quaternions.Orientation
<b>(f) orientationConstraint</b>	Return residues of orientation constraints (shall be zero)
<b>(f) angularVelocity1</b>	Compute angular velocity resolved in frame 1 from quaternion orientation object and its derivative
<b>(f) angularVelocity2</b>	Compute angular velocity resolved in frame 2 from quaternions orientation object and its derivative
<b>(f) resolve1</b>	Transform vector from frame 2 to frame 1
<b>(f) resolve2</b>	Transform vector from frame 1 to frame 2

(f) multipleResolve1	Transform several vectors from frame 2 to frame 1
(f) multipleResolve2	Transform several vectors from frame 1 to frame 2
(f) nullRotation	Return quaternions orientation object that does not rotate a frame
(f) inverseRotation	Return inverse quaternions orientation object
(f) relativeRotation	Return relative quaternions orientation object
(f) absoluteRotation	Return absolute quaternions orientation object from another absolute and a relative quaternions orientation object
(f) planarRotation	Return quaternions orientation object of a planar rotation
(f) smallRotation	Return rotation angles valid for a small rotation
(f) from_T	Return quaternions orientation object Q from transformation matrix T
(f) from_T_inv	Return quaternions orientation object Q from inverse transformation matrix T_inv
(f) to_T	Return transformation matrix T from quaternion orientation object Q
(f) to_T_inv	Return inverse transformation matrix T_inv from quaternion orientation object Q

## Types and constants

```

type Orientation
  "Orientation type defining rotation from a frame 1 into a frame 2 with
  quaternions {p1,p2,p3,p0}"

  extends InternalQuaternionBase;

  encapsulated function equalityConstraint
    "Return the constraint residues to express that two frames have the same
    quaternion orientation"

    import Modelica;
    import Modelica.Mechanics.MultiBody.Frames.Quaternions;
    extends Modelica.Icons.Function;
    input Quaternions.Orientation Q1
    "Quaternions orientation object to rotate frame 0 into frame 1";
    input Quaternions.Orientation Q2
    "Quaternions orientation object to rotate frame 0 into frame 2";
    output Real residue[3]
    "The half of the rotation angles around x-, y-, and z-axis of frame 1 to
    rotate frame 1 into frame 2 for a small rotation (shall be zero)";
    algorithm
      residue := [Q1[4], Q1[3], -Q1[2], -Q1[1]; -Q1[3], Q1[4], Q1[1], -Q1[2];
                  Q1[2], -Q1[1], Q1[4], -Q1[3]]*Q2;
    end equalityConstraint;

  end Orientation;

type der_Orientation = Real[4] (each unit="1/s")
  "First time derivative of Quaternions.Orientation";

```

---

## Modelica.Mechanics.MultiBody.Frames.Quaternions.Orientation

Orientation type defining rotation from a frame 1 into a frame 2 with quaternions {p1,p2,p3,p0}

## Information

This type describes the **rotation** to rotate a frame 1 into a frame 2 using quaternions (also called **Euler parameters**) according to the following definition:

```
Quaternions.Orientation Q;
Real n[3];
Real phi(unit="rad");
Q = [ n*sin(phi/2)
      cos(phi/2) ]
```

where "n" is the **axis of rotation** to rotate frame 1 into frame 2 and "phi" is the **rotation angle** for this rotation. Vector "n" is either resolved in frame 1 or in frame 2 (the result is the same since the coordinates of "n" with respect to frame 1 are identical to its coordinates with respect to frame 2).

The term "quaternions" is preferred over the historically more reasonable "Euler parameters" in order to not get confused with Modelica "parameters".

## Modelica.Mechanics.MultiBody.Frames.Quaternions.der\_Orientation

First time derivative of Quaternions.Orientation

### Modelica.Mechanics.MultiBody.Frames.Quaternions.orientationConstraint

Return residues of orientation constraints (shall be zero)



#### Inputs

Name	Default	Description
Q		Quaternions orientation object to rotate frame 1 into frame 2

#### Outputs

Name	Description
residue[1]	Residue constraint (shall be zero)

## Modelica.Mechanics.MultiBody.Frames.Quaternions.angularVelocity1

Compute angular velocity resolved in frame 1 from quaternion orientation object and its derivative



#### Inputs

Name	Default	Description
Q		Quaternions orientation object to rotate frame 1 into frame 2
der_Q		Derivative of Q [1/s]

#### Outputs

Name	Description
w[3]	Angular velocity resolved in frame 1 [rad/s]

**Modelica.Mechanics.MultiBody.Frames.Quaternions.angularVelocity2**

Compute angular velocity resolved in frame 2 from quaternions orientation object and its derivative

**Inputs**

Name	Default	Description
Q		Quaternions orientation object to rotate frame 1 into frame 2
der_Q		Derivative of Q [1/s]

**Outputs**

Name	Description
w[3]	Angular velocity of frame 2 with respect to frame 1 resolved in frame 2 [rad/s]

**Modelica.Mechanics.MultiBody.Frames.Quaternions.resolve1**

Transform vector from frame 2 to frame 1

**Inputs**

Name	Default	Description
Q		Quaternions orientation object to rotate frame 1 into frame 2
v2[3]		Vector in frame 2

**Outputs**

Name	Description
v1[3]	Vector in frame 1

**Modelica.Mechanics.MultiBody.Frames.Quaternions.resolve2**

Transform vector from frame 1 to frame 2

**Inputs**

Name	Default	Description
Q		Quaternions orientation object to rotate frame 1 into frame 2
v1[3]		Vector in frame 1

**Outputs**

Name	Description
v2[3]	Vector in frame 2

**Modelica.Mechanics.MultiBody.Frames.Quaternions.multipleResolve1**

Transform several vectors from frame 2 to frame 1



---

**482 Modelica.Mechanics.MultiBody.Frames.Quaternions.multipleResolve1****Inputs**

Name	Default	Description
Q		Quaternions orientation object to rotate frame 1 into frame 2
v2[3, :]		Vectors in frame 2

**Outputs**

Name	Description
v1[3, size(v2, 2)]	Vectors in frame 1

---

**Modelica.Mechanics.MultiBody.Frames.Quaternions.multipleResolve2**

Transform several vectors from frame 1 to frame 2

**Inputs**

Name	Default	Description
Q		Quaternions orientation object to rotate frame 1 into frame 2
v1[3, :]		Vectors in frame 1

**Outputs**

Name	Description
v2[3, size(v1, 2)]	Vectors in frame 2

---

**Modelica.Mechanics.MultiBody.Frames.Quaternions.nullRotation**

Return quaternions orientation object that does not rotate a frame

**Outputs**

Name	Description
Q	Quaternions orientation object to rotate frame 1 into frame 2

---

**Modelica.Mechanics.MultiBody.Frames.Quaternions.inverseRotation**

Return inverse quaternions orientation object

**Inputs**

Name	Default	Description
Q		Quaternions orientation object to rotate frame 1 into frame 2

**Outputs**

Name	Description
Q_inv	Quaternions orientation object to rotate frame 2 into frame 1

**Modelica.Mechanics.MultiBody.Frames.Quaternions.relativeRotation**

Return relative quaternions orientation object

**Inputs**

Name	Default	Description
Q1		Quaternions orientation object to rotate frame 0 into frame 1
Q2		Quaternions orientation object to rotate frame 0 into frame 2

**Outputs**

Name	Description
Q_rel	Quaternions orientation object to rotate frame 1 into frame 2

**Modelica.Mechanics.MultiBody.Frames.Quaternions.absoluteRotation**

Return absolute quaternions orientation object from another absolute and a relative quaternions orientation object

**Inputs**

Name	Default	Description
Q1		Quaternions orientation object to rotate frame 0 into frame 1
Q_rel		Quaternions orientation object to rotate frame 1 into frame 2

**Outputs**

Name	Description
Q2	Quaternions orientation object to rotate frame 0 into frame 2

**Modelica.Mechanics.MultiBody.Frames.Quaternions.planarRotation**

Return quaternions orientation object of a planar rotation

**Inputs**

Name	Default	Description
e[3]		Normalized axis of rotation (must have length=1)
angle		Rotation angle to rotate frame 1 into frame 2 along axis e [rad]

**Outputs**

Name	Description
Q	Quaternions orientation object to rotate frame 1 into frame 2 along axis e

**Modelica.Mechanics.MultiBody.Frames.Quaternions.smallRotation**

Return rotation angles valid for a small rotation

## Inputs

Name	Default	Description
Q		Quaternions orientation object to rotate frame 1 into frame 2

## Outputs

Name	Description
phi[3]	The rotation angles around x-, y-, and z-axis of frame 1 to rotate frame 1 into frame 2 for a small relative rotation [rad]

## Modelica.Mechanics.MultiBody.Frames.Quaternions.from\_T

Return quaternions orientation object Q from transformation matrix T



## Inputs

Name	Default	Description
T[3, 3]		Transformation matrix to transform vector from frame 1 to frame 2 ( $v2=T*v1$ )
Q_guess	nullRotation()	Guess value for Q (there are 2 solutions; the one close to Q_guess is used)

## Outputs

Name	Description
Q	Quaternions orientation object to rotate frame 1 into frame 2 (Q and -Q have same transformation matrix)

## Modelica.Mechanics.MultiBody.Frames.Quaternions.from\_T\_inv

Return quaternions orientation object Q from inverse transformation matrix T\_inv



## Inputs

Name	Default	Description
T_inv[3, 3]		Inverse transformation matrix to transform vector from frame 2 to frame 1 ( $v1=T\_inv*v2$ )
Q_guess	nullRotation()	Guess value for output Q (there are 2 solutions; the one closer to Q_guess is used)

## Outputs

Name	Description
Q	Quaternions orientation object to rotate frame 1 into frame 2 (Q and -Q have same transformation matrix)

## Modelica.Mechanics.MultiBody.Frames.Quaternions.to\_T

Return transformation matrix T from quaternion orientation object Q



## Inputs

Name	Default	Description
Q		Quaternions orientation object to rotate frame 1 into frame 2

## Outputs

Name	Description
T[3, 3]	Transformation matrix to transform vector from frame 1 to frame 2 ( $v2=T*v1$ )

---

## Modelica.Mechanics.MultiBody.Frames.Quaternions.to\_T\_inv

Return inverse transformation matrix T\_inv from quaternion orientation object Q



## Inputs

Name	Default	Description
Q		Quaternions orientation object to rotate frame 1 into frame 2

## Outputs

Name	Description
T_inv[3, 3]	Transformation matrix to transform vector from frame 2 to frame 1 ( $v1=T*v2$ )

---

## Modelica.Mechanics.MultiBody.Frames.TransformationMatrices

### Functions for transformation matrices

#### Information

Package **Frames.TransformationMatrices** contains type definitions and functions to transform rotational frame quantities using transformation matrices.

#### Content

In the table below an example is given for every function definition. The used variables have the following declaration:

```
Orientation T, T1, T2, T_rel, T_inv;
Real[3]      v1, v2, w1, w2, n_x, n_y, n_z, e, e_x, res_ori, phi;
Real[6]      res_equal;
Real         L, angle;
```

Function/type	Description
<b>Orientation T;</b>	New type defining an orientation object that describes the rotation of frame 1 into frame 2.
<b>der_Orientation der_T;</b>	New type defining the first time derivative of Frames.Orientation.
<b>res_ori = orientationConstraint(T);</b>	Return the constraints between the variables of an orientation object (shall be zero).
<b>w1 = angularVelocity1(T, der_T);</b>	Return angular velocity resolved in frame 1 from orientation object T

	and its derivative der_T.
w2 = <b>angularVelocity2</b> (T, der_T);	Return angular velocity resolved in frame 2 from orientation object T and its derivative der_T.
v1 = <b>resolve1</b> (T,v2);	Transform vector v2 from frame 2 to frame 1.
v2 = <b>resolve2</b> (T,v1);	Transform vector v1 from frame 1 to frame 2.
[v1,w1] = <b>multipleResolve1</b> (T, [v2,w2]);	Transform several vectors from frame 2 to frame 1.
[v2,w2] = <b>multipleResolve2</b> (T, [v1,w1]);	Transform several vectors from frame 1 to frame 2.
D1 = <b>resolveDyade1</b> (T,D2);	Transform second order tensor D2 from frame 2 to frame 1.
D2 = <b>resolveDyade2</b> (T,D1);	Transform second order tensor D1 from frame 1 to frame 2.
T= <b>nullRotation</b> ()	Return orientation object T that does not rotate a frame.
T_inv = <b>inverseRotation</b> (T);	Return inverse orientation object.
T_rel = <b>relativeRotation</b> (T1,T2);	Return relative orientation object from two absolute orientation objects.
T2 = <b>absoluteRotation</b> (T1,T_rel);	Return absolute orientation object from another absolute and a relative orientation object.
T = <b>planarRotation</b> (e, angle);	Return orientation object of a planar rotation.
angle = <b>planarRotationAngle</b> (e, v1, v2);	Return angle of a planar rotation, given the rotation axis and the representations of a vector in frame 1 and frame 2.
T = <b>axisRotation</b> (i, angle);	Return orientation object T for rotation around axis i of frame 1.
T = <b>axesRotations</b> (sequence, angles);	Return rotation object to rotate in sequence around 3 axes. Example: T = axesRotations({1,2,3},{90,45,-90});
angles = <b>axesRotationsAngles</b> (T, sequence);	Return the 3 angles to rotate in sequence around 3 axes to construct the given orientation object.
phi = <b>smallRotation</b> (T);	Return rotation angles phi valid for a small rotation.
T = <b>from_nxy</b> (n_x, n_y);	Return orientation object from n_x and n_y vectors.
T = <b>from_nxz</b> (n_x, n_z);	Return orientation object from n_x and n_z vectors.
R = <b>from_T</b> (T);	Return orientation object R from transformation matrix T.
R = <b>from_T_inv</b> (T_inv);	Return orientation object R from inverse transformation matrix T_inv.
T = <b>from_Q</b> (Q);	Return orientation object T from quaternion orientation object Q.
T = <b>to_T</b> (R);	Return transformation matrix T from orientation object R.
T_inv = <b>to_T_inv</b> (R);	Return inverse transformation matrix T_inv from orientation object R.
Q = <b>to_Q</b> (T);	Return quaternion orientation object Q from orientation object T.
exy = <b>to_exy</b> (T);	Return [e_x, e_y] matrix of an orientation object T, with e_x and e_y vectors of frame 2, resolved in frame 1.

## Package Content

Name	Description
Orientation	Orientation type defining rotation from a frame 1 into a frame 2 with a transformation matrix
der_Orientation	New type defining the first time derivative of Orientation
 <b>orientationConstraint</b>	Return residues of orientation constraints (shall be zero)
 <b>angularVelocity1</b>	Return angular velocity resolved in frame 1 from orientation object and its derivative

(f) <code>angularVelocity2</code>	Return angular velocity resolved in frame 2 from orientation object and its derivative
(f) <code>resolve1</code>	Transform vector from frame 2 to frame 1
(f) <code>resolve2</code>	Transform vector from frame 1 to frame 2
(f) <code>multipleResolve1</code>	Transform several vectors from frame 2 to frame 1
(f) <code>multipleResolve2</code>	Transform several vectors from frame 1 to frame 2
(f) <code>resolveDyade1</code>	Transform second order tensor from frame 2 to frame 1
(f) <code>resolveDyade2</code>	Transform second order tensor from frame 1 to frame 2
(f) <code>nullRotation</code>	Return orientation object that does not rotate a frame
(f) <code>inverseRotation</code>	Return inverse orientation object
(f) <code>relativeRotation</code>	Return relative orientation object
(f) <code>absoluteRotation</code>	Return absolute orientation object from another absolute and a relative orientation object
(f) <code>planarRotation</code>	Return orientation object of a planar rotation
(f) <code>planarRotationAngle</code>	Return angle of a planar rotation, given the rotation axis and the representations of a vector in frame 1 and frame 2
(f) <code>axisRotation</code>	Return rotation object to rotate around one frame axis
(f) <code>axesRotations</code>	Return rotation object to rotate in sequence around 3 axes
(f) <code>axesRotationsAngles</code>	Return the 3 angles to rotate in sequence around 3 axes to construct the given orientation object
(f) <code>smallRotation</code>	Return rotation angles valid for a small rotation and optionally residues that should be zero
(f) <code>from_nxy</code>	Return orientation object from n_x and n_y vectors
(f) <code>from_nxz</code>	Return orientation object from n_x and n_z vectors
(f) <code>from_T</code>	Return orientation object R from transformation matrix T
(f) <code>from_T_inv</code>	Return orientation object R from inverse transformation matrix T_inv
(f) <code>from_Q</code>	Return orientation object T from quaternion orientation object Q
(f) <code>to_T</code>	Return transformation matrix T from orientation object R
(f) <code>to_T_inv</code>	Return inverse transformation matrix T_inv from orientation object R
(f) <code>to_Q</code>	Return quaternion orientation object Q from orientation object T
(f) <code>to_vector</code>	Map rotation object into vector
(f) <code>to_exy</code>	Map rotation object into e_x and e_y vectors of frame 2, resolved in frame 1

## Types and constants

```

type Orientation
  "Orientation type defining rotation from a frame 1 into a frame 2 with a
  transformation matrix"

  extends Internal.TransformationMatrix;

  encapsulated function equalityConstraint
    "Return the constraint residues to express that two frames have the same
    orientation"

```

```
import Modelica;
import Modelica.Mechanics.MultiBody.Frames.TransformationMatrices;
extends Modelica.Icons.Function;
input TransformationMatrices.Orientation T1
"Orientation object to rotate frame 0 into frame 1";
input TransformationMatrices.Orientation T2
"Orientation object to rotate frame 0 into frame 2";
output Real residue[3]
"The rotation angles around x-, y-, and z-axis of frame 1 to rotate frame
1 into frame 2 for a small rotation (should be zero)";
algorithm
residue := {cross(T1[1, :], T1[2, :])*T2[2, :], -cross(T1[1, :], T1[2, :])
*T2[1, :], T1[2, :]*T2[1, :]};
end equalityConstraint;
end Orientation;

type der_Orientation = Real[3, 3] (each unit="1/s")
"New type defining the first time derivative of Orientation";
```

---

## Modelica.Mechanics.MultiBody.Frames.TransformationMatrices.Orientation

Orientation type defining rotation from a frame 1 into a frame 2 with a transformation matrix

### Information

This type describes the **rotation** from a **frame 1** into a **frame 2**. An instance **R** of type **Orientation** has the following interpretation:

$$\mathbf{T} = [\mathbf{e}_x, \mathbf{e}_y, \mathbf{e}_z]; \\ \text{e.g., } \mathbf{T} = [1, 0, 0; 0, 1, 0; 0, 0, 1]$$

where  $\mathbf{e}_x, \mathbf{e}_y, \mathbf{e}_z$  are unit vectors in the direction of the x-axis, y-axis, and z-axis of frame 1, resolved in frame 2, respectively. Therefore, if  $\mathbf{v}_1$  is vector  $\mathbf{v}$  resolved in frame 1 and  $\mathbf{v}_2$  is vector  $\mathbf{v}$  resolved in frame 2, the following relationship holds:

$$\mathbf{v}_2 = \mathbf{T} * \mathbf{v}_1$$

The **inverse** orientation  $\mathbf{T}_{\text{inv}} = \mathbf{T}^T$  describes the rotation from frame 2 into frame 1.

Since the orientation is described by 9 variables, there are 6 constraints between these variables. These constraints are defined in function **TransformationMatrices.orientationConstraint**.

Note, that in the MultiBody library the rotation object is never directly accessed but only with the access functions provided in package TransformationMatrices. As a consequence, other implementations of Rotation can be defined by adapting this package correspondingly.

---

## Modelica.Mechanics.MultiBody.Frames.TransformationMatrices.der\_Orientation

New type defining the first time derivative of Orientation

---

**Modelica.Mechanics.MultiBody.Frames.TransformationMatrices.orientationConstraint**

Return residues of orientation constraints (shall be zero)

**Inputs**

Name	Default	Description
T		Orientation object to rotate frame 1 into frame 2

**Outputs**

Name	Description
residue[6]	Residues of constraints between elements of orientation object (shall be zero)

**Modelica.Mechanics.MultiBody.Frames.TransformationMatrices.angularVelocity1**

Return angular velocity resolved in frame 1 from orientation object and its derivative

**Inputs**

Name	Default	Description
T		Orientation object to rotate frame 1 into frame 2
der_T		Derivative of T [1/s]

**Outputs**

Name	Description
w[3]	Angular velocity of frame 2 with respect to frame 1 resolved in frame 1 [rad/s]

**Modelica.Mechanics.MultiBody.Frames.TransformationMatrices.angularVelocity2**

Return angular velocity resolved in frame 2 from orientation object and its derivative

**Inputs**

Name	Default	Description
T		Orientation object to rotate frame 1 into frame 2
der_T		Derivative of T [1/s]

**Outputs**

Name	Description
w[3]	Angular velocity of frame 2 with respect to frame 1 resolved in frame 2 [rad/s]

**Modelica.Mechanics.MultiBody.Frames.TransformationMatrices.resolve1**

Transform vector from frame 2 to frame 1

---

## 490 Modelica.Mechanics.MultiBody.Frames.TransformationMatrices.resolve1

---

### Inputs

Name	Default	Description
T		Orientation object to rotate frame 1 into frame 2
v2[3]		Vector in frame 2

### Outputs

Name	Description
v1[3]	Vector in frame 1

---

## Modelica.Mechanics.MultiBody.Frames.TransformationMatrices.resolve2



Transform vector from frame 1 to frame 2

### Inputs

Name	Default	Description
T		Orientation object to rotate frame 1 into frame 2
v1[3]		Vector in frame 1

### Outputs

Name	Description
v2[3]	Vector in frame 2

---

## Modelica.Mechanics.MultiBody.Frames.TransformationMatrices.multipleResolve1



Transform several vectors from frame 2 to frame 1

### Inputs

Name	Default	Description
T		Orientation object to rotate frame 1 into frame 2
v2[3, :]		Vectors in frame 2

### Outputs

Name	Description
v1[3, size(v2, 2)]	Vectors in frame 1

---

## Modelica.Mechanics.MultiBody.Frames.TransformationMatrices.multipleResolve2



Transform several vectors from frame 1 to frame 2

### Inputs

Name	Default	Description
T		Orientation object to rotate frame 1 into frame 2
v1[3, :]		Vectors in frame 1

## Outputs

Name	Description
v2[3, size(v1, 2)]	Vectors in frame 2

---

## Modelica.Mechanics.MultiBody.Frames.TransformationMatrices.resolveDyade1

Transform second order tensor from frame 2 to frame 1



## Inputs

Name	Default	Description
T		Orientation object to rotate frame 1 into frame 2
D2[3, 3]		Second order tensor resolved in frame 2

## Outputs

Name	Description
D1[3, 3]	Second order tensor resolved in frame 1

---

## Modelica.Mechanics.MultiBody.Frames.TransformationMatrices.resolveDyade2

Transform second order tensor from frame 1 to frame 2



## Inputs

Name	Default	Description
T		Orientation object to rotate frame 1 into frame 2
D1[3, 3]		Second order tensor resolved in frame 1

## Outputs

Name	Description
D2[3, 3]	Second order tensor resolved in frame 2

---

## Modelica.Mechanics.MultiBody.Frames.TransformationMatrices.nullRotation

Return orientation object that does not rotate a frame



## Outputs

Name	Description
T	Orientation object such that frame 1 and frame 2 are identical

---

## Modelica.Mechanics.MultiBody.Frames.TransformationMatrices.inverseRotation

Return inverse orientation object



## Inputs

Name	Default	Description
T		Orientation object to rotate frame 1 into frame 2

## Outputs

Name	Description
T_inv	Orientation object to rotate frame 2 into frame 1

## Modelica.Mechanics.MultiBody.Frames.TransformationMatrices.relativeRotation

Return relative orientation object



## Inputs

Name	Default	Description
T1		Orientation object to rotate frame 0 into frame 1
T2		Orientation object to rotate frame 0 into frame 2

## Outputs

Name	Description
T_rel	Orientation object to rotate frame 1 into frame 2

## Modelica.Mechanics.MultiBody.Frames.TransformationMatrices.absoluteRotation

Return absolute orientation object from another absolute and a relative orientation object



## Inputs

Name	Default	Description
T1		Orientation object to rotate frame 0 into frame 1
T_rel		Orientation object to rotate frame 1 into frame 2

## Outputs

Name	Description
T2	Orientation object to rotate frame 0 into frame 2

## Modelica.Mechanics.MultiBody.Frames.TransformationMatrices.planarRotation

Return orientation object of a planar rotation



## Inputs

Name	Default	Description
e[3]		Normalized axis of rotation (must have length=1)
angle		Rotation angle to rotate frame 1 into frame 2 along axis e [rad]

## Outputs

Name	Description
T	Orientation object to rotate frame 1 into frame 2

## Modelica.Mechanics.MultiBody.Frames.TransformationMatrices.planarRotationAngle

Return angle of a planar rotation, given the rotation axis and the representations of a vector in frame 1 and frame 2



## Information

A call to this function of the form

```
Real[3] e, v1, v2;
Modelica.SIunits.Angle angle;
equation
  angle = planarRotationAngle(e, v1, v2);
```

computes the rotation angle "**angle**" of a planar rotation along unit vector **e**, rotating frame 1 into frame 2, given the coordinate representations of a vector "v" in frame 1 (**v1**) and in frame 2 (**v2**). Therefore, the result of this function fulfills the following equation:

$$v2 = \text{resolve2}(\text{planarRotation}(e, \text{angle}), v1)$$

The rotation angle is returned in the range

$$-\pi \leq \text{angle} \leq \pi$$

This function makes the following assumptions on the input arguments

- Vector **e** has length 1, i.e.,  $\text{length}(e) = 1$
- Vector "v" is not parallel to **e**, i.e.,  $\text{length}(\text{cross}(e, v)) \neq 0$

The function does not check the above assumptions. If these assumptions are violated, a wrong result will be returned and/or a division by zero will occur.

## Inputs

Name	Default	Description
e[3]		Normalized axis of rotation to rotate frame 1 around e into frame 2 (must have length=1)
v1[3]		A vector v resolved in frame 1 (shall not be parallel to e)
v2[3]		Vector v resolved in frame 2, i.e., $v2 = \text{resolve2}(\text{planarRotation}(e, \text{angle}), v1)$

## Outputs

Name	Description
angle	Rotation angle to rotate frame 1 into frame 2 along axis e in the range: $-\pi \leq \text{angle} \leq \pi$ [rad]

## Modelica.Mechanics.MultiBody.Frames.TransformationMatrices.axisRotation

Return rotation object to rotate around one frame axis



## Inputs

Name	Default	Description
axis		Rotate around 'axis' of frame 1
angle		Rotation angle to rotate frame 1 into frame 2 along 'axis' of frame 1 [rad]

## Outputs

Name	Description
T	Orientation object to rotate frame 1 into frame 2

## Modelica.Mechanics.MultiBody.Frames.TransformationMatrices.axesRotations



Return rotation object to rotate in sequence around 3 axes

## Inputs

Name	Default	Description
sequence[3]	{1,2,3}	Sequence of rotations from frame 1 to frame 2 along axis sequence[i]
angles[3]	{0,0,0}	Rotation angles around the axes defined in 'sequence' [rad]

## Outputs

Name	Description
T	Orientation object to rotate frame 1 into frame 2

## Modelica.Mechanics.MultiBody.Frames.TransformationMatrices.axesRotationsAngles



Return the 3 angles to rotate in sequence around 3 axes to construct the given orientation object

## Information

A call to this function of the form

```
TransformationMatrices.Orientation      T;
parameter Integer      sequence[3] = {1,2,3};
Modelica.SIunits.Angle angles[3];
equation
  angle = axesRotationAngles(T, sequence);
```

computes the rotation angles "**angles[1:3]**" to rotate frame 1 into frame 2 along axes **sequence[1:3]**, given the orientation object **T** from frame 1 to frame 2. Therefore, the result of this function fulfills the following equation:

```
T = axesRotation(sequence, angles)
```

The rotation angles are returned in the range

```
-π <= angles[i] <= π
```

There are **two solutions** for "angles[1]" in this range. Via the third argument **guessAngle1** (default = 0) the returned solution is selected such that |angles[1] - guessAngle1| is minimal. The orientation object **T** may be

in a singular configuration, i.e., there is an infinite number of angle values leading to the same T. The returned solution is selected by setting angles[1] = guessAngle1. Then angles[2] and angles[3] can be uniquely determined in the above range.

Note, that input argument **sequence** has the restriction that only values 1,2,3 can be used and that sequence[1] ≠ sequence[2] and sequence[2] ≠ sequence[3]. Often used values are:

```
sequence = {1,2,3} // Cardan angle sequence
= {3,1,3} // Euler angle sequence
= {3,2,1} // Tait-Bryan angle sequence
```

## Inputs

Name	Default	Description
T		Orientation object to rotate frame 1 into frame 2
sequence[3]	{1,2,3}	Sequence of rotations from frame 1 to frame 2 along axis sequence[i]
guessAngle1	0	Select angles[1] such that  angles[1] - guessAngle1  is a minimum [rad]

## Outputs

Name	Description
angles[3]	Rotation angles around the axes defined in 'sequence' such that T=TransformationMatrices.axesRotation(sequence,angles); -pi < angles[i] <= pi [rad]

---

## Modelica.Mechanics.MultiBody.Frames.TransformationMatrices.smallRotation

Return rotation angles valid for a small rotation and optionally residues that should be zero



## Inputs

Name	Default	Description
T		Orientation object to rotate frame 1 into frame 2
withResidues	false	= false/true, if 'angles'/angles and residues' are returned in phi

## Outputs

Name	Description
phi[if withResidues then 6 else 3]	The rotation angles around x-, y-, and z-axis of frame 1 to rotate frame 1 into frame 2 for a small rotation + optionally 3 residues that should be zero [rad]

---

## Modelica.Mechanics.MultiBody.Frames.TransformationMatrices.from\_nxy

Return orientation object from n\_x and n\_y vectors



## Information

It is assumed that the two input vectors n\_x and n\_y are resolved in frame 1 and are directed along the x and y axis of frame 2 (i.e., n\_x and n\_y are orthogonal to each other) The function returns the orientation object T to rotate from frame 1 to frame 2.

The function is robust in the sense that it returns always an orientation object T, even if n\_y is not orthogonal to n\_x. This is performed in the following way:

## 496 Modelica.Mechanics.MultiBody.Frames.TransformationMatrices.from\_nxy

---

If  $n_x$  and  $n_y$  are not orthogonal to each other, first a unit vector  $e_y$  is determined that is orthogonal to  $n_x$  and is lying in the plane spanned by  $n_x$  and  $n_y$ . If  $n_x$  and  $n_y$  are parallel or nearly parallel to each other, a vector  $e_y$  is selected arbitrarily such that  $e_x$  and  $e_y$  are orthogonal to each other.

### Inputs

Name	Default	Description
$n_x[3]$		Vector in direction of x-axis of frame 2, resolved in frame 1
$n_y[3]$		Vector in direction of y-axis of frame 2, resolved in frame 1

### Outputs

Name	Description
T	Orientation object to rotate frame 1 into frame 2

---

## Modelica.Mechanics.MultiBody.Frames.TransformationMatrices.from\_nxz



Return orientation object from  $n_x$  and  $n_z$  vectors

### Information

It is assumed that the two input vectors  $n_x$  and  $n_z$  are resolved in frame 1 and are directed along the x and z axis of frame 2 (i.e.,  $n_x$  and  $n_z$  are orthogonal to each other) The function returns the orientation object T to rotate from frame 1 to frame 2.

The function is robust in the sense that it returns always an orientation object T, even if  $n_z$  is not orthogonal to  $n_x$ . This is performed in the following way:

If  $n_x$  and  $n_z$  are not orthogonal to each other, first a unit vector  $e_z$  is determined that is orthogonal to  $n_x$  and is lying in the plane spanned by  $n_x$  and  $n_z$ . If  $n_x$  and  $n_z$  are parallel or nearly parallel to each other, a vector  $e_z$  is selected arbitrarily such that  $n_x$  and  $e_z$  are orthogonal to each other.

### Inputs

Name	Default	Description
$n_x[3]$		Vector in direction of x-axis of frame 2, resolved in frame 1
$n_z[3]$		Vector in direction of z-axis of frame 2, resolved in frame 1

### Outputs

Name	Description
T	Orientation object to rotate frame 1 into frame 2

---

## Modelica.Mechanics.MultiBody.Frames.TransformationMatrices.from\_T



Return orientation object R from transformation matrix T

### Inputs

Name	Default	Description
$T[3, 3]$		Transformation matrix to transform vector from frame 1 to frame 2 ( $v2 = T \cdot v1$ )

## Outputs

Name	Description
R	Orientation object to rotate frame 1 into frame 2

---

## Modelica.Mechanics.MultiBody.Frames.TransformationMatrices.from\_T\_inv

Return orientation object R from inverse transformation matrix T\_inv



## Inputs

Name	Default	Description
T_inv[3, 3]		Inverse transformation matrix to transform vector from frame 2 to frame 1 ( $v1=T\_inv*v2$ )

## Outputs

Name	Description
R	Orientation object to rotate frame 1 into frame 2

---

## Modelica.Mechanics.MultiBody.Frames.TransformationMatrices.from\_Q

Return orientation object T from quaternion orientation object Q



## Inputs

Name	Default	Description
Q		Quaternions orientation object to rotate frame 1 into frame 2

## Outputs

Name	Description
T	Orientation object to rotate frame 1 into frame 2

---

## Modelica.Mechanics.MultiBody.Frames.TransformationMatrices.to\_T

Return transformation matrix T from orientation object R



## Inputs

Name	Default	Description
R		Orientation object to rotate frame 1 into frame 2

## Outputs

Name	Description
T[3, 3]	Transformation matrix to transform vector from frame 1 to frame 2 ( $v2=T*v1$ )

---

## Modelica.Mechanics.MultiBody.Frames.TransformationMatrices.to\_T\_inv

Return inverse transformation matrix T\_inv from orientation object R



---

**498 Modelica.Mechanics.MultiBody.Frames.TransformationMatrices.to\_T\_inv****Inputs**

Name	Default	Description
R		Orientation object to rotate frame 1 into frame 2

**Outputs**

Name	Description
T_inv[3, 3]	Inverse transformation matrix to transform vector from frame 2 into frame 1 ( $v1=T\_inv*v2$ )

---

**Modelica.Mechanics.MultiBody.Frames.TransformationMatrices.to\_Q**

Return quaternion orientation object Q from orientation object T

**Inputs**

Name	Default	Description
T		Orientation object to rotate frame 1 into frame 2
Q_guess	Quaternions.nullRotation() ()	Guess value for output Q (there are 2 solutions; the one closer to Q_guess is used)

**Outputs**

Name	Description
Q	Quaternions orientation object to rotate frame 1 into frame 2

---

**Modelica.Mechanics.MultiBody.Frames.TransformationMatrices.to\_vector**

Map rotation object into vector

**Inputs**

Name	Default	Description
T		Orientation object to rotate frame 1 into frame 2

**Outputs**

Name	Description
vec[9]	Elements of T in one vector

---

**Modelica.Mechanics.MultiBody.Frames.TransformationMatrices.to\_exy**

Map rotation object into e\_x and e\_y vectors of frame 2, resolved in frame 1

**Inputs**

Name	Default	Description
T		Orientation object to rotate frame 1 into frame 2

## Outputs

Name	Description
exy[3, 2]	= [e_x, e_y] where e_x and e_y are axes unit vectors of frame 2, resolved in frame 1

---

## Modelica.Mechanics.MultiBody.Interfaces

Connectors and partial models for 3-dim. mechanical components

## Information

This package contains connectors and partial models (i.e. models that are only used to build other models) of the MultiBody library.

## Package Content

Name	Description
 Frame	Coordinate system fixed to the component with one cut-force and cut-torque (no icon)
 Frame_a	Coordinate system fixed to the component with one cut-force and cut-torque (filled rectangular icon)
 Frame_b	Coordinate system fixed to the component with one cut-force and cut-torque (non-filled rectangular icon)
 Frame_resolve	Coordinate system fixed to the component used to express in which coordinate system a vector is resolved (non-filled rectangular icon)
 FlangeWithBearing	Connector consisting of 1-dim. rotational flange and its bearing frame
 FlangeWithBearingAdaptor	Adaptor to allow direct connections to the sub-connectors of FlangeWithBearing
 . PartialTwoFrames	Base model for components providing two frame connectors + outer world + assert to guarantee that the component is connected
 . PartialTwoFramesDoubleSize	Base model for components providing two frame connectors + outer world + assert to guarantee that the component is connected (default icon size is factor 2 larger as usual)
 . PartialOneFrame_a	Base model for components providing one frame_a connector + outer world + assert to guarantee that the component is connected
 . PartialOneFrame_b	Base model for components providing one frame_b connector + outer world + assert to guarantee that the component is connected
 . PartialElementaryJoint	Base model for elementary joints (has two frames + outer world + assert to guarantee that the joint is connected)
 . PartialForce	Base model for force elements (provide frame_b.f and frame_b.t in subclasses)
 . PartialLineForce	Base model for line force elements
 . PartialAbsoluteSensor	Base model to measure an absolute frame variable
 . PartialRelativeSensor	Base model to measure a relative variable between two frames
 . PartialCutForceSensor	Base model to measure the cut force and/or torque between two frames
 . PartialVisualizer	Base model for visualizers (has a frame_a on the left side + outer world + assert to guarantee that the component is connected)

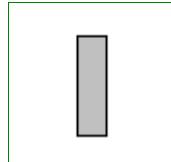
**Modelica.Mechanics.MultiBody.Interfaces.Frame****Coordinate system fixed to the component with one cut-force and cut-torque (no icon)****Information**

Basic definition of a coordinate system that is fixed to a mechanical component. In the origin of the coordinate system the cut-force and the cut-torque is acting. This component has no icon definition and is only used by inheritance from frame connectors to define different icons.

**Contents**

Name	Description
r_0[3]	Position vector from world frame to the connector frame origin, resolved in world frame [m]
R	Orientation object to rotate the world frame into the connector frame
f[3]	Cut-force resolved in connector frame [N]
t[3]	Cut-torque resolved in connector frame [N.m]

---

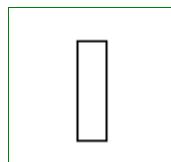
**Modelica.Mechanics.MultiBody.Interfaces.Frame\_a****Coordinate system fixed to the component with one cut-force and cut-torque (filled rectangular icon)****Information**

Basic definition of a coordinate system that is fixed to a mechanical component. In the origin of the coordinate system the cut-force and the cut-torque is acting. This component has a filled rectangular icon.

**Contents**

Name	Description
r_0[3]	Position vector from world frame to the connector frame origin, resolved in world frame [m]
R	Orientation object to rotate the world frame into the connector frame
f[3]	Cut-force resolved in connector frame [N]
t[3]	Cut-torque resolved in connector frame [N.m]

---

**Modelica.Mechanics.MultiBody.Interfaces.Frame\_b****Coordinate system fixed to the component with one cut-force and cut-torque (non-filled rectangular icon)****Information**

Basic definition of a coordinate system that is fixed to a mechanical component. In the origin of the coordinate system the cut-force and the cut-torque is acting. This component has a non-filled rectangular icon.

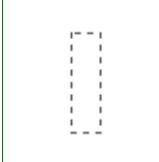
**Contents**

Name	Description
r_0[3]	Position vector from world frame to the connector frame origin, resolved in world frame [m]

R	Orientation object to rotate the world frame into the connector frame
f[3]	Cut-force resolved in connector frame [N]
t[3]	Cut-torque resolved in connector frame [N.m]

## Modelica.Mechanics.MultiBody.Interfaces.Frame\_resolve

Coordinate system fixed to the component used to express in which coordinate system a vector is resolved (non-filled rectangular icon)



### Information

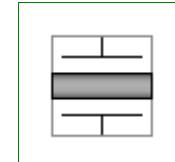
Basic definition of a coordinate system that is fixed to a mechanical component. In the origin of the coordinate system the cut-force and the cut-torque is acting. This coordinate system is used to express in which coordinate system a vector is resolved. A component that uses a Frame\_resolve connector has to set the cut-force and cut-torque of this frame to zero. When connecting from a Frame\_resolve connector to another frame connector, by default the connecting line has line style "dotted". This component has a non-filled rectangular icon.

### Contents

Name	Description
r_0[3]	Position vector from world frame to the connector frame origin, resolved in world frame [m]
R	Orientation object to rotate the world frame into the connector frame
f[3]	Cut-force resolved in connector frame [N]
t[3]	Cut-torque resolved in connector frame [N.m]

## Modelica.Mechanics.MultiBody.Interfaces.FlangeWithBearing

Connector consisting of 1-dim. rotational flange and its bearing frame



### Information

This hierarchical connector models a 1-dim. rotational flange connector and its optional bearing defined by a 3-dim. frame connector. If a connection to the subconnectors should be clearly visible, connect first an instance of [FlangeWithBearingAdaptor](#) to the FlangeWithBearing connector.

### Parameters

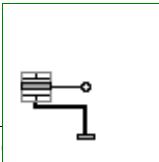
Name	Default	Description
includeBearingConnector	false	= true, if bearing frame connector is present, otherwise not present

### Contents

Name	Description
includeBearingConnector	= true, if bearing frame connector is present, otherwise not present
flange	1-dim. rotational flange
bearingFrame	3-dim. frame in which the 1-dim. shaft is mounted

## Modelica.Mechanics.MultiBody.Interfaces.FlangeWithBearingAdaptor

Adaptor to allow direct connections to the sub-connectors of FlangeWithBearing



## Information

Adaptor object to make a more visible connection to the flange and frame subconnectors of a FlangeWithBearing connector.

## Parameters

Name	Default	Description
includeBearingConnector	false	= true, if bearing frame connector is present, otherwise not present

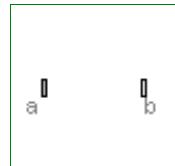
## Connectors

Name	Description
flangeAndFrame	Compound connector consisting of 1-dim. rotational flange and 3-dim. frame mounting
flange	1-dim. rotational flange
frame	3-dim. frame in which the 1-dim. shaft is mounted

---

## Modelica.Mechanics.MultiBody.Interfaces.PartialTwoFrames

**Base model for components providing two frame connectors + outer world + assert to guarantee that the component is connected**



## Information

This partial model provides two frame connectors, access to the world object and an assert to check that both frame connectors are connected. Therefore, inherit from this partial model if the two frame connectors are needed and if the two frame connectors should be connected for a correct model.

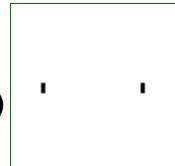
## Connectors

Name	Description
frame_a	Coordinate system fixed to the component with one cut-force and cut-torque
frame_b	Coordinate system fixed to the component with one cut-force and cut-torque

---

## Modelica.Mechanics.MultiBody.Interfaces.PartialTwoFramesDoubleSize

**Base model for components providing two frame connectors + outer world + assert to guarantee that the component is connected (default icon size is factor 2 larger as usual)**



## Information

This partial model provides two frame connectors, access to the world object and an assert to check that both frame connectors are connected. Therefore, inherit from this partial model if the two frame connectors are needed and if the two frame connectors should be connected for a correct model.

When dragging "PartialTwoFrames", the default size is a factor of two larger as usual. This partial model is used by the Joint.Assemblies joint aggregation models.

## Connectors

Name	Description
frame_a	Coordinate system fixed to the component with one cut-force and cut-torque
frame_b	Coordinate system fixed to the component with one cut-force and cut-torque

**Modelica.Mechanics.MultiBody.Interfaces.PartialOneFrame\_a**

**Base model for components providing one frame\_a connector + outer world + assert to guarantee that the component is connected**

**Information**

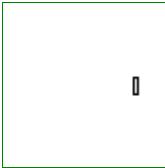
This partial model provides one frame\_a connector, access to the world object and an assert to check that the frame\_a connector is connected. Therefore, inherit from this partial model if the frame\_a connector is needed and if this connector should be connected for a correct model.

**Connectors**

Name	Description
frame_a	Coordinate system fixed to the component with one cut-force and cut-torque

**Modelica.Mechanics.MultiBody.Interfaces.PartialOneFrame\_b**

**Base model for components providing one frame\_b connector + outer world + assert to guarantee that the component is connected**

**Information**

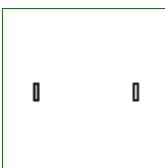
This partial model provides one frame\_b connector, access to the world object and an assert to check that the frame\_b connector is connected. Therefore, inherit from this partial model if the frame\_b connector is needed and if this connector should be connected for a correct model.

**Connectors**

Name	Description
frame_b	Coordinate system fixed to the component with one cut-force and cut-torque

**Modelica.Mechanics.MultiBody.Interfaces.PartialElementaryJoint**

**Base model for elementary joints (has two frames + outer world + assert to guarantee that the joint is connected)**

**Information**

All **elementary joints** should inherit from this base model, i.e., joints that are directly defined by equations, provided they compute either the rotation object of frame\_b from the rotation object of frame\_a and from relative quantities (or vice versa), or there is a constraint equation between the rotation objects of the two frames. In other cases, a joint object should inherit from **Interfaces.PartialTwoFrames** (e.g., joint Spherical, because there is no constraint between the rotation objects of frame\_a and frame\_b or joint Cylindrical because it is not an elementary joint).

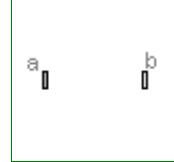
This partial model provides two frame connectors, a "defineBranch" between frame\_a and frame\_b, access to the world object and an assert to check that both frame connectors are connected.

## Connectors

Name	Description
frame_a	Coordinate system fixed to the joint with one cut-force and cut-torque
frame_b	Coordinate system fixed to the joint with one cut-force and cut-torque

## Modelica.Mechanics.MultiBody.Interfaces.PartialForce

Base model for force elements (provide frame\_b.f and frame\_b.t in subclasses)



### Information

All **3-dimensional force and torque elements** should be based on this superclass. This model defines frame\_a and frame\_b, computes the relative translation and rotation between the two frames and calculates the cut-force and cut-torque at frame\_a by a force and torque balance from the cut-force and cut-torque at frame\_b. As a result, in a subclass, only the relationship between the cut-force and cut-torque at frame\_b has to be defined as a function of the following relative quantities:

```
r_rel_b[3]: Position vector from origin of frame_a to origin
            of frame_b, resolved in frame_b
R_rel      : Relative orientation object to rotate from frame_a to frame_b
```

Assume that force  $f = \{100, 0, 0\}$  should be applied on the body to which this force element is attached at frame\_b, then the definition should be:

```
model Constant_x_Force
  extends Modelica.Mechanics.MultiBody.Interfaces.PartialForce;
equation
  frame_b.f = {-100, 0, 0};
  frame_b.t = zeros(3);
end Constant_x_Force;
```

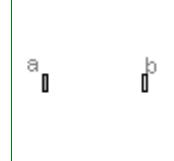
Note, that frame\_b.f and frame\_b.t are flow variables and therefore the negative value of frame\_b.f and frame\_b.t is acting at the part to which this force element is connected.

## Connectors

Name	Description
frame_a	Coordinate system fixed to the joint with one cut-force and cut-torque
frame_b	Coordinate system fixed to the joint with one cut-force and cut-torque

## Modelica.Mechanics.MultiBody.Interfaces.PartialLineForce

Base model for line force elements



### Information

All **line force** elements should be based on this base model. This model defines frame\_a and frame\_b, computes the relative distance  $s$  and provides the force and torque balance of the cut-forces and cut-torques at frame\_a and frame\_b, respectively. In sub-models, only the line force  $f$ , acting at frame\_b on the line from frame\_a to frame\_b, as a function of the relative distance  $s$  and its derivative  $\text{der}(s)$  has to be defined. Example:

```
model Spring
  parameter Real c "spring constant",
```

```

parameter Real s_unstretched "unstretched spring length";
extends Modelica.Mechanics.MultiBody.Interfaces.PartialLineForce;
equation
  f = c*(s-s_unstretched);
end Spring;

```

## Parameters

Name	Default	Description
<b>Advanced</b>		
s_small	1.E-6	Prevent zero-division if relative distance s=0 [m]

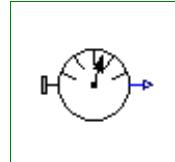
## Connectors

Name	Description
frame_a	Coordinate system fixed to the force element with one cut-force and cut-torque
frame_b	Coordinate system fixed to the force element with one cut-force and cut-torque

---

## Modelica.Mechanics.MultiBody.Interfaces.PartialAbsoluteSensor

Base model to measure an absolute frame variable



## Information

This is the base class of a 3-dim. mechanics component with one frame and one output port in order to measure an absolute quantity in the frame connector and to provide the measured signal as output for further processing with the blocks of package Modelica.Blocks.

## Parameters

Name	Default	Description
n_out	1	Number of output signals

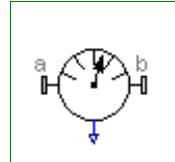
## Connectors

Name	Description
frame_a	Coordinate system from which absolute quantities are provided as output signals
y[n_out]	Measured data as signal vector

---

## Modelica.Mechanics.MultiBody.Interfaces.PartialRelativeSensor

Base model to measure a relative variable between two frames



## Information

This is a base class for 3-dim. mechanical components with two frames and one output port in order to measure relative quantities between the two frames or the cut-forces/torques in the frame and to provide the measured signals as output for further processing with the blocks of package Modelica.Blocks.

## Parameters

Name	Default	Description

## 506 Modelica.Mechanics.MultiBody.Interfaces.PartialRelativeSensor

---

n_out	1	Number of output signals
-------	---	--------------------------

### Connectors

Name	Description
frame_a	Coordinate system a
frame_b	Coordinate system b
y[n_out]	Measured data as signal vector

---

## Modelica.Mechanics.MultiBody.Interfaces.PartialCutForceSensor

Base model to measure the cut force and/or torque between two frames



### Information

This is a base class for 3-dim. mechanical components with two frames and one output port in order to measure the cut-force and/or cut-torque acting between the two frames and to provide the measured signals as output for further processing with the blocks of package Modelica.Blocks.

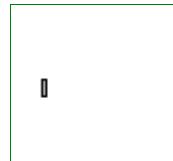
### Connectors

Name	Description
frame_a	Coordinate system with one cut-force and cut-torque
frame_b	Coordinate system with one cut-force and cut-torque
frame_resolve	If connected, the output signals are resolved in this frame (cut-force/-torque are set to zero)

---

## Modelica.Mechanics.MultiBody.Interfaces.PartialVisualizer

Base model for visualizers (has a frame\_a on the left side + outer world + assert to guarantee that the component is connected)



### Information

This partial model provides one frame\_a connector, access to the world object and an assert to check that the frame\_a connector is connected. It is used by inheritance from all visualizer objects.

### Connectors

Name	Description
frame_a	Coordinate system in which visualization data is resolved

---

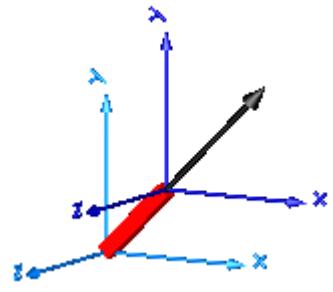
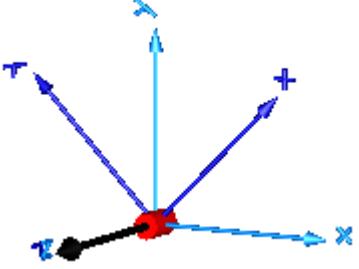
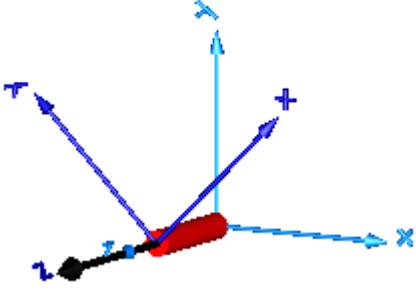
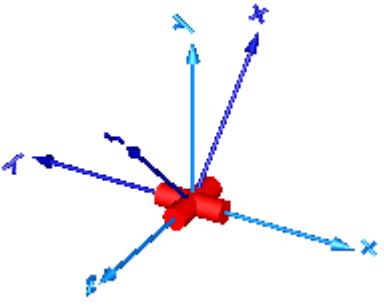
## Modelica.Mechanics.MultiBody.Joints

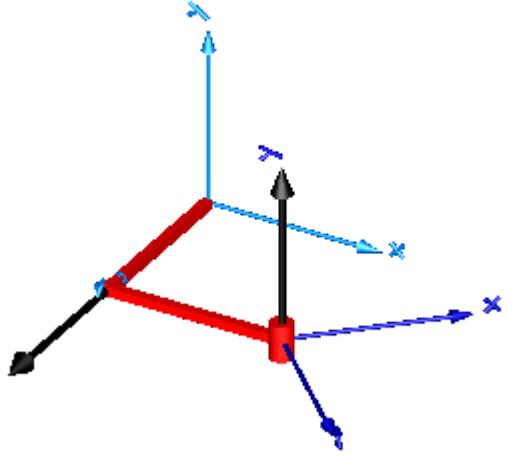
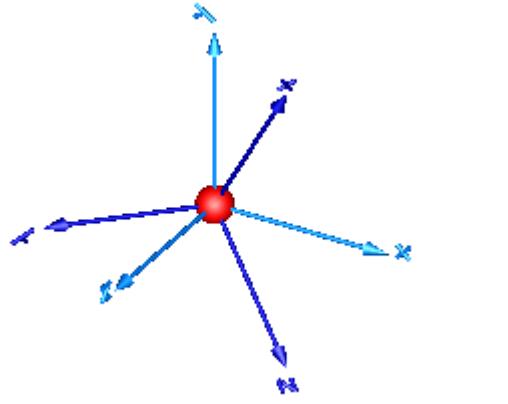
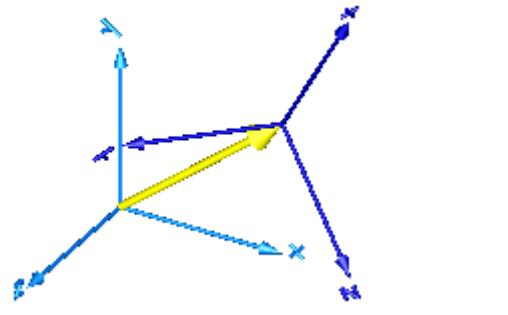
Components that constrain the motion between two frames

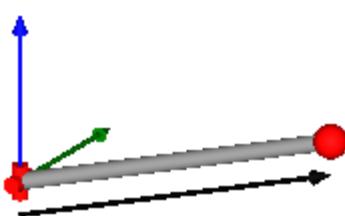
### Information

This package contains **joint components**, that is, idealized, massless elements that constrain the motion between frames. In subpackage **Assemblies** aggregation joint components are provided to handle kinematic loops analytically (this means that non-linear systems of equations occurring in these joint aggregations are analytically solved, i.e., robustly and efficiently).

## Content

<i>Model</i>	<i>Description</i>
<b>Prismatic ActuatedPrismatic</b>	Prismatic joint and actuated prismatic joint (1 translational degree-of-freedom, 2 potential states) 
<b>Revolute ActuatedRevolute</b>	Revolute and actuated revolute joint (1 rotational degree-of-freedom, 2 potential states) 
<b>Cylindrical</b>	Cylindrical joint (2 degrees-of-freedom, 4 potential states) 
<b>Universal</b>	Universal joint (2 degrees-of-freedom, 4 potential states) 
<b>Planar</b>	Planar joint (3 degrees-of-freedom, 6 potential states)

	
<b>Spherical</b>	Spherical joint (3 constraints and no potential states, or 3 degrees-of-freedom and 3 states) 
<b>FreeMotion</b>	Free motion joint (6 degrees-of-freedom, 12 potential states) 
<b>SphericalSpherical</b>	Spherical - spherical joint aggregation (1 constraint, no potential states) with an optional point mass in the middle
<b>UniversalSpherical</b>	Universal - spherical joint aggregation (1 constraint, no potential states)



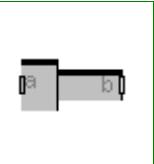
<b>GearConstraint</b>	Ideal 3-dim. gearbox (arbitrary shaft directions)
<b>MultiBody.Joints.Assemblies</b>	Package of joint aggregations for analytic loop handling.

## Package Content

Name	Description
Prismatic	Prismatic joint (1 translational degree-of-freedom, 2 potential states)
ActuatedPrismatic	Actuated prismatic joint (1 translational degree-of-freedom, 2 potential states)
Revolute	Revolute joint (1 rotational degree-of-freedom, 2 potential states)
ActuatedRevolute	Actuated revolute joint (1 rotational degree-of-freedom, 2 potential states)
Cylindrical	Cylindrical joint (2 degrees-of-freedom, 4 potential states)
Universal	Universal joint (2 degrees-of-freedom, 4 potential states)
Planar	Planar joint (3 degrees-of-freedom, 6 potential states)
Spherical	Spherical joint (3 constraints and no potential states, or 3 degrees-of-freedom and 3 states)
FreeMotion	Free motion joint (6 degrees-of-freedom, 12 potential states)
SphericalSpherical	Spherical - spherical joint aggregation (1 constraint, no potential states) with an optional point mass in the middle
UniversalSpherical	Universal - spherical joint aggregation (1 constraint, no potential states)
GearConstraint	Ideal 3-dim. gearbox (arbitrary shaft directions)
Assemblies	Joint aggregations for analytic loop handling

## Modelica.Mechanics.MultiBody.Joints.Prismatic

Prismatic joint (1 translational degree-of-freedom, 2 potential states)

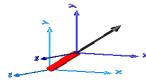


### Information

Joint where frame\_b is translated along axis n which is fixed in frame\_a. The two frames coincide when "s + s\_offset = 0", where "s\_offset" is a parameter with a zero default and "s" is the relative distance.

In the "Advanced" menu it can be defined via parameter **enforceStates** that the relative distance "s" and its derivative shall be definitely used as states (this means that the Modelica attributes stateSelect=StateSelect.always are set on these variables). The states are usually selected automatically. In certain situations, especially when closed kinematic loops are present, it might be slightly more efficient, when using the "enforceStates" setting.

In the following figure the animation of a prismatic joint is shown. The light blue coordinate system is frame\_a and the dark blue coordinate system is frame\_b of the joint. The black arrow is parameter vector "n" defining the translation axis (here:  $n = \{1, 1, 0\}$ ).



## Parameters

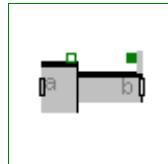
Name	Default	Description
animation	true	= true, if animation shall be enabled
n	{1,0,0}	Axis of translation resolved in frame_a (= same as in frame_b)
s_offset	0	Relative distance offset (distance between frame_a and frame_b = s_offset + s) [m]
<b>Initialization</b>		
initType	Modelica.Mechanics.MultiBody..	Type of initialization (defines usage of start values below)
s_start	0	Initial value of distance (fixed or guess value) [m]
v_start	0	Initial value of relative velocity v = der(s) [m/s]
a_start	0	Initial value of relative acceleration a = der(v) [m/s <sup>2</sup> ]
<b>Animation</b>		
if animation = true		
boxWidthDirection	{0,1,0}	Vector in width direction of box, resolved in frame_a
boxWidth	world.defaultJointWidth	Width of prismatic joint box [m]
boxHeight	boxWidth	Height of prismatic joint box [m]
boxColor	Modelica.Mechanics.MultiBody..	Color of prismatic joint box
specularCoefficient	world.defaultSpecularCoeffic...	Reflection of ambient light (= 0: light is completely absorbed)
<b>Advanced</b>		
enforceStates	false	= true, if generalized variables (s,v) shall be used as states (StateSelect.always)

## Connectors

Name	Description
frame_a	Coordinate system fixed to the joint with one cut-force and cut-torque
frame_b	Coordinate system fixed to the joint with one cut-force and cut-torque

## Modelica.Mechanics.MultiBody.Joints.ActuatedPrismatic

Actuated prismatic joint (1 translational degree-of-freedom, 2 potential states)



## Information

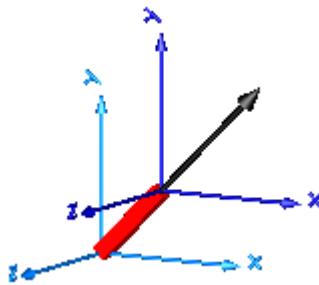
Joint where frame\_b is translated along axis n which is fixed in frame\_a. The two frames coincide when "s + s\_offset = 0", where "s\_offset" is a parameter with a zero default and "s" is the relative distance.

The prismatic joint has two additional 1-dimensional mechanical flanges (flange "axis" represents the driving flange and flange "bearing" represents the bearing) where it can be driven with elements of the Modelica.Mechanics.Translational library.

In the "Advanced" menu it can be defined via parameter **enforceStates** that the relative distance "s" and its derivative shall be definitely used as states (this means that the Modelica attributes stateSelect=StateSelect.always are set on these variables). The states are usually selected automatically. In

certain situations, especially when closed kinematic loops are present, it might be slightly more efficient, when using the "enforceStates" setting.

In the following figure the animation of an actuated prismatic joint is shown. The light blue coordinate system is frame\_a and the dark blue coordinate system is frame\_b of the joint. The black arrow is parameter vector "n" defining the translation axis (here:  $n = \{1,1,0\}$ ).



## Parameters

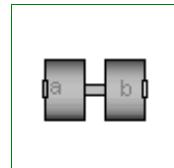
Name	Default	Description
animation	true	= true, if animation shall be enabled
n	{1,0,0}	Axis of translation resolved in frame_a (= same as in frame_b)
s_offset	0	Relative distance offset (distance between frame_a and frame_b = s_offset + s) [m]
<b>Initialization</b>		
initType	Modelica.Mechanics.MultiBody..	Type of initialization (defines usage of start values below)
s_start	0	Initial value of distance (fixed or guess value) [m]
v_start	0	Initial value of relative velocity $v = \text{der}(s)$ [m/s]
a_start	0	Initial value of relative acceleration $a = \text{der}(v)$ [m/s <sup>2</sup> ]
<b>Animation</b>		
if animation = true		
boxWidthDirection	{0,1,0}	Vector in width direction of box, resolved in frame_a
boxWidth	world.defaultJointWidth	Width of prismatic joint box [m]
boxHeight	boxWidth	Height of prismatic joint box [m]
boxColor	Modelica.Mechanics.MultiBody..	Color of prismatic joint box
specularCoefficient	world.defaultSpecularCoeffic...	Reflection of ambient light (= 0: light is completely absorbed)
<b>Advanced</b>		
enforceStates	false	= true, if generalized variables (s,v) shall be used as states (StateSelect.always)

## Connectors

Name	Description
frame_a	Coordinate system fixed to the joint with one cut-force and cut-torque
frame_b	Coordinate system fixed to the joint with one cut-force and cut-torque
axis	1-dim. translational flange that drives the joint
bearing	1-dim. translational flange of the drive bearing

## Modelica.Mechanics.MultiBody.Joints.Revolute

Revolute joint (1 rotational degree-of-freedom, 2 potential states)



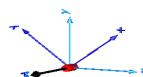
### Information

Joint where frame\_b rotates around axis n which is fixed in frame\_a. The two frames coincide when "phi + phi\_offset = 0", where "phi\_offset" is a parameter with a zero default and "phi" is the rotation angle.

In the "Advanced" menu it can be defined via parameter **enforceStates** that the rotation angle "phi" and its derivative shall be definitely used as states (this means that the Modelica attributes `stateSelect=StateSelect.always` are set on these variables). The states are usually selected automatically. In certain situations, especially when closed kinematic loops are present, it might be slightly more efficient, when using the "enforceStates" setting.

If a **planar loop** is present, e.g., consisting of 4 revolute joints where the joint axes are all parallel to each other, then there is no longer a unique mathematical solution and the symbolic algorithms will fail. Usually, an error message will be printed pointing out this situation. In this case, parameter **planarCutJoint** in the "Advanced" menu of one of the revolute joints has to be set to **true**. The effect is that from the 5 constraints of a usual revolute joint, 3 constraints are removed and replaced by appropriate known variables (e.g., the force in the direction of the axis of rotation is treated as known with value equal to zero; for standard revolute joints, this force is an unknown quantity).

In the following figure the animation of a revolute joint is shown. The light blue coordinate system is frame\_a and the dark blue coordinate system is frame\_b of the joint. The black arrow is parameter vector "n" defining the translation axis (here:  $n = \{0,0,1\}$ ,  $\text{phi\_start} = 45^\circ$ ).



### Parameters

Name	Default	Description
animation	true	= true, if animation shall be enabled (show axis as cylinder)
n	$\{0,0,1\}$	Axis of rotation resolved in frame_a (= same as in frame_b)
phi_offset	0	Relative angle offset (angle = phi + from_deg(phi_offset)) [deg]
<b>Initialization</b>		
initType	Modelica.Mechanics.MultiBody.. .	Type of initialization (defines usage of start values below)
phi_start	0	Initial value of rotation angle phi (fixed or guess value) [deg]
w_start	0	Initial value of relative angular velocity w = der(phi) [deg/s]
a_start	0	Initial value of relative angular acceleration a = der(w) [deg/s <sup>2</sup> ]
<b>Animation</b>		
if animation = true		
cylinderLength	world.defaultJointLength	Length of cylinder representing the joint axis [m]
cylinderDiameter	world.defaultJointWidth	Diameter of cylinder representing the joint axis [m]
cylinderColor	Modelica.Mechanics.MultiBody.. .	Color of cylinder representing the joint axis
specularCoefficient	world.defaultSpecularCoeffic...	Reflection of ambient light (= 0: light is completely

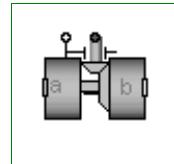
		absorbed)
<b>Advanced</b>		
enforceStates	false	= true, if generalized variables (phi,w) shall be used as states (StateSelect.always)
planarCutJoint	false	= true, if joint shall be used as cut-joint in a planar loop

## Connectors

Name	Description
frame_a	Coordinate system fixed to the joint with one cut-force and cut-torque
frame_b	Coordinate system fixed to the joint with one cut-force and cut-torque

## Modelica.Mechanics.MultiBody.Joints.ActuatedRevolute

Actuated revolute joint (1 rotational degree-of-freedom, 2 potential states)



## Information

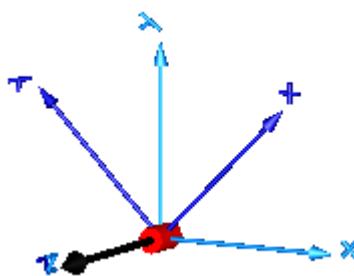
Joint where frame\_b rotates around axis n which is fixed in frame\_a. The two frames coincide when "phi + phi\_offset = 0", where "phi\_offset" is a parameter with a zero default and "phi" is the rotation angle.

The revolute joint has two additional 1-dimensional mechanical flanges (flange "axis" represents the driving flange and flange "bearing" represents the bearing) where it can be driven with elements of the Modelica.Mechanics.Rotational library.

In the "Advanced" menu it can be defined via parameter **enforceStates** that the rotation angle "phi" and its derivative shall be definitely used as states (this means that the Modelica attributes stateSelect=StateSelect.always are set on these variables). The states are usually selected automatically. In certain situations, especially when closed kinematic loops are present, it might be slightly more efficient, when using the "enforceStates" setting.

If a **planar loop** is present, e.g., consisting of 4 revolute joints where the joint axes are all parallel to each other, then there is no longer a unique mathematical solution and the symbolic algorithms will fail. Usually, an error message will be printed pointing out this situation. In this case, parameter **planarCutJoint** in the "Advanced" menu of one of the revolute joints has to be set to **true**. The effect is that from the 5 constraints of a usual revolute joint, 3 constraints are removed and replaced by appropriate known variables (e.g., the force in the direction of the axis of rotation is treated as known with value equal to zero; for standard revolute joints, this force is an unknown quantity).

In the following figure the animation of an actuated revolute joint is shown. The light blue coordinate system is frame\_a and the dark blue coordinate system is frame\_b of the joint. The black arrow is parameter vector "n" defining the translation axis (here:  $n = \{0,0,1\}$ ,  $\text{phi\_start} = 45^\circ$ ).



## Parameters

Name	Default	Description

## 514 Modelica.Mechanics.MultiBody.Joints.ActuatedRevolute

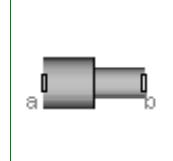
animation	true	= true, if animation shall be enabled (show axis as cylinder)
n	{0,0,1}	Axis of rotation resolved in frame_a (= same as in frame_b)
phi_offset	0	Relative angle offset (angle = phi + from_deg(phi_offset)) [deg]
<b>Initialization</b>		
initType	Modelica.Mechanics.MultiBody..	Type of initialization (defines usage of start values below)
phi_start	0	Initial value of rotation angle phi (fixed or guess value) [deg]
w_start	0	Initial value of relative angular velocity w = der(phi) [deg/s]
a_start	0	Initial value of relative angular acceleration a = der(w) [deg/s <sup>2</sup> ]
<b>Animation</b>		
if animation = true		
cylinderLength	world.defaultJointLength	Length of cylinder representing the joint axis [m]
cylinderDiameter	world.defaultJointWidth	Diameter of cylinder representing the joint axis [m]
cylinderColor	Modelica.Mechanics.MultiBody..	Color of cylinder representing the joint axis
specularCoefficient	world.defaultSpecularCoeffic...	Reflection of ambient light (= 0: light is completely absorbed)
<b>Advanced</b>		
enforceStates	false	= true, if generalized variables (phi,w) shall be used as states (StateSelect.always)
planarCutJoint	false	= true, if joint shall be used as cut-joint in a planar loop

## Connectors

Name	Description
frame_a	Coordinate system fixed to the joint with one cut-force and cut-torque
frame_b	Coordinate system fixed to the joint with one cut-force and cut-torque
axis	1-dim. rotational flange that drives the joint
bearing	1-dim. rotational flange of the drive bearing

## Modelica.Mechanics.MultiBody.Joints.Cylindrical

Cylindrical joint (2 degrees-of-freedom, 4 potential states)



## Information

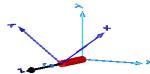
Joint where frame\_b rotates around and translates along axis n which is fixed in frame\_a. The two frames coincide when "revolute.phi=0" and "prismatic.s=0". This joint has the following potential states:

- The relative angle revolute.phi [rad] around axis n,
- the relative distance prismatic.s [m] along axis n,
- the relative angular velocity revolute.w [rad/s] (= der(revolute.phi)) and
- the relative velocity prismatic.v [m/s] (= der(prismatic.s)).

They are used as candidates for automatic selection of states from the tool. This may be enforced by setting "enforceStates=true" in the **Advanced** menu (this means that the Modelica attributes

stateSelect=StateSelect.always are set on these variables). The states are usually selected automatically. In certain situations, especially when closed kinematic loops are present, it might be slightly more efficient, when using the "enforceStates" setting.

In the following figure the animation of a cylindrical joint is shown. The light blue coordinate system is frame\_a and the dark blue coordinate system is frame\_b of the joint. The black arrow is parameter vector "n" defining the cylinder axis (here:  $n = \{0,0,1\}$ ).



## Parameters

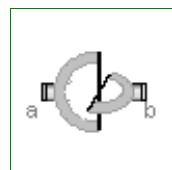
Name	Default	Description
animation	true	= true, if animation shall be enabled (show cylinder)
n	{1,0,0}	Cylinder axis resolved in frame_a (= same as in frame_b)
<b>Initialization</b>		
initType	Modelica.Mechanics.MultiBody...	Type of initialization (defines usage of start values below)
phi_start	0	Initial value of rotation angle phi (fixed or guess value) [deg]
s_start	0	Initial value of relative distance (fixed or guess value) [m]
w_start	0	Initial value of relative angular velocity w = der(phi) [deg/s]
v_start	0	Initial value of relative velocity v = der(s) [m/s]
a_start	0	Initial value of relative acceleration a = der(v) [m/s <sup>2</sup> ]
wd_start	0	Initial value of relative angular acceleration wd = der(w) [deg/s <sup>2</sup> ]
<b>Animation</b>		
if animation = true		
cylinderDiameter	world.defaultJointWidth	Diameter of cylinder [m]
cylinderColor	Modelica.Mechanics.MultiBody...	Color of cylinder
specularCoefficient	world.defaultSpecularCoeffic...	Reflection of ambient light (= 0: light is completely absorbed)
<b>Advanced</b>		
enforceStates	false	= true, if generalized variables shall be used as states (StateSelect.always)

## Connectors

Name	Description
frame_a	Coordinate system fixed to the component with one cut-force and cut-torque
frame_b	Coordinate system fixed to the component with one cut-force and cut-torque

## Modelica.Mechanics.MultiBody.Joints.Universal

Universal joint (2 degrees-of-freedom, 4 potential states)



## Information

Joint where frame\_a rotates around axis n\_a which is fixed in frame\_a and frame\_b rotates around axis n\_b which is fixed in frame\_b. The two frames coincide when "revolute\_a.phi=0" and "revolute\_b.phi=0". This joint has the following potential states;

- The relative angle revolute\_a.phi [rad] around axis n\_a,
- the relative angle revolute\_b.phi [rad] around axis n\_b,
- the relative angular velocity revolute\_a.w [rad/s] (= der(revolute\_a.phi)) and
- the relative angular velocity revolute\_b.w [rad/s] (= der(revolute\_b.phi)).

They are used as candidates for automatic selection of states from the tool. This may be enforced by setting "enforceStates=true" in the **Advanced** menu (this means that the Modelica attributes stateSelect=StateSelect.always are set on these variables). The states are usually selected automatically. In certain situations, especially when closed kinematic loops are present, it might be slightly more efficient, when using the "enforceStates" setting.

In the following figure the animation of a universal joint is shown. The light blue coordinate system is frame\_a and the dark blue coordinate system is frame\_b of the joint (here: n\_a = {0,0,1}, n\_b = {0,1,0}, phi\_start\_a = 90°, phi\_start\_b = 45°).



## Parameters

Name	Default	Description
animation	true	= true, if animation shall be enabled
n_a	{1,0,0}	Axis of revolute joint 1 resolved in frame_a
n_b	{0,1,0}	Axis of revolute joint 2 resolved in frame_b
<b>Initialization</b>		
initType	Modelica.Mechanics.MultiBody...	Type of initialization (defines usage of start values below)
phi_start_a	0	Initial value of rotation angle at frame_a (fixed or guess value) [deg]
phi_start_b	0	Initial value of rotation angle at frame_b (fixed or guess value) [deg]
w_start_a	0	Initial value of derivative of rotation angle at frame_a [deg/s]
w_start_b	0	Initial value of derivative of rotation angle at frame_b [deg/s]
a_start_a	0	Initial value of second derivative of rotation angle at frame_a [deg/s <sup>2</sup> ]
a_start_b	0	Initial value of second derivative of rotation angle at frame_b [deg/s <sup>2</sup> ]
<b>Animation</b>		
if animation = true		
cylinderLength	world.defaultJointLength	Length of cylinders representing the joint axes [m]
cylinderDiameter	world.defaultJointWidth	Diameter of cylinders representing the joint axes [m]
cylinderColor	Modelica.Mechanics.MultiBody...	Color of cylinders representing the joint axes
specularCoefficient	world.defaultSpecularCoeffic...	Reflection of ambient light (= 0: light is completely absorbed)
<b>Advanced</b>		
enforceStates	false	= true, if generalized variables shall be used as states

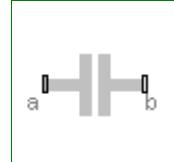
(StateSelect.always)

## Connectors

Name	Description
frame_a	Coordinate system fixed to the component with one cut-force and cut-torque
frame_b	Coordinate system fixed to the component with one cut-force and cut-torque

## Modelica.Mechanics.MultiBody.Joints.Planar

Planar joint (3 degrees-of-freedom, 6 potential states)



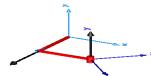
## Information

Joint where frame\_b can move in a plane and can rotate around an axis orthogonal to the plane. The plane is defined by vector n which is perpendicular to the plane and by vector n\_x, which points in the direction of the x-axis of the plane. frame\_a and frame\_b coincide when prismatic\_a.s=0, prismatic\_b=0 and revolute.phi=0. This joint has the following potential states:

- the relative distance prismatic\_x.s [m] along axis n\_x,
- the relative distance prismatic\_y.s [m] along axis n\_y = cross(n,n\_x),
- the relative angle revolute.phi [rad] around axis n,
- the relative velocity prismatic\_x.v [m/s] (= der(prismatic\_x.s)).
- the relative velocity prismatic\_y.v [m/s] (= der(prismatic\_y.s)).
- the relative angular velocity revolute.w [rad/s] (= der(revolute.phi))

The potential states are used as candidates for automatic selection of states from the tool. This may be enforced by setting "enforceStates=true" in the **Advanced** menu (this means that the Modelica attributes stateSelect=StateSelect.always are set on these variables). The states are usually selected automatically. In certain situations, especially when closed kinematic loops are present, it might be slightly more efficient, when using the "enforceStates" setting.

In the following figure the animation of a planar joint is shown. The light blue coordinate system is frame\_a and the dark blue coordinate system is frame\_b of the joint. The black arrows are parameter vectors "n" and "n\_x" (here: n = {0,1,0}, n\_x = {0,0,1}, s\_start\_x = 0.5, s\_start\_y = 0.5, phi\_start = 45°).



## Parameters

Name	Default	Description
animation	true	= true, if animation shall be enabled
n	{0,0,1}	Axis orthogonal to unconstrained plane, resolved in frame_a (= same as in frame_b)
n_x	{1,0,0}	Vector in direction of x-axis of plane, resolved in frame_a (n_x shall be orthogonal to n)
Initialization		
initType	Modelica.Mechanics.MultiBody..	Type of initialization (defines usage of start values below)
s_start_x	0	Initial value of x-distance (along n_x; fixed or guess value) [m]
s_start_y	0	Initial value of y-distance (along cross(n,n_x); fixed or guess value) [m]

## 518 Modelica.Mechanics.MultiBody.Joints.Planar

phi_start	0	Initial value of rotation angle along n (fixed or guess value) [deg]
v_start_x	0	Initial value of derivative of x-distance [m/s]
v_start_y	0	Initial value of derivative of y-distance [m/s]
w_start	0	Initial value of derivative of rotation angle [deg/s]
a_start_x	0	Initial value of second derivative of x-distance [m/s <sup>2</sup> ]
a_start_y	0	Initial value of second derivative of y-distance [m/s <sup>2</sup> ]
wd_start	0	Initial value of second derivative of rotation angle [deg/s <sup>2</sup> ]

Animation		
if animation = true		
cylinderLength	world.defaultJointLength	Length of revolute cylinder [m]
cylinderDiameter	world.defaultJointWidth	Diameter of revolute cylinder [m]
cylinderColor	Modelica.Mechanics.MultiBody.. .	Color of revolute cylinder
boxWidth	0.3*cylinderDiameter	Width of prismatic joint boxes [m]
boxHeight	boxWidth	Height of prismatic joint boxes [m]
boxColor	Modelica.Mechanics.MultiBody.. .	Color of prismatic joint boxes

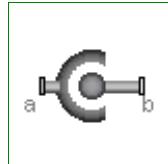
Advanced		
enforceStates	false	= true, if generalized variables (s,phi,v,w) shall be used as states (StateSelect.always)

## Connectors

Name	Description
frame_a	Coordinate system fixed to the component with one cut-force and cut-torque
frame_b	Coordinate system fixed to the component with one cut-force and cut-torque

## Modelica.Mechanics.MultiBody.Joints.Spherical

Spherical joint (3 constraints and no potential states, or 3 degrees-of-freedom and 3 states)



## Information

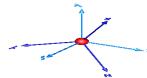
Joint with **3 constraints** that define that the origin of frame\_a and the origin of frame\_b coincide. By default this joint defines only the 3 constraints without any potential states. If parameter **enforceStates** is set to **true** in the "Advanced" menu, three states are introduced. Depending on parameter **useQuaternions** these are either quaternions and the relative angular velocity or 3 angles and the angle derivatives. In the latter case the orientation of frame\_b is computed by rotating frame\_a along the axes defined in parameter vector "sequence\_angleStates" (default = {1,2,3}, i.e., the Cardan angle sequence) around the angles used as states. For example, the default is to rotate the x-axis of frame\_a around angles[1], the new y-axis around angles[2] and the new z-axis around angles[3], arriving at frame\_b. If angles are used as states there is the slight disadvantage that a singular configuration is present leading to a division by zero.

If this joint is used in a **chain** structure, a Modelica translator has to select orientation coordinates of a body as states, if the default setting is used. It is usually better to use relative coordinates in the spherical joint as states, and therefore in this situation parameter **enforceStates** might be set to **true**.

If this joint is used in a **loop** structure, the default setting results in a **cut-joint** that breaks the loop in independent kinematic pieces, hold together by the constraints of this joint. As a result, a Modelica translator will first try to select 3 generalized coordinates in the joints of the remaining parts of the loop and their first

derivative as states and if this is not possible, e.g., because there are only spherical joints in the loop, will select coordinates from a body of the loop as states.

In the following figure the animation of a spherical joint is shown. The light blue coordinate system is frame\_a and the dark blue coordinate system is frame\_b of the joint. (here: angles\_start = {45, 45, 45} $^{\circ}$ ).



## Parameters

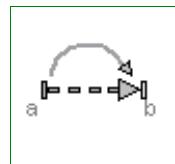
Name	Default	Description
animation	true	= true, if animation shall be enabled (show sphere)
<b>if animation = true</b>		
sphereDiameter	world.defaultJointLength	Diameter of sphere representing the spherical joint [m]
sphereColor	Modelica.Mechanics.MultiBody..	Color of sphere representing the spherical joint
specularCoefficient	world.defaultSpecularCoeffic...	Reflection of ambient light (= 0: light is completely absorbed)
<b>Initialization</b>		
initType	Modelica.Mechanics.MultiBody..	Type of initialization (defines usage of start values below)
sequence_start	{1,2,3}	Sequence of rotations to rotate frame_a into frame_b at initial time
angles_start[3]	{0,0,0}	Initial values of angles to rotate frame_a around 'sequence_start' axes into frame_b [deg]
w_rel_a_start[3]	{0,0,0}	Initial values of angular velocity of frame_b with respect to frame_a, resolved in frame_a [deg/s]
z_rel_a_start[3]	{0,0,0}	Initial values of angular acceleration z_rel = der(w_rel) [deg/s <sup>2</sup> ]
<b>Advanced</b>		
enforceStates	false	= true, if relative variables of spherical joint shall be used as states (StateSelect.always)
useQuaternions	true	= true, if quaternions shall be used as states otherwise use 3 angles as states (provided enforceStates=true)
sequence_angleStates	{1,2,3}	Sequence of rotations to rotate frame_a into frame_b around the 3 angles used as states

## Connectors

Name	Description
frame_a	Coordinate system fixed to the component with one cut-force and cut-torque
frame_b	Coordinate system fixed to the component with one cut-force and cut-torque

## Modelica.Mechanics.MultiBody.Joints.FreeMotion

Free motion joint (6 degrees-of-freedom, 12 potential states)



## Information

Joint which does not constrain the motion between frame\_a and frame\_b. Such a joint is only meaningful if the **relative** distance and orientation between frame\_a and frame\_b, and their derivatives, shall be used as **states**.

Note, that **bodies** such as Parts.Body, Parts.BodyShape, have potential states describing the distance and orientation, and their derivatives, between the **world frame** and a **body fixed frame**. Therefore, if these potential state variables are suited, a FreeMotion joint is not needed.

The states of the FreeMotion object are:

- The **relative position vector**  $r_{rel\_a}$  from the origin of frame\_a to the origin of frame\_b, resolved in frame\_a and the **relative velocity**  $v_{rel\_a}$  of the origin of frame\_b with respect to the origin of frame\_a, resolved in frame\_a ( $= \text{der}(r_{rel\_a})$ ).
- If parameter **useQuaternions** in the "Advanced" menu is **true** (this is the default), then **4 quaternions** are states. Additionally, the coordinates of the relative angular velocity vector are **3 potential states**.  
If **useQuaternions** in the "Advanced" menu is **false**, then **3 angles** and the derivatives of these angles are potential states. The orientation of frame\_b is computed by rotating frame\_a along the axes defined in parameter vector "sequence\_angleStates" (default = {1,2,3}, i.e., the Cardan angle sequence) around the angles used as states. For example, the default is to rotate the x-axis of frame\_a around angles[1], the new y-axis around angles[2] and the new z-axis around angles[3], arriving at frame\_b.

The quaternions have the slight disadvantage that there is a non-linear constraint equation between the 4 quaternions. Therefore, at least one non-linear equation has to be solved during simulation. A tool might, however, analytically solve this simple constraint equation. Using the 3 angles as states has the disadvantage that there is a singular configuration in which a division by zero will occur. If it is possible to determine in advance for an application class that this singular configuration is outside of the operating region, the 3 angles might be used as states by setting **useQuaternions = false**.

In text books about 3-dimensional mechanics often 3 angles and the angular velocity are used as states. This is not the case here, since 3 angles and their derivatives are used as states (if **useQuaternions = false**). The reason is that for real-time simulation the discretization formula of the integrator might be "inlined" and solved together with the model equations. By appropriate symbolic transformation the performance is drastically increased if angles and their derivatives are used as states, instead of angles and the angular velocity.

If parameter **enforceStates** is set to **true** (= the default) in the "Advanced" menu, then FreeMotion variables are forced to be used as states according to the setting of parameters "useQuaternions" and "sequence\_angleStates".

In the following figure the animation of a FreeMotion joint is shown. The light blue coordinate system is frame\_a and the dark blue coordinate system is frame\_b of the joint. (here:  $r_{rel\_a\_start} = \{0.5, 0, 0.5\}$ ,  $\text{angles\_start} = \{45, 45, 45\}^\circ$ ).



## Parameters

Name	Default	Description
animation	true	= true, if animation shall be enabled (show arrow from frame_a to frame_b)
Initialization		
initType	Modelica.Mechanics.MultiBody.. . .	Type of initialization (defines usage of start values below)
$r_{rel\_a\_start}[3]$	{0,0,0}	Initial values of $r_{rel\_a}$ (vector from origin of frame_a to origin of frame_b resolved in frame_a) [m]

sequence_start	{1,2,3}	Sequence of rotations to rotate frame_a into frame_b at initial time
angles_start[3]	{0,0,0}	Initial values of angles to rotate frame_a around 'sequence_start' axes into frame_b [deg]
v_rel_a_start[3]	{0,0,0}	Initial values of velocity v_rel_a = der(r_rel_a) [m/s]
w_rel_a_start[3]	{0,0,0}	Initial values of angular velocity of frame_b with respect to frame_a resolved in frame_a [deg/s]
a_rel_a_start[3]	{0,0,0}	Initial values of acceleration a_rel_a = der(v_rel_a) [m/s <sup>2</sup> ]
z_rel_a_start[3]	{0,0,0}	Initial values of angular acceleration z_rel_a = der(w_rel_a) [deg/s <sup>2</sup> ]

**Animation**

if animation = true

arrowDiameter	world.defaultArrowDiameter	Diameter of arrow from frame_a to frame_b [m]
arrowColor	Modelica.Mechanics.MultiBody.. .	Color of arrow
specularCoefficient	world.defaultSpecularCoeffic...	Reflection of ambient light (= 0: light is completely absorbed)

**Advanced**

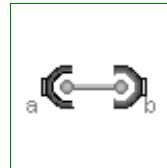
enforceStates	true	= true, if relative variables between frame_a and frame_b shall be used as states
useQuaternions	true	= true, if quaternions shall be used as states otherwise use 3 angles as states
sequence_angleStates	{1,2,3}	Sequence of rotations to rotate frame_a into frame_b around the 3 angles used as states

**Connectors**

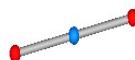
Name	Description
frame_a	Coordinate system fixed to the component with one cut-force and cut-torque
frame_b	Coordinate system fixed to the component with one cut-force and cut-torque

**Modelica.Mechanics.MultiBody.Joints.SphericalSpherical**

Spherical - spherical joint aggregation (1 constraint, no potential states) with an optional point mass in the middle

**Information**

Joint that has a spherical joint on each of its two ends. The rod connecting the two spherical joints is approximated by a point mass that is located in the middle of the rod. When the mass is set to zero (default), special code for a massless body is generated. In the following default animation figure, the two spherical joints are represented by two red spheres, the connecting rod by a grey cylinder and the point mass in the middle of the rod by a light blue sphere:



This joint introduces **one constraint** defining that the distance between the origin of frame\_a and the origin of frame\_b is constant (= rodLength). It is highly recommended to use this joint in loops whenever possible, because this enhances the efficiency considerably due to smaller systems of non-linear algebraic equations.

It is sometimes desirable to **compute** the **rodLength** of the connecting rod during initialization. For this,

parameter **computeLength** has to be set to **true** and instead **one** other, easier to determine, position variable in the same loop needs to have a fixed attribute of **true**. For example, if a loop consists of one Revolute joint, one Prismatic joint and a SphericalSpherical joint, one may fix the start values of the revolute joint angle and of the relative distance of the prismatic joint in order to compute the rodLength of the rod.

It is not possible to connect other components, such as a body with mass properties or a special visual shape object to the rod connecting the two spherical joints. If this is needed, use instead joint Joints.**UniversalSpherical** that has this property.

## Parameters

Name	Default	Description
animation	true	= true, if animation shall be enabled
showMass	true	= true, if mass shall be shown (provided animation = true and m > 0)
computeRodLength	false	= true, if rodLength shall be computed during initialization (see info)
rodLength	1	Distance between the origins of frame_a and frame_b (if computeRodLength=true, guess value) [m]
m	0	Mass of rod (= point mass located in middle of rod) [kg]

## Animation

if animation = true

sphereDiameter	world.defaultJointLength	Diameter of spheres representing the spherical joints [m]
sphereColor	Modelica.Mechanics.MultiBody..	Color of spheres representing the spherical joints
rodDiameter	sphereDiameter/Types.Default...	Diameter of rod connecting the two spherical joint [m]
rodColor	Modelica.Mechanics.MultiBody..	Color of rod connecting the two spherical joints
specularCoefficient	world.defaultSpecularCoeffic...	Reflection of ambient light (= 0: light is completely absorbed)

if animation = true and showMass = true and m > 0

massDiameter	sphereDiameter	Diameter of sphere representing the mass point [m]
massColor	Modelica.Mechanics.MultiBody..	Color of sphere representing the mass point

## Advanced

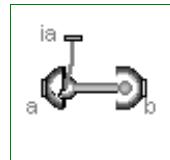
kinematicConstraint	true	= false, if no constraint shall be defined, due to analytically solving a kinematic loop
checkTotalPower	false	= true, if total power flowing into this component shall be determined (must be zero)

## Connectors

Name	Description
frame_a	Coordinate system fixed to the component with one cut-force and cut-torque
frame_b	Coordinate system fixed to the component with one cut-force and cut-torque

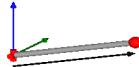
## Modelica.Mechanics.MultiBody.Joints.UniversalSpherical

Universal - spherical joint aggregation (1 constraint, no potential states)



## Information

This component consists of a **universal joint** at frame\_a and a **spherical joint** at frame\_b that are connected together with a **rigid rod**, see default animation figure (the arrows are not part of the default animation):



This joint aggregation has no mass and no inertia and introduces the constraint that the distance between the origin of frame\_a and the origin of frame\_b is constant (= Frames.length(rRod\_ia)). The universal joint is defined in the following way:

- The rotation **axis** of revolute joint 1 is along parameter vector n1\_a which is fixed in frame\_a.
- The rotation **axis** of revolute joint 2 is perpendicular to axis 1 and to the line connecting the universal and the spherical joint.

The definition of axis 2 of the universal joint is performed according to the most often occurring case. In a future release, axis 2 might be explicitly definable via a parameter. However, the treatment is much more complicated and the number of operations is considerably higher, if axis 2 is not orthogonal to axis 1 and to the connecting rod.

Note, there is a **singularity** when axis 1 and the connecting rod are parallel to other. Therefore, if possible n1\_a should be selected in such a way that it is perpendicular to rRod\_ia in the initial configuration (i.e., the distance to the singularity is as large as possible).

An additional **frame\_ia** is present. It is **fixed** in the connecting **rod** at the origin of **frame\_a**. The placement of frame\_ia on the rod is implicitly defined by the universal joint (frame\_a and frame\_ia coincide when the angles of the two revolute joints of the universal joint are zero) and by parameter vector **rRod\_ia**, the position vector from the origin of frame\_a to the origin of frame\_b, resolved in frame\_ia.

The easiest way to define the parameters of this joint is by moving the MultiBody system in a **reference configuration** where **all frames** of all components are **parallel** to other (alternatively, at least frame\_a and frame\_ia of the UniversalSpherical joint should be parallel to other when defining an instance of this component). Since frame\_a and frame\_ia are parallel to other, vector **rRod\_ia** from frame\_a to frame\_b resolved in frame\_ia can be resolved in frame\_a (or the **world frame**, if all frames are parallel to other).

This joint aggregation can be used in cases where in reality a rod with spherical joints at end are present. Such a system has an additional degree of freedom to rotate the rod along its axis. In practice this rotation is usually of no interested and is mathematically removed by replacing one of the spherical joints by a universal joint. Still, in most cases the Joints.SphericalSpherical joint aggregation can be used instead of the UniversalSpherical joint since the rod is animated and its mass properties are approximated by a point mass in the middle of the rod. The SphericalSpherical joint has the advantage that it does not have a singular configuration.

In the public interface of the UniversalSpherical joint, the following (final) **parameters** are provided:

```
parameter Real rodLength(unit="m") "Length of rod";
parameter Real eRod_ia[3] "Unit vector along rod, resolved in frame_ia";
parameter Real e2_ia [3] "Unit vector along axis 2, resolved in frame_ia";
```

This allows a more convenient definition of data which is related to the rod. For example, if a box shall be connected at frame\_ia directing from the origin of frame\_a to the middle of the rod, this might be defined as:

```
Modelica.Mechanics.MultiBody.Joints.UniversalSpherical jointUS(rRod_ia={1.2,
1, 0.2});
Modelica.Mechanics.MultiBody.Visualizers.FixedShape shape(shapeType
= "box",
lengthDirection = jointUS.eRod_ia,
widthDirection = jointUS.e2_ia,
length =
width =
jointUS.rodLength/2,
```

```

jointUS.rodLength/10);
equation
  connect(jointUS.frame_ia, shape.frame_a);

```

## Parameters

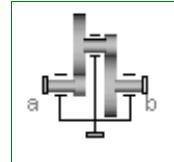
Name	Default	Description
animation	true	= true, if animation shall be enabled
showUniversalAxes	true	= true, if universal joint shall be visualized with two cylinders, otherwise with a sphere (provided animation=true)
computeRodLength	false	= true, if distance between frame_a and frame_b shall be computed during initialization (see info)
n1_a	{0,0,1}	Axis 1 of universal joint resolved in frame_a (axis 2 is orthogonal to axis 1 and to rod)
rRod_ia[3]	{1,0,0}	Vector from origin of frame_a to origin of frame_b, resolved in frame_ia (if computeRodLength=true, rRod_ia is only an axis vector along the connecting rod) [m]
<b>Animation</b>		
if animation = true		
sphereDiameter	world.defaultJointLength	Diameter of spheres representing the universal and the spherical joint [m]
sphereColor	Modelica.Mechanics.MultiBody..	Color of spheres representing the universal and the spherical joint
rodShapeType	"cylinder"	Shape type of rod connecting the universal and the spherical joint
rodWidth	sphereDiameter/Types.Default..	Width of rod shape in direction of axis 2 of universal joint. [m]
rodHeight	rodWidth	Height of rod shape in direction that is orthogonal to rod and to axis 2 [m]
rodExtra	0.0	Additional parameter depending on rodShapeType
rodColor	Modelica.Mechanics.MultiBody..	Color of rod shape connecting the universal and the spherical joints
specularCoefficient	world.defaultSpecularCoeffic...	Reflection of ambient light (= 0: light is completely absorbed)
if animation = true and showUniversalAxes		
cylinderLength	world.defaultJointLength	Length of cylinders representing the two universal joint axes [m]
cylinderDiameter	world.defaultJointWidth	Diameter of cylinders representing the two universal joint axes [m]
cylinderColor	Modelica.Mechanics.MultiBody..	Color of cylinders representing the two universal joint axes
<b>Advanced</b>		
kinematicConstraint	true	= false, if no constraint shall be defined, due to analytically solving a kinematic loop
checkTotalPower	false	= true, if total power flowing into this component shall be determined (must be zero)

## Connectors

Name	Description
frame_a	Coordinate system fixed to the component with one cut-force and cut-torque
frame_b	Coordinate system fixed to the component with one cut-force and cut-torque
frame_ia	Coordinate system at the origin of frame_a, fixed at the rod connecting the universal with the spherical joint

## Modelica.Mechanics.MultiBody.Joints.GearConstraint

Ideal 3-dim. gearbox (arbitrary shaft directions)



## Information

This ideal massless joint provides a gear constraint between frames frame\_a and frame\_b. The axes of rotation of frame\_a and frame\_b may be arbitrary.

## Reference

SCHWEIGER, Christian ; OTTER, Martin: Modelling 3D Mechanical Effects of 1-dim. Powertrains. In: *Proceedings of the 3rd International Modelica Conference*. Linköping : The Modelica Association and Linköping University, November 3-4, 2003, pp. 149-158

## Parameters

Name	Default	Description
ratio	2	Gear speed ratio
n_a	{1,0,0}	Axis of rotation of shaft a (same coordinates in frame_a, frame_b, bearing)
n_b	{1,0,0}	Axis of rotation of shaft b (same coordinates in frame_a, frame_b, bearing)
r_a[3]	{0,0,0}	Vector from frame bearing to frame_a resolved in bearing [m]
r_b[3]	{0,0,0}	Vector from frame bearing to frame_b resolved in bearing [m]

## Connectors

Name	Description
frame_a	Coordinate system fixed to the component with one cut-force and cut-torque
frame_b	Coordinate system fixed to the component with one cut-force and cut-torque
bearing	Coordinate system fixed in the bearing

## Modelica.Mechanics.MultiBody.Joints.Assemblies

Joint aggregations for analytic loop handling

## Information

The joints in this package are mainly designed to be used in **kinematic loop** structures. Every component consists of **3 elementary joints**. These joints are combined in such a way that the kinematics of the 3 joints between frame\_a and frame\_b are computed from the movement of frame\_a and frame\_b, i.e., there are **no constraints** between frame\_a and frame\_b. This requires to solve a **non-linear system of equations** which is performed **analytically** (i.e., when a mathematical solution exists, it is computed efficiently and reliably). A detailed description how to use these joints is provided in [MultiBody.UsersGuide.Tutorial.LoopStructures.AnalyticLoopHandling](#).

The assembly joints in this package are named **JointXYZ** where **XYZ** are the first letters of the elementary

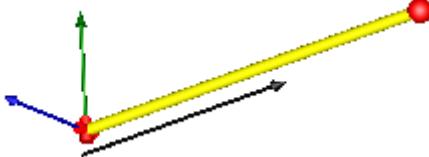
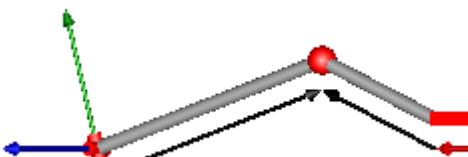
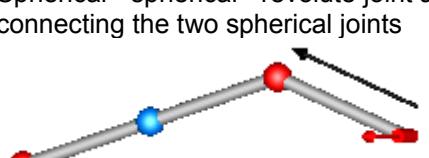
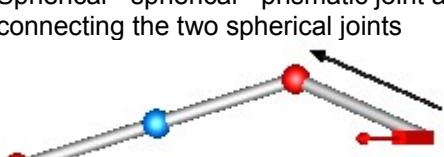
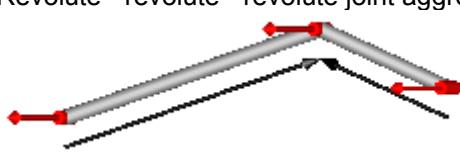
joints used in the component, in particular:

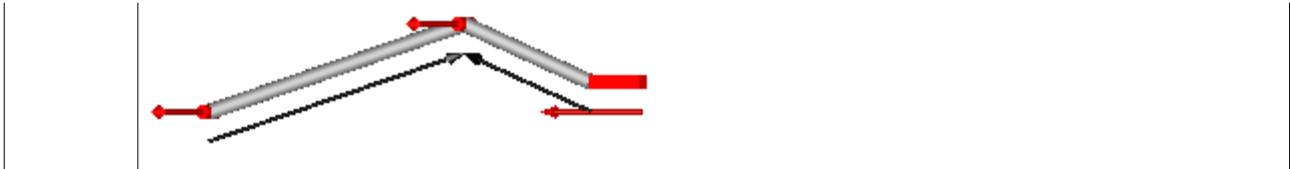
<b>P</b>	Prismatic joint
<b>R</b>	Revolute joint
<b>S</b>	Spherical joint
<b>U</b>	Universal joint

For example, JointUSR is an assembly joint consisting of a universal, a spherical and a revolute joint.

This package contains the following models:

## Content

Model	Description
<b>JointUPS</b>	Universal - prismatic - spherical joint aggregation 
<b>JointUSR</b>	Universal - spherical - revolute joint aggregation 
<b>JointUSP</b>	Universal - spherical - prismatic joint aggregation 
<b>JointSSR</b>	Spherical - spherical - revolute joint aggregation with an optional mass point at the rod connecting the two spherical joints 
<b>JointSSP</b>	Spherical - spherical - prismatic joint aggregation with an optional mass point at the rod connecting the two spherical joints 
<b>JointRRR</b>	Revolute - revolute - revolute joint aggregation for planar loops 
<b>JointRRP</b>	Revolute - revolute - prismatic joint aggregation for planar loops



Note, no component of this package has potential states, since the components are designed in such a way that the generalized coordinates of the used elementary joints are computed from the frame\_a and frame\_b coordinates. Still, it is possible to use the components in a tree structure. In this case states are selected from bodies that are connected to the frame\_a or frame\_b side of the component. In most cases this gives a less efficient solution, as if elementary joints of package Modelica.Mechanics.MultiBody.Joints would be used directly.

The analytic handling of kinematic loops by using joint aggregations with 6 degrees of freedom as provided in this package, is a **new** methodology. It is based on a more general method for solving non-linear equations of kinematic loops developed by Woernle and Hiller. An automatic application of this more general method is difficult, and a manual application is only suited for specialists in this field. The method introduced here is a compromise: It can be quite easily applied by an end user, but for a smaller class of kinematic loops. The method of the "characteristic pair of joints" from Woernle and Hiller is described in:

Woernle C.:

**Ein systematisches Verfahren zur Aufstellung der geometrischen Schliessbedingungen in kinematischen Schleifen mit Anwendung bei der Rückwärtstransformation für Industrieroboter.**  
Fortschritt-Berichte VDI, Reihe 18, Nr. 59, Düsseldorf: VDI-Verlag 1988, ISBN 3-18-145918-6.

Hiller M., and Woernle C.: **A Systematic Approach for Solving the Inverse Kinematic Problem of Robot Manipulators.**

Proceedings 7th World Congress Th. Mach. Mech., Sevilla 1987.

## Package Content

Name	Description
JointUPS	Universal - prismatic - spherical joint aggregation (no constraints, no potential states)
JointUSR	Universal - spherical - revolute joint aggregation (no constraints, no potential states)
JointUSP	Universal - spherical - prismatic joint aggregation (no constraints, no potential states)
JointSSR	Spherical - spherical - revolute joint aggregation with mass (no constraints, no potential states)
JointSSP	Spherical - spherical - prismatic joint aggregation with mass (no constraints, no potential states)
JointRRR	Planar revolute - revolute - revolute joint aggregation (no constraints, no potential states)
JointRRP	Planar revolute - revolute - prismatic joint aggregation (no constraints, no potential states)

### Modelica.Mechanics.MultiBody.Joints.Assemblies.JointUPS

Universal - prismatic - spherical joint aggregation (no constraints, no potential states)



#### Information

This component consists of a **universal** joint at frame\_a, a **spherical** joint at frame\_b and a **prismatic** joint along the line connecting the origin of frame\_a and the origin of frame\_b, see the default animation in the following figure (the axes vectors are not part of the default animation):



This joint aggregation has no mass and no inertia and introduces neither constraints nor potential state variables. It is especially useful to build up more complicated force elements where the mass and/or inertia of the force element shall be taken into account.

The universal joint is defined in the following way:

- The rotation **axis** of revolute joint **1** is along parameter vector **n1\_a** which is fixed in **frame\_a**.
- The rotation **axis** of revolute joint **2** is perpendicular to axis **1** and to the line connecting the universal and the spherical joint.

The definition of axis 2 of the universal joint is performed according to the most often occurring case. In a future release, axis 2 might be explicitly definable via a parameter. However, the treatment is much more complicated and the number of operations is considerably higher, if axis 2 is not orthogonal to axis 1 and to the connecting rod.

Note, there is a **singularity** when axis 1 and the connecting line are parallel to each other. Therefore, if possible **n1\_a** should be selected in such a way that it is perpendicular to **nAxis\_ia** in the initial configuration (i.e., the distance to the singularity is as large as possible).

An additional **frame\_ia** is present. It is **fixed** on the line connecting the universal and the spherical joint at the origin of **frame\_a**. The placement of **frame\_ia** on this line is implicitly defined by the universal joint (**frame\_a** and **frame\_ia** coincide when the angles of the two revolute joints of the universal joint are zero) and by parameter vector **nAxis\_ia**, an axis vector directed along the line from the origin of **frame\_a** to the spherical joint, resolved in **frame\_ia**.

An additional **frame\_ib** is present. It is **fixed** in the line connecting the prismatic and the spherical joint at the origin of **frame\_b**. It is always parallel to **frame\_ia**.

Note, this joint aggregation can be used in cases where in reality a rod with spherical joints at each end are present. Such a system has an additional degree of freedom to rotate the rod along its axis. In practice this rotation is usually of no interest and is mathematically removed by replacing one of the spherical joints by a universal joint.

The easiest way to define the parameters of this joint is by moving the MultiBody system in a **reference configuration** where **all frames** of all components are **parallel** to each other (alternatively, at least **frame\_a**, **frame\_ia** and **frame\_ib** of the **JointUSP** joint should be parallel to each other when defining an instance of this component).

## Parameters

Name	Default	Description
animation	true	= true, if animation shall be enabled
showUniversalAxes	true	= true, if universal joint shall be visualized with two cylinders, otherwise with a sphere (provided animation=true)
n1_a	{0,0,1}	Axis 1 of universal joint resolved in <b>frame_a</b> (axis 2 is orthogonal to axis 1 and to line from universal to spherical joint)
nAxis_ia[3]	{1,0,0}	Axis vector along line from origin of <b>frame_a</b> to origin of <b>frame_b</b> , resolved in <b>frame_ia</b> [m]
s_offset	0	Relative distance offset (distance between <b>frame_a</b> and <b>frame_b</b> = $s(t) + s_{\text{offset}}$ ) [m]
<b>Animation</b>		
if animation = true		

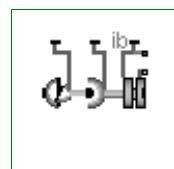
sphereDiameter	world.defaultJointLength	Diameter of spheres representing the spherical joints [m]
sphereColor	Modelica.Mechanics.MultiBody..	Color of spheres representing the spherical joints
axisDiameter	sphereDiameter/Types.Default...	Diameter of cylinder on the connecting line from frame_a to frame_b [m]
axisColor	Modelica.Mechanics.MultiBody..	Color of cylinder on the connecting line from frame_a to frame_b
specularCoefficient	world.defaultSpecularCoeffic...	Reflection of ambient light (= 0: light is completely absorbed)
<b>if animation = true and showUniversalAxes</b>		
cylinderLength	world.defaultJointLength	Length of cylinders representing the two universal joint axes [m]
cylinderDiameter	world.defaultJointWidth	Diameter of cylinders representing the two universal joint axes [m]
cylinderColor	Modelica.Mechanics.MultiBody..	Color of cylinders representing the two universal joint axes
<b>Advanced</b>		
checkTotalPower	false	= true, if total power flowing into this component shall be determined (must be zero)

## Connectors

Name	Description
frame_a	Coordinate system fixed to the component with one cut-force and cut-torque
frame_b	Coordinate system fixed to the component with one cut-force and cut-torque
frame_ia	Coordinate system at origin of frame_a fixed at prismatic joint
frame_ib	Coordinate system at origin of frame_b fixed at prismatic joint
axis	1-dim. translational flange that drives the prismatic joint
bearing	1-dim. translational flange of the drive bearing of the prismatic joint

## Modelica.Mechanics.MultiBody.Joints.Assemblies.JointUSR

Universal - spherical - revolute joint aggregation (no constraints, no potential states)



## Information

This component consists of a **universal** joint at frame\_a, a **revolute** joint at frame\_b and a **spherical** joint which is connected via **rod1** to the universal and via **rod2** to the revolute joint, see the default animation in the following figure (the axes vectors are not part of the default animation):



This joint aggregation has no mass and no inertia and introduces neither constraints nor potential state variables. It should be used in kinematic loops whenever possible since the non-linear system of equations introduced by this joint aggregation is solved **analytically** (i.e., a solution is always computed, if a unique solution exists).

The universal joint is defined in the following way:

- The rotation **axis** of revolute joint 1 is along parameter vector n1\_a which is fixed in frame\_a.
- The rotation **axis** of revolute joint 2 is perpendicular to axis 1 and to the line connecting the universal

and the spherical joint (= rod 1).

The definition of axis 2 of the universal joint is performed according to the most often occurring case. In a future release, axis 2 might be explicitly definable via a parameter. However, the treatment is much more complicated and the number of operations is considerably higher, if axis 2 is not orthogonal to axis 1 and to the connecting rod.

Note, there is a **singularity** when axis 1 and the connecting rod are parallel to each other. Therefore, if possible  $n_1\_a$  should be selected in such a way that it is perpendicular to  $rRod1\_ia$  in the initial configuration (i.e., the distance to the singularity is as large as possible).

The rest of this joint aggregation is defined by the following parameters:

- The position of the spherical joint with respect to the universal joint is defined by vector **rRod1\_ia**. This vector is directed from **frame\_a** to the spherical joint and is resolved in **frame\_ia** (it is most simple to select **frame\_ia** such that it is parallel to **frame\_a** in the reference or initial configuration).
- The position of the spherical joint with respect to the revolute joint is defined by vector **rRod2\_ib**. This vector is directed from the inner frame of the revolute joint (**frame\_ib** or **revolute.frame\_a**) to the spherical joint and is resolved in **frame\_ib** (note, that **frame\_ib** and **frame\_b** are parallel to each other).
- The axis of rotation of the revolute joint is defined by axis vector **n\_b**. It is fixed and resolved in **frame\_b**.
- When specifying this joint aggregation with the definitions above, **two** different **configurations** are possible. Via parameter **phi\_guess** a guess value for **revolute.phi(t0)** at the initial time **t0** is given. The configuration is selected that is closest to **phi\_guess** ( $|revolute.phi - phi\_guess|$  is minimal).

An additional **frame\_ia** is present. It is **fixed** in the rod connecting the universal and the spherical joint at the origin of **frame\_a**. The placement of **frame\_ia** on the rod is implicitly defined by the universal joint (**frame\_a** and **frame\_ia** coincide when the angles of the two revolute joints of the universal joint are zero) and by parameter vector **rRod1\_ia**, the position vector from the origin of **frame\_a** to the spherical joint, resolved in **frame\_ia**.

An additional **frame\_ib** is present. It is **fixed** in the rod connecting the revolute and the spherical joint at the side of the revolute joint that is connected to this rod (= **rod2.frame\_a** = **revolute.frame\_a**).

An additional **frame\_im** is present. It is **fixed** in the rod connecting the revolute and the spherical joint at the side of the spherical joint that is connected to this rod (= **rod2.frame\_b**). It is always parallel to **frame\_ib**.

The easiest way to define the parameters of this joint is by moving the MultiBody system in a **reference configuration** where **all frames** of all components are **parallel** to each other (alternatively, at least **frame\_a** and **frame\_ia** of the JointUSR joint should be parallel to each other when defining an instance of this component).

In the public interface of the JointUSR joint, the following (final) **parameters** are provided:

```
parameter Real rod1Length(unit="m")    "Length of rod 1";
parameter Real eRod1_ia[3] "Unit vector along rod 1, resolved in frame_ia";
parameter Real e2_ia [3]   "Unit vector along axis 2, resolved in frame_ia";
```

This allows a more convenient definition of data which is related to rod 1. For example, if a box shall be connected at **frame\_ia** directing from the origin of **frame\_a** to the middle of rod 1, this might be defined as:

```
Modelica.Mechanics.MultiBody.Joints.Assemblies.JointUSR
jointUSR(rRod1_ia={1.2, 1, 0.2});
  Modelica.Mechanics.MultiBody.Visualizers.FixedShape      shape(shapeType
= "box",
                           lengthDirection =
jointUSR.eRod1_ia,
                           widthDirection = jointUSR.e2_ia,
                           length        =
                           width         =
jointUSR.rod1Length/2,
                           width         =
jointUSR.rod1Length/10);
```

```

equation
  connect(jointUSP.frame_ia, shape.frame_a);

```

## Parameters

Name	Default	Description
animation	true	= true, if animation shall be enabled
showUniversalAxes	true	= true, if universal joint shall be visualized with two cylinders, otherwise with a sphere (provided animation=true)
n1_a	{0,0,1}	Axis 1 of universal joint fixed and resolved in frame_a (axis 2 is orthogonal to axis 1 and to rod 1)
n_b	{0,0,1}	Axis of revolute joint fixed and resolved in frame_b
rRod1_ia[3]	{1,0,0}	Vector from origin of frame_a to spherical joint, resolved in frame_ia [m]
rRod2_ib[3]	{-1,0,0}	Vector from origin of frame_ib to spherical joint, resolved in frame_ib [m]
phi_offset	0	Relative angle offset of revolute joint (angle = phi(t) + from_deg(phi_offset)) [deg]
phi_guess	0	Select the configuration such that at initial time  phi(t0) - from_deg(phi_guess)  is minimal [deg]

## Animation

if animation = true

sphereDiameter	world.defaultJointLength	Diameter of the spheres representing the universal and the spherical joint [m]
sphereColor	Modelica.Mechanics.MultiBody..	Color of the spheres representing the universal and the spherical joint
rod1Diameter	sphereDiameter/Types.Default...	Diameter of rod 1 connecting the universal and the spherical joint [m]
rod1Color	Modelica.Mechanics.MultiBody..	Color of rod 1 connecting the universal and the spherical joint
rod2Diameter	rod1Diameter	Diameter of rod 2 connecting the revolute and the spherical joint [m]
rod2Color	rod1Color	Color of rod 2 connecting the revolute and the spherical joint
revoluteDiameter	world.defaultJointWidth	Diameter of cylinder representing the revolute joint [m]
revoluteLength	world.defaultJointLength	Length of cylinder representing the revolute joint [m]
revoluteColor	Modelica.Mechanics.MultiBody..	Color of cylinder representing the revolute joint
specularCoefficient	world.defaultSpecularCoeffic...	Reflection of ambient light (= 0: light is completely absorbed)

if animation = true and showUniversalAxes

cylinderLength	world.defaultJointLength	Length of cylinders representing the two universal joint axes [m]
cylinderDiameter	world.defaultJointWidth	Diameter of cylinders representing the two universal joint axes [m]
cylinderColor	Modelica.Mechanics.MultiBody..	Color of cylinders representing the two universal joint

	.	axes
<b>Advanced</b>		
checkTotalPower	false	= true, if total power flowing into this component shall be determined (must be zero)

## Connectors

Name	Description
frame_a	Coordinate system fixed to the component with one cut-force and cut-torque
frame_b	Coordinate system fixed to the component with one cut-force and cut-torque
frame_ia	Coordinate system at origin of frame_a fixed at connecting rod of universal and spherical joint
frame_ib	Coordinate system at origin of frame_b fixed at connecting rod of spherical and revolute joint
frame_im	Coordinate system at origin of spherical joint fixed at connecting rod of spherical and revolute joint
axis	1-dim. rotational flange that drives the revolute joint
bearing	1-dim. rotational flange of the drive bearing of the revolute joint

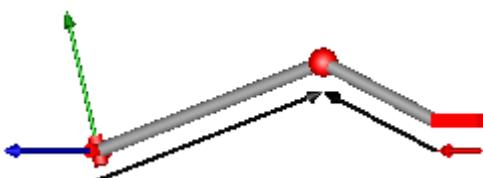
## Modelica.Mechanics.MultiBody.Joints.Assemblies.JointUSP

Universal - spherical - prismatic joint aggregation (no constraints, no potential states)



## Information

This component consists of a **universal** joint at frame\_a, a **prismatic** joint at frame\_b and a **spherical** joint which is connected via **rod1** to the universal and via **rod2** to the prismatic joint, see the default animation in the following figure (the axes vectors are not part of the default animation):



This joint aggregation has no mass and no inertia and introduces neither constraints nor potential state variables. It should be used in kinematic loops whenever possible since the non-linear system of equations introduced by this joint aggregation is solved **analytically** (i.e., a solution is always computed, if a unique solution exists).

The universal joint is defined in the following way:

- The rotation **axis** of revolute joint 1 is along parameter vector **n1\_a** which is fixed in frame\_a.
- The rotation **axis** of revolute joint 2 is perpendicular to axis 1 and to the line connecting the universal and the spherical joint (= rod 1).

The definition of axis 2 of the universal joint is performed according to the most often occurring case. In a future release, axis 2 might be explicitly definable via a parameter. However, the treatment is much more complicated and the number of operations is considerably higher, if axis 2 is not orthogonal to axis 1 and to the connecting rod.

Note, there is a **singularity** when axis 1 and the connecting rod are parallel to each other. Therefore, if possible **n1\_a** should be selected in such a way that it is perpendicular to **rRod1\_ia** in the initial configuration (i.e., the distance to the singularity is as large as possible).

The rest of this joint aggregation is defined by the following parameters:

- The position of the spherical joint with respect to the universal joint is defined by vector **rRod1\_ia**. This vector is directed from frame\_a to the spherical joint and is resolved in frame\_ia (it is most

- simple to select frame\_ia such that it is parallel to frame\_a in the reference or initial configuration).
- The position of the spherical joint with respect to the prismatic joint is defined by vector **rRod2\_ib**. This vector is directed from the inner frame of the prismatic joint (frame\_ib or prismatic.frame\_a) to the spherical joint and is resolved in frame\_ib (note, that frame\_ib and frame\_b are parallel to each other).
- The axis of translation of the prismatic joint is defined by axis vector **n\_b**. It is fixed and resolved in frame\_b.
- The two frames of the prismatic joint, i.e., frame\_b and frame\_ib, are parallel to each other. The distance between the origins of these two frames along axis n\_b is equal to "prismatic.s(t) + s\_offset", where "prismatic.s(t)" is a time varying variable and "s\_offset" is a fixed, constant offset parameter.
- When specifying this joint aggregation with the definitions above, **two different configurations** are possible. Via parameter **s\_guess** a guess value for prismatic.s(t0) at the initial time t0 is given. The configuration is selected that is closest to s\_guess ( $|prismatic.s - s_{guess}|$  is minimal).

An additional **frame\_ia** is present. It is **fixed** in the rod connecting the universal and the spherical joint at the origin of **frame\_a**. The placement of frame\_ia on the rod is implicitly defined by the universal joint (frame\_a and frame\_ia coincide when the angles of the two revolute joints of the universal joint are zero) and by parameter vector **rRod1\_ia**, the position vector from the origin of frame\_a to the spherical joint, resolved in **frame\_ia**.

An additional **frame\_ib** is present. It is **fixed** in the rod connecting the prismatic and the spherical joint at the side of the prismatic joint that is connected to this rod (= rod2.frame\_a = prismatic.frame\_a). It is always parallel to **frame\_b**.

An additional **frame\_im** is present. It is **fixed** in the rod connecting the prismatic and the spherical joint at the side of the spherical joint that is connected to this rod (= rod2.frame\_b). It is always parallel to **frame\_b**.

The easiest way to define the parameters of this joint is by moving the MultiBody system in a **reference configuration** where **all frames** of all components are **parallel** to each other (alternatively, at least frame\_a and frame\_ia of the JointUSP joint should be parallel to each other when defining an instance of this component).

In the public interface of the JointUSP joint, the following (final) **parameters** are provided:

```
parameter Real rod1Length(unit="m")    "Length of rod 1";
parameter Real eRod1_ia[3] "Unit vector along rod 1, resolved in frame_ia";
parameter Real e2_ia [3]  "Unit vector along axis 2, resolved in frame_ia";
```

This allows a more convenient definition of data which is related to rod 1. For example, if a box shall be connected at frame\_ia directing from the origin of frame\_a to the middle of rod 1, this might be defined as:

```
Modelica.Mechanics.MultiBody.Joints.Assemblies.JointUSP
jointUSP(rRod1_ia={1.2, 1, 0.2});
  Modelica.Mechanics.MultiBody.Visualizers.FixedShape      shape(shapeType
= "box",
                           lengthDirection =
jointUSP.eRod1_ia,
                           widthDirection = jointUSP.e2_ia,
                           length        =
                           width         =
jointUSP.rod1Length/2,
                           jointUSP.rod1Length/10);
equation
  connect(jointUSP.frame_ia, shape.frame_a);
```

## Parameters

Name	Default	Description
animation	true	= true, if animation shall be enabled
showUniversalAxes	true	= true, if universal joint shall be visualized with two cylinders, otherwise with a sphere (provided animation=true)
n1_a	{0,0,1}	Axis 1 of universal joint fixed and resolved in frame_a (axis 2 is orthogonal to axis 1 and to rod 1)
n_b	{-1,0,0}	Axis of prismatic joint fixed and resolved in frame_b
rRod1_ia[3]	{1,0,0}	Vector from origin of frame_a to spherical joint, resolved in frame_ia [m]
rRod2_ib[3]	{-1,0,0}	Vector from origin of frame_ib to spherical joint, resolved in frame_ib (frame_ib is parallel to frame_b) [m]
s_offset	0	Relative distance offset of prismatic joint (distance between the prismatic joint frames = s(t) + s_offset) [m]
s_guess	0	Select the configuration such that at initial time  s(t0)-s_guess  is minimal [m]

## Animation

if animation = true

sphereDiameter	world.defaultJointLength	Diameter of the spheres representing the universal and the spherical joint [m]
sphereColor	Modelica.Mechanics.MultiBody..	Color of the spheres representing the universal and the spherical joint
rod1Diameter	sphereDiameter/Types.Default...	Diameter of rod 1 connecting the universal and the spherical joint [m]
rod1Color	Modelica.Mechanics.MultiBody..	Color of rod 1 connecting the universal and the spherical joint
rod2Diameter	rod1Diameter	Diameter of rod 2 connecting the prismatic and the spherical joint [m]
rod2Color	rod1Color	Color of rod 2 connecting the prismatic and the spherical joint
boxWidthDirection	{0,1,0}	Vector in width direction of prismatic joint, resolved in frame_b
boxWidth	world.defaultJointWidth	Width of prismatic joint box [m]
boxHeight	boxWidth	Height of prismatic joint box [m]
boxColor	sphereColor	Color of prismatic joint box
specularCoefficient	world.defaultSpecularCoeffic...	Reflection of ambient light (= 0: light is completely absorbed)

if animation = true and showUniversalAxes

cylinderLength	world.defaultJointLength	Length of cylinders representing the two universal joint axes [m]
cylinderDiameter	world.defaultJointWidth	Diameter of cylinders representing the two universal joint axes [m]
cylinderColor	Modelica.Mechanics.MultiBody..	Color of cylinders representing the two universal joint axes

## Advanced

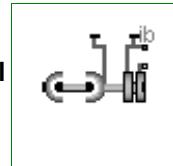
checkTotalPower	false	= true, if total power flowing into this component shall be determined (must be zero)
-----------------	-------	---

## Connectors

Name	Description
frame_a	Coordinate system fixed to the component with one cut-force and cut-torque
frame_b	Coordinate system fixed to the component with one cut-force and cut-torque
frame_ia	Coordinate system at origin of frame_a fixed at connecting rod of universal and spherical joint
frame_ib	Coordinate system at origin of frame_b fixed at connecting rod of spherical and prismatic joint
frame_im	Coordinate system at origin of spherical joint fixed at connecting rod of spherical and prismatic joint
axis	1-dim. translational flange that drives the prismatic joint
bearing	1-dim. translational flange of the drive bearing of the prismatic joint

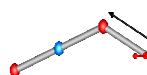
## Modelica.Mechanics.MultiBody.Joints.Assemblies.JointSSR

Spherical - spherical - revolute joint aggregation with mass (no constraints, no potential states)



### Information

This component consists of a **spherical** joint 1 at frame\_a, a **revolute** joint at frame\_b and a **spherical** joint 2 which is connected via rod 1 to the spherical joint 1 and via rod 2 to the revolute joint, see the default animation in the following figure (the axes vectors are not part of the default animation):



Besides an optional point mass in the middle of rod 1, this joint aggregation has no mass and no inertia, and introduces neither constraints nor potential state variables. It should be used in kinematic loops whenever possible since the non-linear system of equations introduced by this joint aggregation is solved **analytically** (i.e., a solution is always computed, if a unique solution exists).

An additional **frame\_ib** is present. It is **fixed** in rod 2 connecting the revolute and the spherical joint at the side of the revolute joint that is connected to this rod (= rod2.frame\_a = revolute.frame\_a).

An additional **frame\_im** is present. It is **fixed** in rod 2 connecting the revolute and the spherical joint at the side of spherical joint 2 that is connected to this rod (= rod2.frame\_b). It is always parallel to **frame\_ib**.

The easiest way to define the parameters of this joint is by moving the MultiBody system in a **reference configuration** where **all frames** of all components are **parallel** to each other (alternatively, at least frame\_b and frame\_ib of the JointSSR joint should be parallel to each other when defining an instance of this component).

### Parameters

Name	Default	Description
animation	true	= true, if animation shall be enabled
showMass	true	= true, if point mass on rod 1 shall be shown (provided animation = true and rod1Mass > 0)
rod1Length	1	Distance between the origins of the two spherical joints [m]
rod1Mass	0	Mass of rod 1 (= point mass located in middle of rod connecting the two spherical joints) [kg]
n_b	{0,0,1}	Axis of revolute joint fixed and resolved in frame_b

## 536 Modelica.Mechanics.MultiBody.Joints.Assemblies.JointSSR

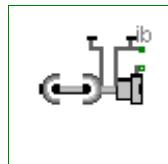
rRod2_ib[3]	{1,0,0}	Vector from origin of frame_ib to spherical joint in the middle, resolved in frame_ib [m]
phi_offset	0	Relative angle offset of revolute joint (angle = phi(t) + from_deg(phi_offset)) [deg]
phi_guess	0	Select the configuration such that at initial time  phi(t0) - from_deg(phi_guess)  is minimal [deg]
<b>Animation</b>		
if animation = true		
sphereDiameter	world.defaultJointLength	Diameter of the spheres representing the two spherical joints [m]
sphereColor	Modelica.Mechanics.MultiBody..	Color of the spheres representing the two spherical joints
rod1Diameter	sphereDiameter/Types.Default...	Diameter of rod 1 connecting the two spherical joints [m]
rod1Color	Modelica.Mechanics.MultiBody..	Color of rod 1 connecting the two spherical joint
rod2Diameter	rod1Diameter	Diameter of rod 2 connecting the revolute joint and spherical joint 2 [m]
rod2Color	rod1Color	Color of rod 2 connecting the revolute joint and spherical joint 2
revoluteDiameter	world.defaultJointWidth	Diameter of cylinder representing the revolute joint [m]
revoluteLength	world.defaultJointLength	Length of cylinder representing the revolute joint [m]
revoluteColor	Modelica.Mechanics.MultiBody..	Color of cylinder representing the revolute joint
specularCoefficient	world.defaultSpecularCoeffic...	Reflection of ambient light (= 0: light is completely absorbed)
<b>Advanced</b>		
checkTotalPower	false	= true, if total power flowing into this component shall be determined (must be zero)

## Connectors

Name	Description
frame_a	Coordinate system fixed to the component with one cut-force and cut-torque
frame_b	Coordinate system fixed to the component with one cut-force and cut-torque
frame_ib	Coordinate system at origin of frame_b fixed at connecting rod of spherical and revolute joint
frame_im	Coordinate system at origin of spherical joint in the middle fixed at connecting rod of spherical and revolute joint
axis	1-dim. rotational flange that drives the revolute joint
bearing	1-dim. rotational flange of the drive bearing of the revolute joint

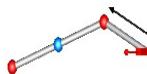
## Modelica.Mechanics.MultiBody.Joints.Assemblies.JointSSP

Spherical - spherical - prismatic joint aggregation with mass (no constraints, no potential states)



## Information

This component consists of a **spherical** joint 1 at frame\_a, a **prismatic** joint at frame\_b and a **spherical** joint 2 which is connected via rod 1 to the spherical joint 1 and via rod 2 to the prismatic joint, see the default animation in the following figure (the axes vectors are not part of the default animation):



Besides an optional point mass in the middle of rod 1, this joint aggregation has no mass and no inertia, and introduces neither constraints nor potential state variables. It should be used in kinematic loops whenever possible since the non-linear system of equations introduced by this joint aggregation is solved **analytically** (i.e., a solution is always computed, if a unique solution exists).

An additional **frame\_ib** is present. It is **fixed** in rod 2 connecting the prismatic and the spherical joint at the side of the prismatic joint that is connected to this rod (= rod2.frame\_a = prismatic.frame\_a).

An additional **frame\_im** is present. It is **fixed** in rod 2 connecting the prismatic and the spherical joint at the side of spherical joint 2 that is connected to this rod (= rod2.frame\_b). It is always parallel to **frame\_ib**.

The easiest way to define the parameters of this joint is by moving the MultiBody system in a **reference configuration** where **all frames** of all components are **parallel** to each other (alternatively, at least frame\_b and frame\_ib of the JointSSP joint should be parallel to each other when defining an instance of this component).

## Parameters

Name	Default	Description
animation	true	= true, if animation shall be enabled
showMass	true	= true, if point mass on rod 1 shall be shown (provided animation = true and rod1Mass > 0)
rod1Length	1	Distance between the origins of the two spherical joints [m]
rod1Mass	0	Mass of rod 1 (= point mass located in middle of rod connecting the two spherical joints) [kg]
n_b	{0,0,1}	Axis of prismatic joint fixed and resolved in frame_b
rRod2_ib[3]	{1,0,0}	Vector from origin of frame_ib to spherical joint in the middle, resolved in frame_ib [m]
s_offset	0	Relative distance offset of prismatic joint (distance between frame_b and frame_ib = s(t) + s_offset) [m]
s_guess	0	Select the configuration such that at initial time  s(t0)-s_guess  is minimal [m]

## Animation

if animation = true

sphereDiameter	world.defaultJointLength	Diameter of the spheres representing the two spherical joints [m]
sphereColor	Modelica.Mechanics.MultiBody..	Color of the spheres representing the two spherical joints
rod1Diameter	sphereDiameter/Types.Default...	Diameter of rod 1 connecting the two spherical joints [m]
rod1Color	Modelica.Mechanics.MultiBody..	Color of rod 1 connecting the two spherical joint
rod2Diameter	rod1Diameter	Diameter of rod 2 connecting the revolute joint and spherical joint 2 [m]
rod2Color	rod1Color	Color of rod 2 connecting the revolute joint and spherical joint 2
boxWidthDirection	{0,1,0}	Vector in width direction of prismatic joint box, resolved in frame_b

## 538 Modelica.Mechanics.MultiBody.Joints.Assemblies.JointSSP

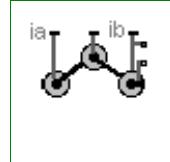
boxWidth	world.defaultJointWidth	Width of prismatic joint box [m]
boxHeight	boxWidth	Height of prismatic joint box [m]
boxColor	Modelica.Mechanics.MultiBody.. .	Color of prismatic joint box
specularCoefficient	world.defaultSpecularCoeffic...	Reflection of ambient light (= 0: light is completely absorbed)
<b>Advanced</b>		
checkTotalPower	false	= true, if total power flowing into this component shall be determined (must be zero)

## Connectors

Name	Description
frame_a	Coordinate system fixed to the component with one cut-force and cut-torque
frame_b	Coordinate system fixed to the component with one cut-force and cut-torque
frame_ib	Coordinate system at origin of frame_b fixed at connecting rod of spherical and prismatic joint
frame_im	Coordinate system at origin of spherical joint in the middle fixed at connecting rod of spherical and prismatic joint
axis	1-dim. translational flange that drives the prismatic joint
bearing	1-dim. translational flange of the drive bearing of the prismatic joint

## Modelica.Mechanics.MultiBody.Joints.Assemblies.JointRRR

Planar revolute - revolute - revolute joint aggregation (no constraints, no potential states)



## Information

This component consists of **3 revolute** joints with parallel axes of rotation that are connected together by two rods, see the default animation in the following figure (the axes vectors are not part of the default animation):



This joint aggregation introduces neither constraints nor state variables and should therefore be used in kinematic loops whenever possible to avoid non-linear systems of equations. It is only meaningful to use this component in **planar loops**. Basically, the position and orientation of the 3 revolute joints as well as of frame\_ia, frame\_ib, and frame\_im are calculated by solving analytically a non-linear equation, given the position and orientation at frame\_a and at frame\_b.

Connector **frame\_a** is the "left" side of the first revolute joint whereas **frame\_ia** is the "right" side of this revolute joint, fixed in rod 1. Connector **frame\_b** is the "right" side of the third revolute joint whereas **frame\_ib** is the "left" side of this revolute joint, fixed in rod 2. Finally, connector **frame\_im** is the connector at the "right" side of the revolute joint in the middle, fixed in rod 2.

The easiest way to define the parameters of this joint is by moving the MultiBody system in a **reference configuration** where **all frames** of all components are **parallel** to each other (alternatively, at least frame\_a, frame\_ia, frame\_im, frame\_ib, frame\_b of the JointRRR joint should be parallel to each other when defining an instance of this component).

Basically, the JointRRR model consists internally of a universal - spherical - revolute joint aggregation (= JointUSR). In a planar loop this will behave as if 3 revolute joints with parallel axes are connected by rigid rods.

## Parameters

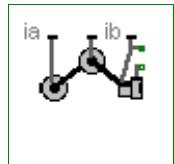
Name	Default	Description
animation	true	= true, if animation shall be enabled
n_a	{0,0,1}	Axes of revolute joints resolved in frame_a (all axes are parallel to each other)
rRod1_ia[3]	{1,0,0}	Vector from origin of frame_a to revolute joint in the middle, resolved in frame_ia [m]
rRod2_ib[3]	{-1,0,0}	Vector from origin of frame_ib to revolute joint in the middle, resolved in frame_ib [m]
phi_offset	0	Relative angle offset of revolute joint at frame_b (angle = phi(t) + from_deg(phi_offset)) [deg]
phi_guess	0	Select the configuration such that at initial time  phi(t0) - from_deg(phi_guess)  is minimal [deg]
<b>Animation</b>		
if animation = true		
cylinderLength	world.defaultJointLength	Length of cylinders representing the revolute joints [m]
cylinderDiameter	world.defaultJointWidth	Diameter of cylinders representing the revolute joints [m]
cylinderColor	Modelica.Mechanics.MultiBody..	Color of cylinders representing the revolute joints
rodDiameter	1.1*cylinderDiameter	Diameter of the two rods connecting the revolute joints [m]
rodColor	Modelica.Mechanics.MultiBody..	Color of the two rods connecting the revolute joint
specularCoefficient	world.defaultSpecularCoeffic...	Reflection of ambient light (= 0: light is completely absorbed)
<b>Advanced</b>		
checkTotalPower	false	= true, if total power flowing into this component shall be determined (must be zero)

## Connectors

Name	Description
frame_a	Coordinate system fixed to the component with one cut-force and cut-torque
frame_b	Coordinate system fixed to the component with one cut-force and cut-torque
frame_ia	Coordinate system at origin of frame_a fixed at connecting rod of left and middle revolute joint
frame_ib	Coordinate system at origin of frame_b fixed at connecting rod of middle and right revolute joint
frame_im	Coordinate system at origin of revolute joint in the middle fixed at connecting rod of middle and right revolute joint
axis	1-dim. rotational flange that drives the right revolute joint at frame_b
bearing	1-dim. rotational flange of the drive bearing of the right revolute joint at frame_b

## Modelica.Mechanics.MultiBody.Joints.Assemblies.JointRRP

Planar revolute - revolute - prismatic joint aggregation (no constraints, no potential states)



## Information

This component consists of **2 revolute** joints with parallel axes of rotation that and a **prismatic** joint with a

translational axis that is orthogonal to the revolute joint axes, see the default animation in the following figure (the axes vectors are not part of the default animation):



This joint aggregation introduces neither constraints nor state variables and should therefore be used in kinematic loops whenever possible to avoid non-linear systems of equations. It is only meaningful to use this component in **planar loops**. Basically, the position and orientation of the 3 joints as well as of frame\_ia, frame\_ib, and frame\_im are calculated by solving analytically a non-linear equation, given the position and orientation at frame\_a and at frame\_b.

Connector **frame\_a** is the "left" side of the first revolute joint whereas **frame\_ia** is the "right" side of this revolute joint, fixed in rod 1. Connector **frame\_b** is the "right" side of the prismatic joint whereas **frame\_ib** is the "left" side of this prismatic joint, fixed in rod 2. Finally, connector **frame\_im** is the connector at the "right" side of the revolute joint in the middle, fixed in rod 2. The frames frame\_b, frame\_ib, frame\_im are always parallel to each other.

The easiest way to define the parameters of this joint is by moving the MultiBody system in a **reference configuration** where **all frames** of all components are **parallel** to each other (alternatively, at least frame\_a, frame\_ia, frame\_im, frame\_ib, frame\_b of the JointRRP joint should be parallel to each other when defining an instance of this component).

Basically, the JointRRP model consists internally of a universal - spherical - prismatic joint aggregation (= JointUSP). In a planar loop this will behave as if 2 revolute joints with parallel axes and 1 prismatic joint are connected by rigid rods.

## Parameters

Name	Default	Description
animation	true	= true, if animation shall be enabled
n_a	{0,0,1}	Axes of the two revolute joints resolved in frame_a (both axes are parallel to each other)
n_b	{-1,0,0}	Axis of prismatic joint fixed and resolved in frame_b (must be orthogonal to revolute joint axes)
rRod1_ia[3]	{1,0,0}	Vector from origin of frame_a to revolute joint in the middle, resolved in frame_ia [m]
rRod2_ib[3]	{-1,0,0}	Vector from origin of frame_ib to revolute joint in the middle, resolved in frame_ib (frame_ib is parallel to frame_b) [m]
s_offset	0	Relative distance offset of prismatic joint (distance between the prismatic joint frames = s(t) + s_offset) [m]
s_guess	0	Select the configuration such that at initial time  s(t0)-s_guess  is minimal [m]

## Animation

if animation = true

cylinderLength	world.defaultJointLength	Length of cylinders representing the revolute joints [m]
cylinderDiameter	world.defaultJointWidth	Diameter of cylinders representing the revolute joints [m]
cylinderColor	Modelica.Mechanics.MultiBody..	Color of cylinders representing the revolute joints
boxWidthDirection	{0,1,0}	Vector in width direction of prismatic joint, resolved in frame_b

boxWidth	world.defaultJointWidth	Width of prismatic joint box [m]
boxHeight	boxWidth	Height of prismatic joint box [m]
boxColor	cylinderColor	Color of prismatic joint box
rodDiameter	1.1*cylinderDiameter	Diameter of the two rods connecting the joints [m]
rodColor	Modelica.Mechanics.MultiBody..	Color of the two rods connecting the joints
specularCoefficient	world.defaultSpecularCoeffic...	Reflection of ambient light (= 0: light is completely absorbed)
<b>Advanced</b>		
checkTotalPower	false	= true, if total power flowing into this component shall be determined (must be zero)

## Connectors

Name	Description
frame_a	Coordinate system fixed to the component with one cut-force and cut-torque
frame_b	Coordinate system fixed to the component with one cut-force and cut-torque
frame_ia	Coordinate system at origin of frame_a fixed at connecting rod of revolute joints
frame_ib	Coordinate system at origin of frame_b fixed at connecting rod of revolute and prismatic joint
frame_im	Coordinate system at origin of revolute joint in the middle fixed at connecting rod of revolute and prismatic joint
axis	1-dim. translational flange that drives the prismatic joint
bearing	1-dim. translational flange of the drive bearing of the prismatic joint

## Modelica.Mechanics.MultiBody.Parts

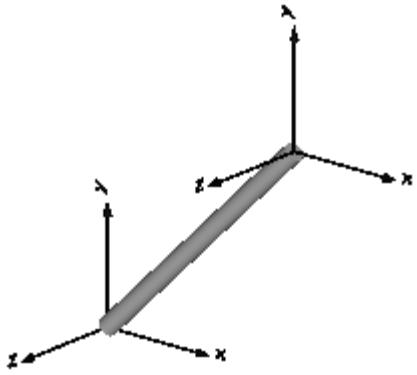
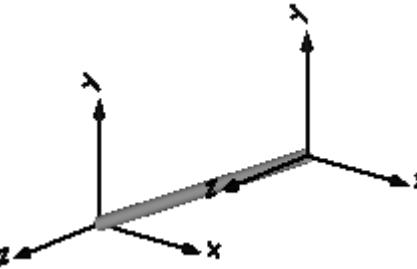
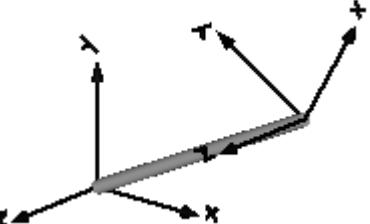
Rigid components such as bodies with mass and inertia and massless rods

## Information

Package **Parts** contains **rigid components** of a multi-body system. These components may be used to build up more complicated structures. For example, a part may be built up of a "Body" and of several "FixedTranslation" components.

## Content

Model	Description
Fixed	Frame fixed in world frame at a given position. It is visualized with a shape, see <b>shapeType</b> below (the frames on the two sides do not belong to the component):

	
<b>FixedTranslation</b>	Fixed translation of frame_b with respect to frame_a. It is visualized with a shape, see <b>shapeType</b> below (the frames on the two sides do not belong to the component):  
<b>FixedRotation</b>	Fixed translation and fixed rotation of frame_b with respect to frame_a. It is visualized with a shape, see <b>shapeType</b> below (the frames on the two sides do not belong to the component):  
<b>Body</b>	Rigid body with mass, inertia tensor and one frame connector. It is visualized with a cylinder and a sphere at the center of mass:  
<b>BodyShape</b>	Rigid body with mass, inertia tensor, different shapes (see <b>shapeType</b> below) for animation, and two frame connectors:  
<b>Fixed BodyBox</b>	Rigid body with box shape (mass and animation properties are computed from box data and from density):

	
<b>BodyCylinder</b>	Rigid body with cylinder shape (mass and animation properties are computed from cylinder data and from density): 
<b>PointMass</b>	Rigid body where inertia tensor and rotation is neglected: 
<b>Mounting1D</b>	Propagate 1-dim. support torque to 3-dim. system
<b>Rotor1D</b>	1D inertia attachable on 3-dim. bodies (without neglecting dynamic effects) 
<b>BevelGear1D</b>	1D gearbox with arbitrary shaft directions (3D bearing frame)

Components **Fixed**, **FixedTranslation**, **FixedRotation** and **BodyShape** are visualized according to parameter **shapeType**, that may have the following values (e.g., **shapeType = "box"**):



All the details of the visualization shape parameters are given in [Visualizers.FixedShape](#)

Colors in all animation parts are defined via parameter **color**. This is an Integer vector with 3 elements, {r, g, b}, and specifies the color of the shape. {r,g,b} are the "red", "green" and "blue" color parts, given in the ranges 0 .. 255, respectively. The predefined type **MultiBody.Types.Color** contains a menu definition of the colors used in the MultiBody library (this will be replaced by a color editor).

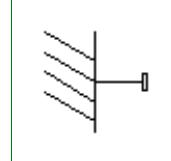
## Package Content

Name	Description
 <b>Fixed</b>	Frame fixed in the world frame at a given position
 <b>FixedTranslation</b>	Fixed translation of frame_b with respect to frame_a
 <b>FixedRotation</b>	Fixed translation followed by a fixed rotation of frame_b with respect to frame_a
 <b>Body</b>	Rigid body with mass, inertia tensor and one frame connector (12 potential states)
 <b>BodyShape</b>	Rigid body with mass, inertia tensor, different shapes for animation, and two frame connectors (12 potential states)
 <b>BodyBox</b>	Rigid body with box shape. Mass and animation properties are computed from box data and density (12 potential states)
 <b>BodyCylinder</b>	Rigid body with cylinder shape. Mass and animation properties are computed from cylinder data and density (12 potential states)
 <b>PointMass</b>	Rigid body where body rotation and inertia tensor is neglected (6 potential states)
 <b>Mounting1D</b>	Propagate 1-dim. support torque to 3-dim. system (provided <code>world.driveTrainMechanics3D=true; default=false</code> )
 <b>Rotor1D</b>	1D inertia attachable on 3-dim. bodies (3D dynamic effects are taken into account if

	<code>world.driveTrainMechanics3D=true; default=false)</code>
 BevelGear1D	1D gearbox with arbitrary shaft directions and 3-dim. bearing frame (3D dynamic effects are taken into account provided <code>world.driveTrainMechanics3D=true; default=false</code> )

## Modelica.Mechanics.MultiBody.Parts.Fixed

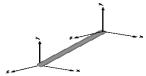
Frame fixed in the world frame at a given position



### Information

Element consisting of a frame (`frame_b`) that is fixed in the world frame at a given position defined by parameter vector `r` (vector from origin of world frame to `frame_b`, resolved in the world frame).

By default, this component is visualized by a cylinder connecting the world frame and `frame_b` of this components, as shown in the figure below. Note, that the visualized world frame on the left side and `Fixed.frame_b` on the right side are not part of the component animation and that the animation may be switched off via parameter animation = `false`.



### Parameters

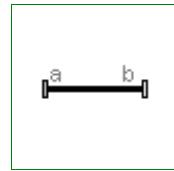
Name	Default	Description
animation	true	= true, if animation shall be enabled
<code>r[3]</code>	{0,0,0}	Position vector from world frame to <code>frame_b</code> , resolved in world frame [m]
<b>Animation</b>		
if <code>animation = true</code>		
<code>shapeType</code>	"cylinder"	Type of shape
<code>r_shape[3]</code>	{0,0,0}	Vector from world frame to shape origin, resolved in world frame [m]
<code>lengthDirection[3]</code>	<code>r - r_shape</code>	Vector in length direction of shape, resolved in world frame [m]
<code>widthDirection[3]</code>	{0,1,0}	Vector in width direction of shape, resolved in world frame [m]
<code>length</code>	<code>Frames.length(r - r_shape)</code>	Length of shape [m]
<code>width</code>	<code>length/world.defaultWidthFra...</code>	Width of shape [m]
<code>height</code>	<code>width</code>	Height of shape [m]
<code>extra</code>	0.0	Additional parameter for cone, pipe etc. (see docu of Visualizers.Advanced.Shape)
<code>color</code>	Modelica.Mechanics.MultiBody..	Color of shape
<code>specularCoefficient</code>	<code>world.defaultSpecularCoeffic...</code>	Reflection of ambient light (= 0: light is completely absorbed)

### Connectors

Name	Description
<code>frame_b</code>	Coordinate system fixed in the world frame

**Modelica.Mechanics.MultiBody.Parts.FixedTranslation**

Fixed translation of frame\_b with respect to frame\_a

**Information**

Component for a **fixed translation** of frame\_b with respect to frame\_a, i.e., the relationship between connectors frame\_a and frame\_b remains constant and frame\_a is always **parallel** to frame\_b.

By default, this component is visualized by a cylinder connecting frame\_a and frame\_b, as shown in the figure below. Note, that the two visualized frames are not part of the component animation and that the animation may be switched off via parameter animation = **false**.

**Parameters**

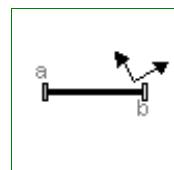
Name	Default	Description
animation	true	= true, if animation shall be enabled
r[3]	{0,0,0}	Vector from frame_a to frame_b resolved in frame_a [m]
<b>Animation</b>		
if animation = true		
shapeType	"cylinder"	Type of shape
r_shape[3]	{0,0,0}	Vector from frame_a to shape origin, resolved in frame_a [m]
lengthDirection	r - r_shape	Vector in length direction of shape, resolved in frame_a
widthDirection	{0,1,0}	Vector in width direction of shape, resolved in frame_a
length	Frames.length(r - r_shape)	Length of shape [m]
width	length/world.defaultWidthFra...	Width of shape [m]
height	width	Height of shape. [m]
extra	0.0	Additional parameter depending on shapeType (see docu of Visualizers.Advanced.Shape).
color	Modelica.Mechanics.MultiBody..	Color of shape
specularCoefficient	world.defaultSpecularCoeffic...	Reflection of ambient light (= 0: light is completely absorbed)

**Connectors**

Name	Description
frame_a	Coordinate system fixed to the component with one cut-force and cut-torque
frame_b	Coordinate system fixed to the component with one cut-force and cut-torque

**Modelica.Mechanics.MultiBody.Parts.FixedRotation**

Fixed translation followed by a fixed rotation of frame\_b with respect to frame\_a

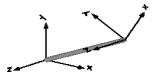


## Information

Component for a **fixed translation** and **fixed rotation** of frame\_b with respect to frame\_a, i.e., the relationship between connectors frame\_a and frame\_b remains constant. There are several possibilities to define the orientation of frame\_b with respect to frame\_a:

- **Planar rotation** along axis 'n' (that is fixed and resolved in frame\_a) with a fixed angle 'angle'.
- **Vectors n\_x and n\_y** that are directed along the corresponding axes direction of frame\_b and are resolved in frame\_a (if n\_y is not orthogonal to n\_x, the y-axis of frame\_b is selected such that it is orthogonal to n\_x and in the plane of n\_x and n\_y).
- **Sequence of three planar axes rotations**. For example, "sequence = {1,2,3}" and "angles = {90, 45, -90}" means to rotate frame\_a around the x axis with 90 degrees, around the new y axis with 45 degrees and around the new z axis around -90 degrees to arrive at frame\_b. Note, that sequence={1,2,3} is the Cardan angle sequence and sequence = {3,1,3} is the Euler angle sequence.

By default, this component is visualized by a cylinder connecting frame\_a and frame\_b, as shown in the figure below. In this figure frame\_b is rotated along the z-axis of frame\_a with 60 degree. Note, that the two visualized frames are not part of the component animation and that the animation may be switched off via parameter animation = **false**.



## Parameters

Name	Default	Description
animation	true	= true, if animation shall be enabled
r[3]	{0,0,0}	Vector from frame_a to frame_b resolved in frame_a [m]
rotationType	Modelica.Mechanics.MultiBody..	Type of rotation description
<b>if rotationType = RotationAxis</b>		
n	{1,0,0}	Axis of rotation in frame_a (= same as in frame_b)
angle	0	Angle to rotate frame_a around axis n into frame_b [deg]
<b>if rotationType = TwoAxesVectors</b>		
n_x	{1,0,0}	Vector along x-axis of frame_b resolved in frame_a
n_y	{0,1,0}	Vector along y-axis of frame_b resolved in frame_a
<b>if rotationType = PlanarRotationSequence</b>		
sequence	{1,2,3}	Sequence of rotations
angles[3]	{0,0,0}	Rotation angles around the axes defined in 'sequence' [deg]
<b>Animation</b>		
<b>if animation = true</b>		
shapeType	"cylinder"	Type of shape
r_shape[3]	{0,0,0}	Vector from frame_a to shape origin, resolved in frame_a [m]
lengthDirection	r - r_shape	Vector in length direction of shape, resolved in frame_a
widthDirection	{0,1,0}	Vector in width direction of shape, resolved in frame_a
length	Frames.length(r - r_shape)	Length of shape [m]
width	length/world.defaultWidthFra...	Width of shape [m]

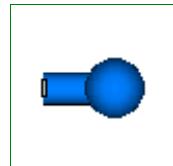
height	width	Height of shape. [m]
extra	0.0	Additional parameter depending on shapeType (see docu of Visualizers.Advanced.Shape).
color	Modelica.Mechanics.MultiBody.. .	Color of shape
specularCoefficient	world.defaultSpecularCoeffic...	Reflection of ambient light (= 0: light is completely absorbed)

## Connectors

Name	Description
frame_a	Coordinate system fixed to the component with one cut-force and cut-torque
frame_b	Coordinate system fixed to the component with one cut-force and cut-torque

## Modelica.Mechanics.MultiBody.Parts.Body

Rigid body with mass, inertia tensor and one frame connector (12 potential states)



## Information

**Rigid body** with mass and inertia tensor. All parameter vectors have to be resolved in frame\_a. The **inertia tensor** has to be defined with respect to a coordinate system that is parallel to frame\_a with the origin at the center of mass of the body.

By default, this component is visualized by a **cylinder** located between frame\_a and the center of mass and by a **sphere** that has its center at the center of mass. If the cylinder length is smaller as the radius of the sphere, e.g., since frame\_a is located at the center of mass, the cylinder is not displayed. Note, that the animation may be switched off via parameter animation = **false**.



## States of Body Components

Every body has potential states. If possible a tool will select the states of joints and not the states of bodies because this is usually the most efficient choice. In this case the position, orientation, velocity and angular velocity of frame\_a of the body will be computed by the component that is connected to frame\_a. However, if a body is moving freely in space, variables of the body have to be used as states. The potential states of the body are:

- The **position vector** frame\_a.r\_0 from the origin of the world frame to the origin of frame\_a of the body, resolved in the world frame and the **absolute velocity** v\_0 of the origin of frame\_a, resolved in the world frame (= der(frame\_a.r\_0)).
- If parameter **useQuaternions** in the "Advanced" menu is **true** (this is the default), then **4 quaternions** are potential states. Additionally, the coordinates of the absolute angular velocity vector of the body are 3 potential states.  
If **useQuaternions** in the "Advanced" menu is **false**, then **3 angles** and the derivatives of these angles are potential states. The orientation of frame\_a is computed by rotating the world frame along the axes defined in parameter vector "sequence\_angleStates" (default = {1,2,3}, i.e., the Cardan angle sequence) around the angles used as potential states. For example, the default is to rotate the x-axis of the world frame around angles[1], the new y-axis around angles[2] and the new z-axis around angles[3], arriving at frame\_a.

The quaternions have the slight disadvantage that there is a non-linear constraint equation between the 4 quaternions. Therefore, at least one non-linear equation has to be solved during simulation. A tool might, however, analytically solve this simple constraint equation. Using the 3 angles as states has the disadvantage that there is a singular configuration in which a division by zero will occur. If it is possible to

determine in advance for an application class that this singular configuration is outside of the operating region, the 3 angles might be used as potential states by setting **useQuaternions = false**.

In text books about 3-dimensional mechanics often 3 angles and the angular velocity are used as states. This is not the case here, since 3 angles and their derivatives are used as potential states (if **useQuaternions = false**). The reason is that for real-time simulation the discretization formula of the integrator might be "inlined" and solved together with the body equations. By appropriate symbolic transformation the performance is drastically increased if angles and their derivatives are used as states, instead of angles and the angular velocity.

Whether or not variables of the body are used as states is usually automatically selected by the Modelica translator. If parameter **enforceStates** is set to **true** in the "Advanced" menu, then body variables are forced to be used as states according to the setting of parameters "useQuaternions" and "sequence\_angleStates".

## Parameters

Name	Default	Description
animation	true	= true, if animation shall be enabled (show cylinder and sphere)
r_CM[3]	{0,0,0}	Vector from frame_a to center of mass, resolved in frame_a [m]
m	1	Mass of rigid body [kg]
<b>Inertia tensor (resolved in center of mass, parallel to frame_a)</b>		
I_11	0.001	(1,1) element of inertia tensor [kg.m <sup>2</sup> ]
I_22	0.001	(2,2) element of inertia tensor [kg.m <sup>2</sup> ]
I_33	0.001	(3,3) element of inertia tensor [kg.m <sup>2</sup> ]
I_21	0	(2,1) element of inertia tensor [kg.m <sup>2</sup> ]
I_31	0	(3,1) element of inertia tensor [kg.m <sup>2</sup> ]
I_32	0	(3,2) element of inertia tensor [kg.m <sup>2</sup> ]
<b>Initialization</b>		
initType	Modelica.Mechanics.MultiBody..	Type of initialization (defines usage of start values below)
r_0_start[3]	{0,0,0}	Initial values of frame_a.r_0 (vector from origin of world frame to origin of frame_a resolved in world frame) [m]
sequence_start	{1,2,3}	Sequence of rotations to rotate world frame into frame_a at initial time
angles_start[3]	{0,0,0}	Initial values of angles to rotate world frame around 'sequence_start' axes into frame_a [deg]
v_0_start[3]	{0,0,0}	Initial values of velocity v_0 = der(frame_a.r_0) [m/s]
w_0_start[3]	{0,0,0}	Initial values of angular velocity of frame_a resolved in world frame [deg/s]
a_0_start[3]	{0,0,0}	Initial values of acceleration a_0 = der(v_0) [m/s <sup>2</sup> ]
z_0_start[3]	{0,0,0}	Initial values of angular acceleration z_0 = der(w_0) [deg/s <sup>2</sup> ]
<b>Animation</b>		
if animation = true		
sphereDiameter	world.defaultBodyDiameter	Diameter of sphere [m]
sphereColor	Modelica.Mechanics.MultiBody..	Color of sphere
cylinderDiameter	sphereDiameter/Types.Default...	Diameter of cylinder [m]
cylinderColor	sphereColor	Color of cylinder

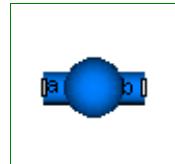
specularCoefficient	world.defaultSpecularCoefficient	Reflection of ambient light (= 0: light is completely absorbed)
<b>Advanced</b>		
enforceStates	false	= true, if absolute variables of body object shall be used as states (StateSelect.always)
useQuaternions	true	= true, if quaternions shall be used as potential states otherwise use 3 angles as potential states
sequence_angleStates	{1,2,3}	Sequence of rotations to rotate world frame into frame_a around the 3 angles used as potential states

## Connectors

Name	Description
frame_a	Coordinate system fixed at body

## Modelica.Mechanics.MultiBody.Parts.BodyShape

Rigid body with mass, inertia tensor, different shapes for animation, and two frame connectors (12 potential states)



## Information

**Rigid body** with mass and inertia tensor and **two frame connectors**. All parameter vectors have to be resolved in frame\_a. The **inertia tensor** has to be defined with respect to a coordinate system that is parallel to frame\_a with the origin at the center of mass of the body. The coordinate system **frame\_b** is always parallel to **frame\_a**.

By default, this component is visualized by any **shape** that can be defined with Modelica.Mechanics.MultiBody.Visualizers.FixedShape. This shape is placed between frame\_a and frame\_b (default: length(shape) = Frames.length(r)). Additionally a **sphere** may be visualized that has its center at the center of mass. Note, that the animation may be switched off via parameter animation = **false**.



The following shapes can be defined via parameter **shapeType**, e.g., **shapeType="cone"**:



A BodyShape component has potential states. For details of these states and of the "Advanced" menu parameters, see model [MultiBody.Parts.Body](#).

## Parameters

Name	Default	Description
animation	true	= true, if animation shall be enabled (show shape between frame_a and frame_b and optionally a sphere at the center of mass)
animateSphere	true	= true, if mass shall be animated as sphere provided animation=true
r[3]	{0,0,0}	Vector from frame_a to frame_b resolved in frame_a [m]
r_CM[3]	{0,0,0}	Vector from frame_a to center of mass, resolved in frame_a [m]

## 550 Modelica.Mechanics.MultiBody.Parts.BodyShape

m	1	Mass of rigid body [kg]
<b>Inertia tensor (resolved in center of mass, parallel to frame_a)</b>		
I_11	0.001	(1,1) element of inertia tensor [kg.m <sup>2</sup> ]
I_22	0.001	(2,2) element of inertia tensor [kg.m <sup>2</sup> ]
I_33	0.001	(3,3) element of inertia tensor [kg.m <sup>2</sup> ]
I_21	0	(2,1) element of inertia tensor [kg.m <sup>2</sup> ]
I_31	0	(3,1) element of inertia tensor [kg.m <sup>2</sup> ]
I_32	0	(3,2) element of inertia tensor [kg.m <sup>2</sup> ]
<b>Initialization</b>		
initType	Modelica.Mechanics.MultiBody..	Type of initialization (defines usage of start values below)
r_0_start[3]	{0,0,0}	Initial values of frame_a.r_0 (vector from origin of world frame to origin of frame_a resolved in world frame) [m]
sequence_start	{1,2,3}	Sequence of rotations to rotate world frame into frame_a at initial time
angles_start[3]	{0,0,0}	Initial values of angles to rotate world frame around 'sequence_start' axes into frame_a [deg]
v_0_start[3]	{0,0,0}	Initial values of velocity v_0 = der(frame_a.r_0) [m/s]
w_0_start[3]	{0,0,0}	Initial values of angular velocity of frame_a resolved in world frame [deg/s]
a_0_start[3]	{0,0,0}	Initial values of acceleration a_0 = der(v_0) [m/s <sup>2</sup> ]
z_0_start[3]	{0,0,0}	Initial values of angular acceleration z_0 = der(w_0) [deg/s <sup>2</sup> ]
<b>Animation</b>		
if animation = true		
shapeType	"cylinder"	Type of shape
r_shape[3]	{0,0,0}	Vector from frame_a to shape origin, resolved in frame_a [m]
lengthDirection	r - r_shape	Vector in length direction of shape, resolved in frame_a
widthDirection	{0,1,0}	Vector in width direction of shape, resolved in frame_a
length	Frames.length(r - r_shape)	Length of shape [m]
width	length/world.defaultWidthFra...	Width of shape [m]
height	width	Height of shape. [m]
extra	0.0	Additional parameter depending on shapeType (see docu of Visualizers.Advanced.Shape).
color	Modelica.Mechanics.MultiBody..	Color of shape
specularCoefficient	world.defaultSpecularCoeffic...	Reflection of ambient light (= 0: light is completely absorbed)
if animation = true and animateSphere = true		
sphereDiameter	2*width	Diameter of sphere [m]
sphereColor	color	Color of sphere of mass
<b>Advanced</b>		
enforceStates	false	= true, if absolute variables of body object shall be used as states (StateSelect.always)
useQuaternions	true	= true, if quaternions shall be used as potential states

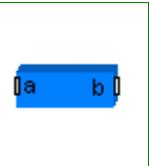
		otherwise use 3 angles as potential states
sequence_angleStates	{1,2,3}	Sequence of rotations to rotate world frame into frame_a around the 3 angles used as potential states

## Connectors

Name	Description
frame_a	Coordinate system fixed to the component with one cut-force and cut-torque
frame_b	Coordinate system fixed to the component with one cut-force and cut-torque

## Modelica.Mechanics.MultiBody.Parts.BodyBox

Rigid body with box shape. Mass and animation properties are computed from box data and density (12 potential states)



## Information

**Rigid body** with **box** shape. The mass properties of the body (mass, center of mass, inertia tensor) are computed from the box data. Optionally, the box may be hollow. The (outer) box shape is by default used in the animation. The hollow part is not shown in the animation. The two connector frames **frame\_a** and **frame\_b** are always parallel to each other. Example of component animation (note, that the animation may be switched off via parameter animation = **false**):



A BodyBox component has potential states. For details of these states and of the "Advanced" menu parameters, see model [MultiBody.Parts.Body](#).

## Parameters

Name	Default	Description
animation	true	= true, if animation shall be enabled (show box between frame_a and frame_b)
r[3]	{0,1,0,0}	Vector from frame_a to frame_b resolved in frame_a [m]
r_shape[3]	{0,0,0}	Vector from frame_a to box origin, resolved in frame_a [m]
lengthDirection	r - r_shape	Vector in length direction of box, resolved in frame_a
widthDirection	{0,1,0}	Vector in width direction of box, resolved in frame_a
length	Frames.length(r - r_shape)	Length of box [m]
width	length/world.defaultWidthFra...	Width of box [m]
height	width	Height of box [m]
innerWidth	0	Width of inner box surface ( $0 \leq \text{innerWidth} \leq \text{width}$ ) [m]
innerHeight	innerWidth	Height of inner box surface ( $0 \leq \text{innerHeight} \leq \text{height}$ ) [m]
density	7.7	Density of box (e.g., steel: 7.7 .. 7.9, wood : 0.4 .. 0.8) [g/cm <sup>3</sup> ]
color	Modelica.Mechanics.MultiBody..	Color of box
specularCoefficient	world.defaultSpecularCoeffic...	Reflection of ambient light (= 0: light is completely

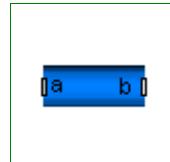
		absorbed)
<b>Initialization</b>		
initType	Modelica.Mechanics.MultiBody.. . .	Type of initialization (defines usage of start values below)
r_0_start[3]	{0,0,0}	Initial values of frame_a.r_0 (vector from origin of world frame to origin of frame_a resolved in world frame) [m]
sequence_start	{1,2,3}	Sequence of rotations to rotate world frame into frame_a at initial time
angles_start[3]	{0,0,0}	Initial values of angles to rotate world frame around 'sequence_start' axes into frame_a [deg]
v_0_start[3]	{0,0,0}	Initial values of velocity v_0 = der(frame_a.r_0) [m/s]
w_0_start[3]	{0,0,0}	Initial values of angular velocity of frame_a resolved in world frame [deg/s]
a_0_start[3]	{0,0,0}	Initial values of acceleration a_0 = der(v_0) [m/s <sup>2</sup> ]
z_0_start[3]	{0,0,0}	Initial values of angular acceleration z_0 = der(w_0) [deg/s <sup>2</sup> ]
<b>Advanced</b>		
enforceStates	false	= true, if absolute variables of body object shall be used as states (StateSelect.always)
useQuaternions	true	= true, if quaternions shall be used as potential states otherwise use 3 angles as potential states
sequence_angleStates	{1,2,3}	Sequence of rotations to rotate world frame into frame_a around the 3 angles used as potential states

## Connectors

Name	Description
frame_a	Coordinate system fixed to the component with one cut-force and cut-torque
frame_b	Coordinate system fixed to the component with one cut-force and cut-torque

## Modelica.Mechanics.MultiBody.Parts.BodyCylinder

Rigid body with cylinder shape. Mass and animation properties are computed from cylinder data and density (12 potential states)



## Information

Rigid body with cylinder shape. The mass properties of the body (mass, center of mass, inertia tensor) are computed from the cylinder data. Optionally, the cylinder may be hollow. The cylinder shape is by default used in the animation. The two connector frames **frame\_a** and **frame\_b** are always parallel to each other. Example of component animation (note, that the animation may be switched off via parameter animation = false):



A BodyCylinder component has potential states. For details of these states and of the "Advanced" menu parameters, see model [MultiBody.Parts.Body](#).

## Parameters

Name	Default	Description
animation	true	= true, if animation shall be enabled (show cylinder between frame_a and frame_b)
r[3]	{0.1,0,0}	Vector from frame_a to frame_b, resolved in frame_a [m]
r_shape[3]	{0,0,0}	Vector from frame_a to cylinder origin, resolved in frame_a [m]
lengthDirection	r - r_shape	Vector in length direction of cylinder, resolved in frame_a
length	Frames.length(r - r_shape)	Length of cylinder [m]
diameter	length/world.defaultWidthFra...	Diameter of cylinder [m]
innerDiameter	0	Inner diameter of cylinder (0 <= innerDiameter <= Diameter) [m]
density	7.7	Density of cylinder (e.g., steel: 7.7 .. 7.9, wood : 0.4 .. 0.8) [g/cm3]
color	Modelica.Mechanics.MultiBody.. .	Color of cylinder
specularCoefficient	world.defaultSpecularCoeffic...	Reflection of ambient light (= 0: light is completely absorbed)

## Initialization

initType	Modelica.Mechanics.MultiBody.. .	Type of initialization (defines usage of start values below)
r_0_start[3]	{0,0,0}	Initial values of frame_a.r_0 (vector from origin of world frame to origin of frame_a resolved in world frame) [m]
sequence_start	{1,2,3}	Sequence of rotations to rotate world frame into frame_a at initial time
angles_start[3]	{0,0,0}	Initial values of angles to rotate world frame around 'sequence_start' axes into frame_a [deg]
v_0_start[3]	{0,0,0}	Initial values of velocity v_0 = der(frame_a.r_0) [m/s]
w_0_start[3]	{0,0,0}	Initial values of angular velocity of frame_a resolved in world frame [deg/s]
a_0_start[3]	{0,0,0}	Initial values of acceleration a_0 = der(v_0) [m/s2]
z_0_start[3]	{0,0,0}	Initial values of angular acceleration z_0 = der(w_0) [deg/s2]

## Advanced

enforceStates	false	= true, if absolute variables of body object shall be used as states (StateSelect.always)
useQuaternions	true	= true, if quaternions shall be used as potential states otherwise use 3 angles as potential states
sequence_angleStates	{1,2,3}	Sequence of rotations to rotate world frame into frame_a around the 3 angles used as potential states

## Connectors

Name	Description
frame_a	Coordinate system fixed to the component with one cut-force and cut-torque
frame_b	Coordinate system fixed to the component with one cut-force and cut-torque

## Modelica.Mechanics.MultiBody.Parts.PointMass

Rigid body where body rotation and inertia tensor is neglected (6 potential states)



### Information

**Rigid body** where the inertia tensor is neglected. This body is solely defined by its mass. By default, this component is visualized by a **sphere** that has its center at frame\_a. Note, that the animation may be switched off via parameter animation = **false**.

Every PointMass has potential states. If possible a tool will select the states of joints and not the states of PointMasses because this is usually the most efficient choice. In this case the position and velocity of frame\_a of the body will be computed by the component that is connected to frame\_a. However, if a PointMass is moving freely in space, variables of the PointMass have to be used as states. The potential states are: The **position vector** frame\_a.r\_0 from the origin of the world frame to the origin of frame\_a of the body, resolved in the world frame and the **absolute velocity** v\_0 of the origin of frame\_a, resolved in the world frame (= der(frame\_a.r\_0)).

Whether or not variables of the body are used as states is usually automatically selected by the Modelica translator. If parameter **enforceStates** is set to **true** in the "Advanced" menu, then PointMass variables frame\_a.r\_0 and der(frame\_a.r\_0) are forced to be used as states.

### Parameters

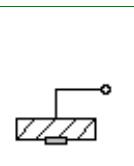
Name	Default	Description
animation	true	= true, if animation shall be enabled (show sphere)
m		Mass of mass point [kg]
<b>Initialization</b>		
initType	Modelica.Mechanics.MultiBody.. .	Type of initialization (defines usage of start values below)
r_0_start[3]	{0,0,0}	Initial values of frame_a.r_0 (vector from origin of world frame to origin of frame_a resolved in world frame) [m]
v_0_start[3]	{0,0,0}	Initial values of velocity v_0 = der(frame_a.r_0) [m/s]
a_0_start[3]	{0,0,0}	Initial values of acceleration a_0 = der(v_0) [m/s <sup>2</sup> ]
<b>Animation</b>		
if animation = true		
sphereDiameter	world.defaultBodyDiameter	Diameter of sphere [m]
sphereColor	Modelica.Mechanics.MultiBody.. .	Color of sphere
specularCoefficient	world.defaultSpecularCoeffic...	Reflection of ambient light (= 0: light is completely absorbed)
<b>Advanced</b>		
enforceStates	false	= true, if frame_a.r_0 and v_0 of body object shall be used as states (StateSelect.always)

### Connectors

Name	Description
frame_a	Coordinate system fixed at center of mass point

## Modelica.Mechanics.MultiBody.Parts.Mounting1D

Propagate 1-dim. support torque to 3-dim. system (provided)



`world.driveTrainMechanics3D=true; default=false)`

## Information

This component is used to acquire support torques from a 1-dim.-rotational mechanical system (e.g., components from Modelica.Mechanics.Rotational) and to propagate them to a carrier body.

The 1-dim. support torque at `flange_b` is transformed into 3-dim. space under consideration of the rotation axis, parameter `n`, which has to be given in the local coordinate system of `frame_a`.

All components of a 1-dim.-rotational mechanical system that are connected to a common **Mounting1D** element need to have the same axis of rotation along parameter vector `n`. This means that, e.g., bevel gears where the axis of rotation of `flange_a` and `flange_b` are different cannot be described properly by connecting to the **Mounting1D** component. In this case, a combination of several **Mounting1D** components or the component **BevelGear1D** should be used.

## Reference

SCHWEIGER, Christian ; OTTER, Martin: *Modelling 3D Mechanical Effects of 1-dim. Powertrains*. In: *Proceedings of the 3rd International Modelica Conference*. Linköping : The Modelica Association and Linköping University, November 3-4, 2003, pp. 149-158

## Parameters

Name	Default	Description
<code>phi0</code>	0	Fixed offset angle of housing [rad]
<code>n</code>	{1,0,0}	Axis of rotation = axis of support torque (resolved in frame_a)

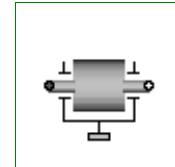
## Connectors

Name	Description
<code>flange_b</code> (right) flange fixed in housing	
<code>frame_a</code>	Frame in which housing is fixed (connector is removed, if <code>world.driveTrainMechanics3D=false</code> )

---

## Modelica.Mechanics.MultiBody.Parts.Rotor1D

1D inertia attachable on 3-dim. bodies (3D dynamic effects are taken into account if `world.driveTrainMechanics3D=true; default=false`)



## Information

This component is used to model the gyroscopic torques exerted by a 1-dim. inertia (so called *rotor*) on its 3-dim. carrier body. Gyroscopic torques appear, if the vector of the carrier body's angular velocity is not parallel to the vector of the rotor's. The axis of rotation of the rotor is defined by the parameter `n`, which has to be given in the local coordinate system of `frame_a`. The default animation of this component is shown in the figure below.



This component is a replacement for [Modelica.Mechanics.Rotational.Inertia](#) for the case, that a 1-dim.-rotational mechanical system should be attached with a 3-dim. carrier body.

The Boolean parameter `exact` was introduced due to performance reasons. If `exact` is set to true, the influence of the carrier body motion on the angular velocity of the rotor is neglected. This influence is usually negligible if the 1-dim.-rotational mechanical system accelerates much faster as the base body (this is, e.g., the case in vehicle powertrains). The essential advantage is that an algebraic loop is removed since then there is only an action on acceleration level from the powertrain to the base body but not vice versa.

## 556 Modelica.Mechanics.MultiBody.Parts.Rotor1D

### Reference

SCHWEIGER, Christian ; OTTER, Martin: Modelling 3D Mechanical Effects of 1-dim. Powertrains. In: *Proceedings of the 3rd International Modelica Conference*. Linköping : The Modelica Association and Linköping University, November 3-4, 2003, pp. 149-158

### Parameters

Name	Default	Description
animation	true	= true, if animation shall be enabled (show rotor as cylinder)
J	1	Moment of inertia of rotor around its axis of rotation [kg.m <sup>2</sup> ]
n	{1,0,0}	Axis of rotation resolved in frame_a
<b>Initialization</b>		
initType	Modelica.Mechanics.MultiBody..	Type of initialization (defines usage of start values below)
phi_start	0	Initial value of rotor rotation angle phi (fixed or guess value) [deg]
w_start	0	Initial value of relative angular velocity w = der(phi) [deg/s]
a_start	0	Initial value of relative angular acceleration a = der(w) [deg/s <sup>2</sup> ]
<b>Animation</b>		
if animation = true		
r_center[3]	zeros(3)	Position vector from origin of frame_a to center of cylinder [m]
cylinderLength	2*world.defaultJointLength	Length of cylinder representing the rotor [m]
cylinderDiameter	2*world.defaultJointWidth	Diameter of cylinder representing the rotor [m]
cylinderColor	Modelica.Mechanics.MultiBody..	Color of cylinder representing the rotor
specularCoefficient	world.defaultSpecularCoeffic...	Reflection of ambient light (= 0: light is completely absorbed)
<b>Advanced</b>		
enforceStates	false	= true, if rotor angle (phi) and rotor speed (w) shall be used as states
exact	true	= true, if exact calculations; false if influence of bearing on rotor acceleration is neglected to avoid an algebraic loop

### Connectors

Name	Description
flange_a	(left) driving flange (flange axis directed INTO cut plane)
flange_b	(right) driven flange (flange axis directed OUT OF cut plane)
frame_a	Frame in which rotor housing is fixed (connector is removed, if world.driveTrainMechanics3D=false)

## Modelica.Mechanics.MultiBody.Parts.BevelGear1D

1D gearbox with arbitrary shaft directions and 3-dim. bearing frame (3D dynamic effects are taken into account provided world.driveTrainMechanics3D=true; default=false)



## Information

This component is used to model a 1-dim. gearbox with non-parallel axes (defined by parameters `n_a`, `n_b`). A 3-dim. bearing frame is necessary to reflect the correct support torque, as the axes of rotation of `flange_a` and `flange_b` and the direction of the support torque vector are different in general.

Note: The name BevelGear1D is kept only for simplicity. Regardless, this component could be used to model any kind of gearbox with non-parallel axes.

## Reference

SCHWEIGER, Christian ; OTTER, Martin: *Modelling 3D Mechanical Effects of 1-dim. Powertrains*. In: *Proceedings of the 3rd International Modelica Conference*. Linköping : The Modelica Association and Linköping University, November 3-4, 2003, pp. 149-158

## Parameters

Name	Default	Description
ratio	1	Gear speed ratio
<code>n_a</code>	{1,0,0}	Axis of rotation of flange_a, resolved in frame_a
<code>n_b</code>	{1,0,0}	Axis of rotation of flange_b, resolved in frame_a

## Connectors

Name	Description
<code>flange_a</code>	
<code>flange_b</code>	
<code>frame_a</code>	Bearing frame

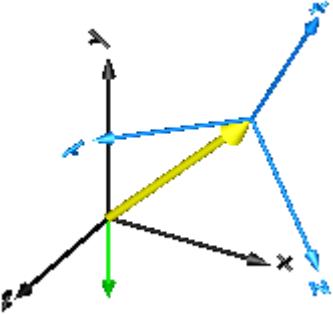
## Modelica.Mechanics.MultiBody.Sensors

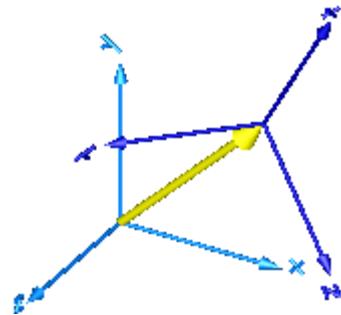
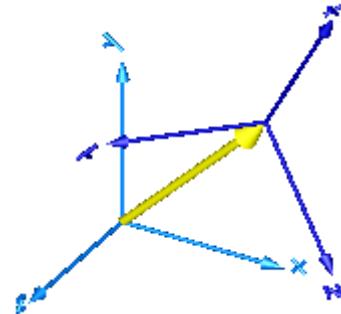
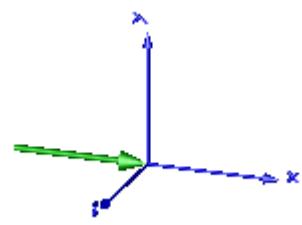
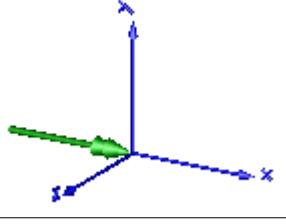
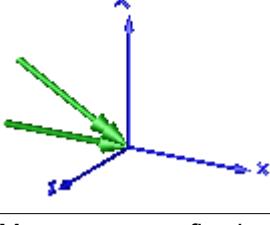
### Sensors to measure variables

## Information

Package **Sensors** contains **ideal measurement** components to determine absolute and relative kinematic quantities, as well as cut-forces and cut-torques. All measured quantities can be provided in every desired coordinate system.

## Content

Model	Description
<b>AbsoluteSensor</b>	Measure absolute kinematic quantities of a frame connector 
<b>RelativeSensor</b>	Measure relative kinematic quantities between two frame connectors

	
<b>Distance</b>	Measure distance between the origins of two frame connectors 
<b>CutForce</b>	Measure cut force vector 
<b>CutTorque</b>	Measure cut torque vector 
<b>CutForceAndTorque</b>	Measure cut force and cut torque vector 
<b>Power</b>	Measure power flowing from frame_a to frame_b

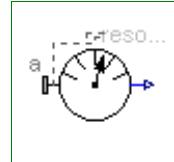
### Package Content

Name	Description
 <a href="#">AbsoluteSensor</a>	Measure absolute kinematic quantities of a frame connector
 <a href="#">RelativeSensor</a>	Measure relative kinematic quantities between two frame connectors
 <a href="#">Distance</a>	Measure the distance between the origins of two frame connectors

 CutForce	Measure cut force vector
 CutTorque	Measure cut torque vector
 CutForceAndTorque	Measure cut force and cut torque vector
 Power	Measure power flowing from frame_a to frame_b

## Modelica.Mechanics.MultiBody.Sensors.AbsoluteSensor

Measure absolute kinematic quantities of a frame connector



### Information

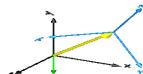
Absolute kinematic quantities of frame\_a are computed and provided at the output signal connector **y** in packed format in the order

1. absolute position vector (= r\_abs)
2. absolute velocity vector (= v\_abs)
3. absolute acceleration vector (= a\_abs)
4. 3 angles to rotate the world frame into frame\_a (= angles)
5. absolute angular velocity vector (= w\_abs)
6. absolute angular acceleration vector (= z\_abs)

For example, if parameters **get\_v** and **get\_w** are **true** and all other get\_XXX parameters are **false**, then **y** contains 6 elements:

```
y[1:3] = absolute velocity
y[4:6] = absolute angular velocity
```

In the following figure the animation of an AbsoluteSensor component is shown. The light blue coordinate system is frame\_a and the yellow arrow is the animated sensor.



If **frame\_resolve** is connected to another frame, then the provided absolute kinematic vectors are resolved in this frame. If **frame\_resolve** is **not** connected then the coordinate system in which the relative quantities are resolved is defined by parameter **resolveInFrame\_a**. If this parameter is **true**, then the provided kinematic vectors are resolved in frame\_a of this component. Otherwise, the kinematic vectors are resolved in the world frame. For example, if **frame\_resolve** is not connected and if **resolveInFrame\_a = false**, and **get\_v = true**, then

```
y = der(frame_a.r) // resolved in world frame
```

is returned, i.e., the derivative of the distance frame\_a.r\_0 from the origin of the world frame to the origin of frame\_a, resolved in the world frame.

Note, the cut-force and the cut-torque in frame\_resolve are always zero, whether frame\_resolve is connected or not.

If **get\_angles = true**, the 3 angles to rotate the world frame into frame\_a along the axes defined by parameter **sequence** are returned. For example, if sequence = {3,1,2} then the world frame is rotated around angles[1] along the z-axis, afterwards it is rotated around angles[2] along the x-axis, and finally it is rotated around angles[3] along the y-axis and is then identical to frame\_a. The 3 angles are returned in the range

```
-π <= angles[i] <= π
```

There are **two solutions** for "angles[1]" in this range. Via parameter **guessAngle1** (default = 0) the returned solution is selected such that |angles[1] - guessAngle1| is minimal. The transformation matrix between the

world frame and frame\_a may be in a singular configuration with respect to "sequence", i.e., there is an infinite number of angle values leading to the same transformation matrix. In this case, the returned solution is selected by setting angles[1] = guessAngle1. Then angles[2] and angles[3] can be uniquely determined in the above range.

Note, that parameter **sequence** has the restriction that only values 1,2,3 can be used and that sequence[1] ≠ sequence[2] and sequence[2] ≠ sequence[3]. Often used values are:

```
sequence = {1,2,3} // Cardan angle sequence
            = {3,1,3} // Euler angle sequence
            = {3,2,1} // Tait-Bryan angle sequence
```

Exact definition of the returned quantities:

1. r\_abs is vector frame\_a.r\_0, resolved according to table below.
2. v\_abs is vector **der**(frame\_a.r\_0), resolved according to table below.
3. a\_abs is vector **der**(**der**(frame\_a.r\_0)), resolved according to table below.
4. angles is a vector of 3 angles such that frame\_a.R = Frames.axesRotations(sequence, angles).
5. w\_abs is vector Modelica.Mechanics.Frames.angularVelocity1(frame\_a.R, **der**(frame\_a.R)), resolved according to table below.
6. z\_abs is vector **der**(w\_abs) (= derivative of absolute angular velocity of frame\_a with respect to the world frame, resolved according to table below).

<b>frame_resolve is</b>	<b>resolveInFrame_a =</b>	<b>vector is resolved in</b>
connected	true	<b>frame_resolve</b>
connected	false	<b>frame_resolve</b>
not connected	true	<b>frame_a</b>
not connected	false	<b>world frame</b>

## Parameters

Name	Default	Description
n_out	3*((if get_r_abs then 1 else...)	Number of output signals
animation	true	= true, if animation shall be enabled (show arrow)
resolvelnFrame_a	false	= true, if vectors are resolved in frame_a, otherwise in the world frame (if connector frame_resolve is connected, vectors are resolved in frame_resolve)
get_r_abs	true	= true, to measure the position vector from the origin of the world frame to the origin of frame_a in [m]
get_v_abs	false	= true, to measure the absolute velocity of the origin of frame_a in [m/s]
get_a_abs	false	= true, to measure the absolute acceleration of the origin of frame_a in [m/s^2]
get_angles	false	= true, to measure the 3 rotation angles to rotate the world frame into frame_a along the axes defined in 'sequence' below in [rad]
get_w_abs	false	= true, to measure the absolute angular velocity of frame_a in [rad/s]
get_z_abs	false	= true, to measure the absolute angular acceleration to frame_a in [rad/s^2]
if get_angles = true		
sequence	{1,2,3}	Angles are returned to rotate world frame around axes sequence[1], sequence[2] and finally sequence[3] into frame_a

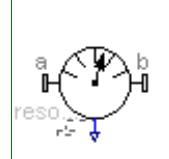
guessAngle1	0	Select angles[1] such that abs(angles[1] - guessAngle1) is a minimum [rad]
<b>Animation</b>		
if animation = true		
arrowDiameter	world.defaultArrowDiameter	Diameter of arrow from world frame to frame_a [m]
arrowColor	Modelica.Mechanics.MultiBody..	Color of arrow from world frame to frame_a
specularCoefficient	world.defaultSpecularCoeffic...	Reflection of ambient light (= 0: light is completely absorbed)

## Connectors

Name	Description
frame_a	Coordinate system from which absolute quantities are provided as output signals
y[n_out]	Measured data as signal vector
frame_resolve	If connected, the output signals are resolved in this frame

## Modelica.Mechanics.MultiBody.Sensors.RelativeSensor

Measure relative kinematic quantities between two frame connectors



## Information

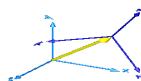
Relative kinematic quantities between frame\_a and frame\_b are determined and provided at the output signal connector **y** in packed format in the order

1. relative position vector (= r\_rel)
2. relative velocity vector (= v\_rel)
3. relative acceleration vector (= a\_rel)
4. 3 angles to rotate frame\_a into frame\_b (= angles)
5. relative angular velocity vector (= w\_rel)
6. relative angular acceleration vector (= z\_rel)

For example, if parameters **get\_v\_rel** and **get\_w\_rel** are **true** and all other get\_XXX parameters are **false**, then **y** contains 6 elements:

```
y = relative velocity
y = relative angular velocity
```

In the following figure the animation of a RelativeSensor component is shown. The light blue coordinate system is frame\_a, the dark blue coordinate system is frame\_b, and the yellow arrow is the animated sensor.



If parameter **resolvelnFrame\_a = true**, then the provided relative kinematic vectors of frame\_b with respect to frame\_a are resolved before differentiation in frame\_a. If this parameter is **false**, the relative kinematic vectors are resolved before differentiation in frame\_b. If **frame\_resolve** is connected to another frame, then the kinematic vector as defined above and/or its required derivatives are resolved in frame\_resolve. Note, derivatives of relative kinematic quantities are always performed with respect to frame\_a (**resolvelnFrame\_a = true**) or with respect to frame\_b (**resolvelnFrame\_a = false**). The resulting vector is then resolved in frame\_resolve, if this connector is connected.

For example, if **frame\_resolve** is not connected and if **resolvelnFrame\_a = false**, and **get\_v = true**, then

```
y = v_rel
= der(r_rel)
```

is returned ( $r_{\text{rel}} = \text{resolve2}(\text{frame\_b.R}, \text{frame\_b.r\_0} - \text{frame\_a.r\_0})$ ), i.e., the derivative of the relative distance from frame\_a to frame\_b, resolved in frame\_b. If frame\_resolve is connected, then

```
y = v_rel
= resolve2(frame_resolve.R, der(r_rel))
```

is returned, i.e., the previous relative velocity vector is additionally resolved in frame\_resolve.

Note, the cut-force and the cut-torque in frame\_resolve are always zero, whether frame\_resolve is connected or not.

If **get\_angles** = true, the 3 angles to rotate frame\_a into frame\_b along the axes defined by parameter **sequence** are returned. For example, if sequence = {3,1,2} then frame\_a is rotated around angles[1] along the z-axis, afterwards it is rotated around angles[2] along the x-axis, and finally it is rotated around angles[3] along the y-axis and is then identical to frame\_b. The 3 angles are returned in the range

```
-π <= angles[i] <= π
```

There are **two solutions** for "angles[1]" in this range. Via parameter **guessAngle1** (default = 0) the returned solution is selected such that |angles[1] - guessAngle1| is minimal. The relative transformation matrix between frame\_a and frame\_b may be in a singular configuration with respect to "sequence", i.e., there is an infinite number of angle values leading to the same relative transformation matrix. In this case, the returned solution is selected by setting angles[1] = guessAngle1. Then angles[2] and angles[3] can be uniquely determined in the above range.

Note, that parameter **sequence** has the restriction that only values 1,2,3 can be used and that sequence[1] ≠ sequence[2] and sequence[2] ≠ sequence[3]. Often used values are:

```
sequence = {1,2,3} // Cardan angle sequence
            = {3,1,3} // Euler angle sequence
            = {3,2,1} // Tait-Bryan angle sequence
```

Exact definition of the returned quantities (r\_rel\_ab, R\_rel\_ab, w\_rel\_ab are defined below the enumeration):

1. r\_rel is vector r\_rel\_ab, resolved according to table below.
2. v\_rel is vector **der**(r\_rel\_ab), resolved according to table below.
3. a\_rel is vector **der**(**der**(r\_rel\_ab)), resolved according to table below.
4. angles is a vector of 3 angles such that R\_rel\_ab = Frames.axesRotations(sequence, angles).
5. w\_rel is vector w\_rel\_ab, resolved according to table below.
6. z\_rel is vector **der**(w\_rel\_ab), resolved according to table below.

using the auxiliary quantities

1. r\_rel\_ab is vector frame\_b.r\_0 - frame\_a.r\_0, resolved either in frame\_a or frame\_b according to parameter resolvelnFrame\_a.
2. R\_rel\_ab is orientation object Frames.relativeRotation(frame\_a.R, frame\_b.R).
3. w\_rel\_ab is vector Frames.angularVelocity1(R\_rel\_ab, der(R\_rel\_ab)), resolved either in frame\_a or frame\_b according to parameter resolvelnFrame\_a.

and resolved in the following frame

frame_resolve is	resolvelnFrame_a =	vector is resolved in
connected	true	frame_resolve
connected	false	frame_resolve
not connected	true	frame_a
not connected	false	frame_b

## Parameters

Name	Default	Description

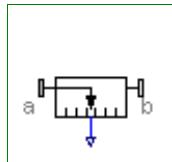
n_out	3*((if get_r_rel then 1 else...)	Number of output signals
animation	true	= true, if animation shall be enabled (show arrow)
resolveInFrame_a	true	= true, if relative vectors from frame_a to frame_b are resolved before differentiation in frame_a, otherwise in frame_b. If frame_resolve is connected, the vector and its derivatives are resolved in frame_resolve
get_r_rel	true	= true, to measure the relative position vector from the origin of frame_a to the origin of frame_b in [m]
get_v_rel	false	= true, to measure the relative velocity of the origin of frame_b with respect to frame_a in [m/s]
get_a_rel	false	= true, to measure the relative acceleration of the origin of frame_b with respect to frame_a in [m/s^2]
get_angles	false	= true, to measure the 3 rotation angles to rotate frame_a into frame_b along the axes defined in 'sequence' below in [rad]
get_w_rel	false	= true, to measure the relative angular velocity of frame_b with respect to frame_a in [rad/s]
get_z_rel	false	= true, to measure the relative angular acceleration of frame_b with respect to frame_a in [rad/s^2]
<b>if get_angles = true</b>		
sequence	{1,2,3}	Angles are returned to rotate frame_a around axes sequence[1], sequence[2] and finally sequence[3] into frame_b
guessAngle1	0	Select angles[1] such that abs(angles[1] - guessAngle1) is a minimum [rad]
<b>Animation</b>		
<b>if animation = true</b>		
arrowDiameter	world.defaultArrowDiameter	Diameter of relative arrow from frame_a to frame_b [m]
arrowColor	Modelica.Mechanics.MultiBody..	Color of relative arrow from frame_a to frame_b
specularCoefficient	world.defaultSpecularCoeffic...	Reflection of ambient light (= 0: light is completely absorbed)

## Connectors

Name	Description
frame_a	Coordinate system a
frame_b	Coordinate system b
y[n_out]	Measured data as signal vector
frame_resolve	If connected, the output signals are resolved in this frame

## Modelica.Mechanics.MultiBody.Sensors.Distance

Measure the distance between the origins of two frame connectors

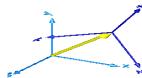


## Information

The **distance** between the origins of frame\_a and of frame\_b are determined and provided at the output signal connector **distance**. This distance is always positive. **Derivatives** of this signal can be easily obtained by connecting the block [Modelica.Blocks.Continuous.Der](#) to "distance" (this block performs analytic

differentiation of the input signal using the der(..) operator).

In the following figure the animation of a Distance sensor is shown. The light blue coordinate system is frame\_a, the dark blue coordinate system is frame\_b, and the yellow arrow is the animated sensor.



If the distance is smaller as parameter **s\_small** (in the "advanced" menu), it is approximated such that its derivative is finite for zero distance. Without such an approximation, the derivative would be infinite and a division by zero would occur. The approximation is performed in the following way: If distance > s\_small, it is computed as  $\sqrt{r \cdot r}$  where r is the position vector from the origin of frame\_a to the origin of frame\_b. If the distance becomes smaller as s\_small, the "sqrt()" function is approximated by a second order polynomial, such that the function value and its first derivative are identical for sqrt() and the polynomial at s\_small. Furthermore, the polynomial passes through zero. The effect is, that the distance function is continuous and differentiable everywhere. The derivative at zero distance is  $3/(2*s\_small)$ .

## Parameters

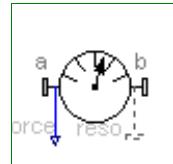
Name	Default	Description
animation	true	= true, if animation shall be enabled (show arrow)
<b>if animation = true</b>		
arrowDiameter	world.defaultArrowDiameter	Diameter of relative arrow from frame_a to frame_b [m]
arrowColor	Modelica.Mechanics.MultiBody...	Color of relative arrow from frame_a to frame_b
specularCoefficient	world.defaultSpecularCoeffic...	Reflection of ambient light (= 0: light is completely absorbed)
<b>Advanced</b>		
s_small	1.E-10	Prevent zero-division if distance between frame_a and frame_b is zero [m]

## Connectors

Name	Description
frame_a	Coordinate system fixed to the component with one cut-force and cut-torque
frame_b	Coordinate system fixed to the component with one cut-force and cut-torque
distance	Distance between the origin of frame_a and the origin of frame_b

## Modelica.Mechanics.MultiBody.Sensors.CutForce

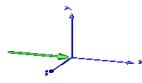
Measure cut force vector



## Information

The cut-force acting at the component to which frame\_b is connected is determined and provided at the output signal connector **force** (= frame\_a.f). If parameter **positiveSign = false**, the negative cut-force is provided (= frame\_b.f). If **frame\_resolve** is connected to another frame, then the cut-force is resolved in frame\_resolve. If **frame\_resolve** is not connected then the coordinate system in which the cut-force is resolved is defined by parameter **resolveInFrame\_a**. If this parameter is **true**, then the cut-force is resolved in frame\_a, otherwise it is resolved in the world frame.

In the following figure the animation of a CutForce sensor is shown. The dark blue coordinate system is frame\_b, and the green arrow is the cut force acting at frame\_b and with negative sign at frame\_a.



## Parameters

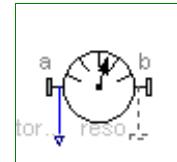
Name	Default	Description
animation	true	= true, if animation shall be enabled (show arrow)
positiveSign	true	= true, if force with positive sign is returned (= frame_a.f), otherwise with negative sign (= frame_b.f)
resolveInFrame_a	true	= true, if force is resolved in frame_a/frame_b, otherwise in the world frame (if connector frame_resolve is connected, the force is resolved in frame_resolve)
if animation = true		
N_to_m	1000	Force arrow scaling (length = force/N_to_m) [N/m]
forceDiameter	world.defaultArrowDiameter	Diameter of force arrow [m]
forceColor	Modelica.Mechanics.MultiBody.. .	Color of force arrow
specularCoefficient	world.defaultSpecularCoeffic...	Reflection of ambient light (= 0: light is completely absorbed)

## Connectors

Name	Description
frame_a	Coordinate system with one cut-force and cut-torque
frame_b	Coordinate system with one cut-force and cut-torque
frame_resolve	If connected, the output signals are resolved in this frame (cut-force/-torque are set to zero)
force[3]	Cut force resolved in frame_a/frame_b or in frame_resolved, if connected

## Modelica.Mechanics.MultiBody.Sensors.CutTorque

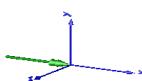
Measure cut torque vector



## Information

The cut-torque acting at the component to which frame\_b is connected is determined and provided at the output signal connector **torque** (= frame\_a.t). If parameter **positiveSign** = **false**, the negative cut-force is provided (= frame\_b.t). If **frame\_resolve** is connected to another frame, then the cut-torque is resolved in **frame\_resolve**. If **frame\_resolve** is not connected then the coordinate system in which the cut-torque is resolved is defined by parameter **resolveInFrame\_a**. If this parameter is **true**, then the cut-torque is resolved in frame\_a, otherwise it is resolved in the world frame.

In the following figure the animation of a CutTorque sensor is shown. The dark blue coordinate system is frame\_b, and the green arrow is the cut torque acting at frame\_b and with negative sign at frame\_a.



## Parameters

Name	Default	Description
animation	true	= true, if animation shall be enabled (show arrow)

## 566 Modelica.Mechanics.MultiBody.Sensors.CutTorque

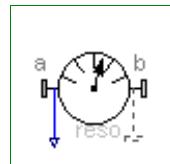
positiveSign	true	= true, if torque with positive sign is returned (= frame_a.t), otherwise with negative sign (= frame_b.t)
resolveInFrame_a	true	= true, if torque is resolved in frame_a/frame_b, otherwise in the world frame (if connector frame_resolve is connected, the torque is resolved in frame_resolve)
if animation = true		
Nm_to_m	1000	Torque arrow scaling (length = torque/Nm_to_m) [N.m/m]
torqueDiameter	world.defaultArrowDiameter	Diameter of torque arrow [m]
torqueColor	Modelica.Mechanics.MultiBody..	Color of torque arrow
specularCoefficient	world.defaultSpecularCoeffic...	Reflection of ambient light (= 0: light is completely absorbed)

### Connectors

Name	Description
frame_a	Coordinate system with one cut-force and cut-torque
frame_b	Coordinate system with one cut-force and cut-torque
frame_resolve	If connected, the output signals are resolved in this frame (cut-force/-torque are set to zero)
torque[3]	Cut torque resolved in frame_a/frame_b or in frame_resolved, if connected

## Modelica.Mechanics.MultiBody.Sensors.CutForceAndTorque

Measure cut force and cut torque vector



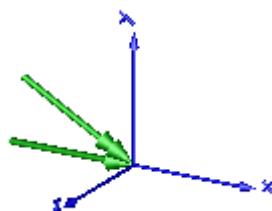
### Information

The cut-force and cut-torque acting at the component to which frame\_b is connected are determined and provided at the output signal connector **load**:

```
load[1:3] = frame_a.f;
load[4:6] = frame_a.t;
```

If parameter **positiveSign** = **false**, the negative cut-force and negative cut-torque is provided (= frame\_b.f and frame\_b.t). If **frame\_resolve** is connected to another frame, then the cut-force and cut-torque are resolved in frame\_resolve. If **frame\_resolve** is **not** connected then the coordinate system in which the cut-force and cut-torque is resolved is defined by parameter **resolveInFrame\_a**. If this parameter is **true**, then the cut-force and cut-torque is resolved in frame\_a, otherwise it is resolved in the world frame.

In the following figure the animation of a CutForceAndTorque sensor is shown. The dark blue coordinate system is frame\_b, and the green arrows are the cut force and the cut torque, respectively, acting at frame\_b and with negative sign at frame\_a.



## Parameters

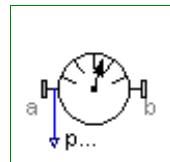
Name	Default	Description
animation	true	= true, if animation shall be enabled (show force and torque arrow)
positiveSign	true	= true, if force and torque with positive sign is returned (= frame_a.f/t), otherwise with negative sign (= frame_b.f/t)
resolveInFrame_a	true	= true, if force and torque are resolved in frame_a/frame_b, otherwise in the world frame (if connector frame_resolve is connected, the force/torque is resolved in frame_resolve)
if animation = true		
N_to_m	1000	Force arrow scaling (length = force/N_to_m) [N/m]
Nm_to_m	1000	Torque arrow scaling (length = torque/Nm_to_m) [N.m/m]
forceDiameter	world.defaultArrowDiameter	Diameter of force arrow [m]
torqueDiameter	forceDiameter	Diameter of torque arrow [m]
forceColor	Modelica.Mechanics.MultiBody.. .	Color of force arrow
torqueColor	Modelica.Mechanics.MultiBody.. .	Color of torque arrow
specularCoefficient	world.defaultSpecularCoeffic... .	Reflection of ambient light (= 0: light is completely absorbed)

## Connectors

Name	Description
frame_a	Coordinate system with one cut-force and cut-torque
frame_b	Coordinate system with one cut-force and cut-torque
frame_resolve	If connected, the output signals are resolved in this frame (cut-force/-torque are set to zero)
load[6]	Cut force and cut torque resolved in frame_a/frame_b or in frame_resolved, if connected

## Modelica.Mechanics.MultiBody.Sensors.Power

Measure power flowing from frame\_a to frame\_b



## Information

This component provides the power flowing from frame\_a to frame\_b as output signal **power**.

## Connectors

Name	Description
frame_a	Coordinate system fixed to the component with one cut-force and cut-torque
frame_b	Coordinate system fixed to the component with one cut-force and cut-torque
power	Power at frame_a as output signal

## Modelica.Mechanics.MultiBody.Types

Constants and types with choices, especially to build menus

## Information

In this package **types** and **constants** are defined that are used in the MultiBody library. The types have additional annotation choices definitions that define the menus to be built up in the graphical user interface when the type is used as parameter in a declaration.

## Package Content

Name	Description
R Axis	Axis vector with choices for menus
S AxisLabel	Label of axis with choices for menus
I RotationSequence	Sequence of planar frame rotations with choices for menus
I Color	RGB representation of color (will be improved with a color editor)
R SpecularCoefficient	Reflection of ambient light (= 0: light is completely absorbed)
S ShapeType	Type of shape (box, sphere, cylinder, pipecylinder, cone, pipe, beam, gearwheel, spring, dxf-file)
R ShapeExtra	Reflection of ambient light (= 0: light is completely absorbed)
R AngularVelocity_degs	Angular velocity type in deg/s
R AngularAcceleration_deg_s2	Angular acceleration type in deg/s <sup>2</sup>
E RotationTypes	Type, constants and menu choices for rotation types, as temporary solution until enumerations are available
E GravityTypes	Type, constants and menu choices for gravity fields, as temporary solution until enumerations are available
E Init	Type, constants and menu choices to define initialization, as temporary solution until enumerations are available
C Defaults	Default settings of the MultiBody library via constants

## Types and constants

```

type Axis = Modelica(Icons.TypeReal[3] "Axis vector with choices for menus";

type AxisLabel = Modelica(Icons.TypeString
"Label of axis with choices for menus";

type RotationSequence = Modelica(Icons.TypeInteger[3] (min={1,1,1}, max={3,3,3})
"Sequence of planar frame rotations with choices for menus";

type Color = Modelica(Icons.TypeInteger[3] (each min=0, each max=255)
"RGB representation of color (will be improved with a color editor)";

type SpecularCoefficient = Modelica(Icons.TypeReal
"Reflection of ambient light (= 0: light is completely absorbed)";

type ShapeType = Modelica(Icons.TypeString
"Type of shape (box, sphere, cylinder, pipecylinder, cone, pipe, beam,
gearwheel, spring, dxf-file)";

type ShapeExtra = Modelica(Icons.TypeReal
"Reflection of ambient light (= 0: light is completely absorbed)";

```

---

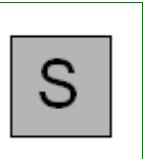
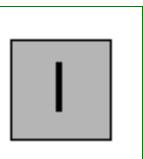
```

type AngularVelocity_degs = Modelica.Icons.TypeReal (final
quantity="AngularVelocity", final unit
= "deg/s") "Angular velocity type in deg/s";

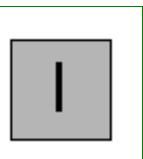
type AngularAcceleration_degs2 = Modelica.Icons.TypeReal (final
quantity="AngularAcceleration",
final unit="deg/s2") "Angular acceleration type in deg/s2";

```

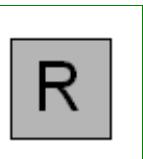
---

**Modelica.Mechanics.MultiBody.Types.Axis****Axis vector with choices for menus****Information****Modelica.Mechanics.MultiBody.Types.AxisLabel****Label of axis with choices for menus****Modelica.Mechanics.MultiBody.Types.RotationSequence****Sequence of planar frame rotations with choices for menus****Parameters**

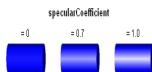
Name	Default	Description
min	{1,1,1}	
max	{3,3,3}	

**Modelica.Mechanics.MultiBody.Types.Color****RGB representation of color (will be improved with a color editor)****Information**

Type **Color** is an Integer vector with 3 elements, {r, g, b}, and specifies the color of a shape. {r,g,b} are the "red", "green" and "blue" color parts. Note, r g, b are given in the range 0 .. 255.

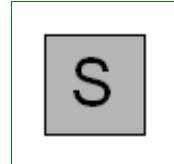
**Modelica.Mechanics.MultiBody.Types.SpecularCoefficient****Reflection of ambient light (= 0: light is completely absorbed)****Information**

Type **SpecularCoefficient** defines the reflection of ambient light on shape surfaces. If value = 0, the light is completely absorbed. Often, 0.7 is a reasonable value. It might be that from some viewing directions, a body is no longer visible, if the SpecularCoefficient value is too high. In the following image, the different values of SpecularCoefficient are shown for a cylinder:



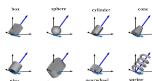
## Modelica.Mechanics.MultiBody.Types.ShapeType

Type of shape (box, sphere, cylinder, pipecylinder, cone, pipe, beam, gearwheel, spring, dxf-file)



### Information

Type **ShapeType** is used to define the shape of the visual object as parameter String. Usually, "shapeType" is used as instance name. The following values for shapeType are possible, e.g., shapeType="box":



The dark blue arrows in the figure above are directed along variable **lengthDirection**. The light blue arrows are directed along variable **widthDirection**. The **coordinate systems** in the figure represent frame\_a of the Shape component.

Additionally, external shapes are specified as DXF-files (only 3-dim.Face is supported). External shapes must be named "1", "2" etc.. The corresponding definitions should be in files "1.dxf", "2.dxf" etc. Since the DXF-files contain color and dimensions for the individual faces, the corresponding information in the model is currently ignored. The DXF-files must be found either in the current directory or in the directory where the Shape instance is stored that references the DXF file.

## Modelica.Mechanics.MultiBody.Types.ShapeExtra



Reflection of ambient light (= 0: light is completely absorbed)

### Information

This type is used in shapes of visual objects to define extra data depending on the shape type. Usually, input variable **extra** is used as instance name:

<b>shapeType</b>	<b>Meaning of variable extra</b>
"cylinder"	if extra > 0, a black line is included in the cylinder to show the rotation of it.
"cone"	extra = diameter-left-side / diameter-right-side, i.e., extra = 1: cylinder extra = 0: "real" cone.
"pipe"	extra = outer-diameter / inner-diameter, i.e, extra = 1: cylinder that is completely hollow extra = 0: cylinder without a hole.
"gearwheel"	extra is the number of teeth of the gear.
"spring"	extra is the number of windings of the spring. Additionally, "height" is <b>not</b> the "height" but 2*coil-width.

## Modelica.Mechanics.MultiBody.Types.AngularVelocity\_degs



Angular velocity type in deg/s

**Modelica.Mechanics.MultiBody.TypesAngularAcceleration\_degs2**Angular acceleration type in deg/s<sup>2</sup>**Modelica.Mechanics.MultiBody.Types.RotationTypes**

Type, constants and menu choices for rotation types, as temporary solution until enumerations are available

**Information****Package Content**

Name	Description
RotationAxis=1	
TwoAxesVectors=2	
PlanarRotationSequence=3	
Temp	Temporary type of RotationTypes with choices for menus (until enumerations are available)

**Types and constants**

```

constant Integer RotationAxis=1;

constant Integer TwoAxesVectors=2;

constant Integer PlanarRotationSequence=3;

type Temp
  "Temporary type of RotationTypes with choices for menus (until enumerations
  are available)"

  extends Modelica.Icons.TypeInteger;

end Temp;

```

**Modelica.Mechanics.MultiBody.Types.RotationTypes.Temp**

Temporary type of RotationTypes with choices for menus (until enumerations are available)

**Information**

Types.RotationTypes.	Meaning
RotationAxis	frame_b is defined by rotating the coordinate system along an axis fixed in frame_a and with a fixed angle.
TwoAxesVectors	frame_b is defined by resolving two vectors of frame_b in frame_a.
PlanarRotationSequence	frame_b is defined by rotating the coordinate system along 3 consecutive axes vectors with fixed rotation angles (e.g. Cardan or Euler angle sequence rotation).

## Modelica.Mechanics.MultiBody.Types.GravityTypes

Type, constants and menu choices for gravity fields, as temporary solution until enumerations are available

### Information

#### Package Content

Name	Description
NoGravity=0	
UniformGravity=1	
PointGravity=2	
Temp	Temporary type of gravity field with choices for menus (until enumerations are available)

### Types and constants

```
constant Integer NoGravity=0;

constant Integer UniformGravity=1;

constant Integer PointGravity=2;

type Temp
  "Temporary type of gravity field with choices for menus (until enumerations
  are available)"

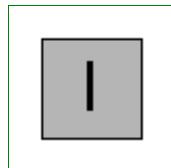
  extends Modelica.Icons.TypeInteger;

end Temp;
```

---

## Modelica.Mechanics.MultiBody.Types.GravityTypes.Temp

Temporary type of gravity field with choices for menus (until enumerations are available)



### Information

Types.GravityTypes.	Meaning
NoGravity	No gravity field
UniformGravity	Gravity field is described by a vector of constant gravity acceleration
PointGravity	Central gravity field. The gravity acceleration vector is directed to the field center and the gravity is proportional to $1/r^2$ , where $r$ is the distance to the field center.

---

## Modelica.Mechanics.MultiBody.Types.Init

Type, constants and menu choices to define initialization, as temporary solution until enumerations are available

## Information

### Package Content

Name	Description
Free=1	
PositionVelocity=2	
SteadyState=3	
Position=4	
Velocity=5	
VelocityAcceleration=6	
PositionVelocityAcceleration=7	
 Temp	Temporary type of Init with choices for menus (until enumerations are available)

### Types and constants

```

constant Integer Free=1;

constant Integer PositionVelocity=2;

constant Integer SteadyState=3;

constant Integer Position=4;

constant Integer Velocity=5;

constant Integer VelocityAcceleration=6;

constant Integer PositionVelocityAcceleration=7;

type Temp
  "Temporary type of Init with choices for menus (until enumerations are
available)"

  extends Modelica.Icons.TypeInteger;

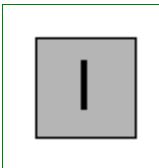
end Temp;

```

---

### Modelica.Mechanics.MultiBody.Types.Init.Temp

Temporary type of Init with choices for menus (until enumerations are available)



### Information

Types.Init.	Meaning
Free	No initialization
PositionVelocity	Initialize generalized position and velocity variables
SteadyState	Initialize in steady state (velocity and acceleration are zero)
Position	Initialize only generalized position variable(s)

## 574 Modelica.Mechanics.MultiBody.Types.Init.Temp

Velocity	Initialize only generalized velocity variable(s)
VelocityAcceleration	Initialize generalized velocity and acceleration variables
PositionVelocityAcceleration	Initialize generalized position, velocity and acceleration variables

## Modelica.Mechanics.MultiBody.Types.Defaults

### Default settings of the MultiBody library via constants

#### Information

This package contains constants used as default setting in the MultiBody library.

#### Package Content

Name	Description
BodyColor={0,128,255}	Default color for body shapes that have mass (light blue)
RodColor={155,155,155}	Default color for massless rod shapes (grey)
JointColor={255,0,0}	Default color for elementary joints (red)
ForceColor={0,128,0}	Default color for force arrow (dark green)
TorqueColor={0,128,0}	Default color for torque arrow (dark green)
SpringColor={0,0,255}	Default color for a spring (blue)
SensorColor={255,255,0}	Default color for sensors (yellow)
FrameColor={0,0,0}	Default color for frame axes and labels (black)
ArrowColor={0,0,255}	Default color for arrows and double arrows (blue)
FrameHeadLengthFraction=5.0	Frame arrow head length / arrow diameter
FrameHeadWidthFraction=3.0	Frame arrow head width / arrow diameter
FrameLabelHeightFraction=3.0	Height of frame label / arrow diameter
ArrowHeadLengthFraction=4.0	Arrow head length / arrow diameter
ArrowHeadWidthFraction=3.0	Arrow head width / arrow diameter
BodyCylinderDiameterFraction=3	Default for body cylinder diameter as a fraction of body sphere diameter
JointRodDiameterFraction=2	Default for rod diameter as a fraction of joint sphere diameter attached to rod

#### Types and constants

```
constant Types.Color BodyColor={0,128,255}
"Default color for body shapes that have mass (light blue)";

constant Types.Color RodColor={155,155,155}
"Default color for massless rod shapes (grey)";

constant Types.Color JointColor={255,0,0}
"Default color for elementary joints (red)";

constant Types.Color ForceColor={0,128,0}
"Default color for force arrow (dark green)";

constant Types.Color TorqueColor={0,128,0}
"Default color for torque arrow (dark green);
```

```

constant Types.Color SpringColor={0,0,255}
"Default color for a spring (blue)";

constant Types.Color SensorColor={255,255,0}
"Default color for sensors (yellow)";

constant Types.Color FrameColor={0,0,0}
"Default color for frame axes and labels (black)";

constant Types.Color ArrowColor={0,0,255}
"Default color for arrows and double arrows (blue)";

constant Real FrameHeadLengthFraction=5.0
"Frame arrow head length / arrow diameter";

constant Real FrameHeadWidthFraction=3.0
"Frame arrow head width / arrow diameter";

constant Real FrameLabelHeightFraction=3.0
"Height of frame label / arrow diameter";

constant Real ArrowHeadLengthFraction=4.0
"Arrow head length / arrow diameter";

constant Real ArrowHeadWidthFraction=3.0 "Arrow head width / arrow diameter";

constant SI.Diameter BodyCylinderDiameterFraction=3
"Default for body cylinder diameter as a fraction of body sphere diameter";

constant Real JointRodDiameterFraction=2
"Default for rod diameter as a fraction of joint sphere diameter attached to
rod";

```

## Modelica.Mechanics.MultiBody.Visualizers

### 3-dimensional visual objects used for animation

#### Information

Package **Visualizers** contains components to visualize 3-dimensional shapes. These components are the basis for the animation features of the MultiBody library.

#### Content

<b>FixedShape</b>	Animation shape of a part with fixed sizes. FixedShape2 has additionally a frame_b for easier connection to further visual objects. The following shape types are supported:
-------------------	--

	<p>box    sphere    cylinder    cone    pipe beam    gearwheel    spring</p>
FixedFrame	Visualizing a coordinate system including axes labels with fixed sizes: 
FixedArrow, SignalArrow	Visualizing an arrow. Model "FixedArrow" provides a fixed sized arrow, model "SignalArrow" provides an arrow with dynamically varying length that is defined by an input signal vector: 
Advanced	<b>Package</b> that contains components to visualize 3-dimensional shapes where all parts of the shape can vary dynamically. Basic knowledge of Modelica is needed in order to utilize the components of this package.

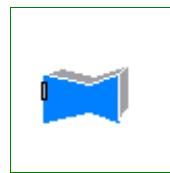
The colors of the visualization components are declared with the predefined type **MultiBody.Types.Color**. This is a vector with 3 elements, {r, g, b}, and specifies the color of the shape. {r,g,b} are the "red", "green" and "blue" color parts. Note, r g, b are given as Integer[3] in the ranges 0 .. 255, respectively.

### Package Content

Name	Description
FixedShape	Animation shape of a part with fixed shape type and dynamically varying shape definition
FixedShape2	Animation shape of a part with fixed shape type and dynamically varying shape definition with two frames
FixedFrame	Visualizing a coordinate system including axes labels (visualization data may vary dynamically)
FixedArrow	Visualizing an arrow with dynamically varying size in frame_a
SignalArrow	Visualizing an arrow with dynamically varying size in frame_a based on input signal
Advanced	Visualizers that require basic knowledge about Modelica in order to use them

### Modelica.Mechanics.MultiBody.Visualizers.FixedShape

Animation shape of a part with fixed shape type and dynamically varying shape definition



### Information

Model **FixedShape** defines a visual shape that is shown at the location of its frame\_a. All describing data such as size and color can vary dynamically by providing appropriate expressions in the input fields of the

parameter menu. The only exception is parameter **shapeType** that cannot be changed during simulation. The following shapes are currently supported via parameter **shapeType** (e.g., **shapeType="box"**):



The dark blue arrows in the figure above are directed along variable **lengthDirection**. The light blue arrows are directed along variable **widthDirection**. The **coordinate systems** in the figure represent frame\_a of the FixedShape component.

Additionally external shapes are specified as DXF-files (only 3-dim.Face is supported). External shapes must be named "1", "2" etc.. The corresponding definitions should be in files "1.dxf", "2.dxf" etc. Since the DXF-files contain color and dimensions for the individual faces, the corresponding information in the model is currently ignored. The DXF-files must be found in the current directory.

The sizes of any of the above components are specified by the **length**, **width** and **height** variables. Via variable **extra** additional data can be defined:

<b>shapeType</b>	<b>Meaning of parameter extra</b>
"cylinder"	if extra > 0, a black line is included in the cylinder to show the rotation of it.
"cone"	extra = diameter-left-side / diameter-right-side, i.e., extra = 1: cylinder extra = 0: "real" cone.
"pipe"	extra = outer-diameter / inner-diameter, i.e., extra = 1: cylinder that is completely hollow extra = 0: cylinder without a hole.
"gearwheel"	extra is the number of teeth of the gear.
"spring"	extra is the number of windings of the spring. Additionally, "height" is <b>not</b> the "height" but 2*coil-width.

Parameter **color** is a vector with 3 elements, {r, g, b}, and specifies the color of the shape. {r,g,b} are the "red", "green" and "blue" color parts. Note, r g, b are given as Integer[3] in the ranges 0 .. 255, respectively. The predefined type **MultiBody.Types.Color** contains a temporary menu definition of the colors used in the MultiBody library (this will be replaced by a color editor).

## Parameters

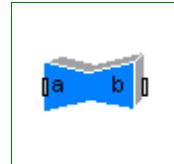
Name	Default	Description
animation	true	= true, if animation shall be enabled
if animation = true		
shapeType	"box"	Type of shape
r_shape[3]	{0,0,0}	Vector from frame_a to shape origin, resolved in frame_a [m]
lengthDirection	{1,0,0}	Vector in length direction of shape, resolved in frame_a
widthDirection	{0,1,0}	Vector in width direction of shape, resolved in frame_a
length	1	Length of shape [m]
width	0.1	Width of shape [m]
height	0.1	Height of shape [m]
color	{0,128,255}	Color of shape
extra	0.0	Additional data for cylinder, cone, pipe, gearwheel and spring
specularCoefficient	world.defaultSpecularCoeffic...	Reflection of ambient light (= 0: light is completely absorbed)

## Connectors

Name	Description
frame_a	Coordinate system in which visualization data is resolved

## Modelica.Mechanics.MultiBody.Visualizers.FixedShape2

Animation shape of a part with fixed shape type and dynamically varying shape definition with two frames



## Information

Model **FixedShape2** defines a visual shape that is shown at the location of its frame\_a. This model is identical to **FixedShape** with the only difference that an additional frame\_b is present which is parallel to frame\_a. This makes it more convenient to connect several visual shapes together when building up more complex visual objects. All describing data such as size and color can vary dynamically by providing appropriate expressions in the input fields of the parameter menu. The only exception is parameter **shapeType** that cannot be changed during simulation. The following shapes are currently supported via parameter **shapeType** (e.g., **shapeType="box"**):



The dark blue arrows in the figure above are directed along variable **lengthDirection**. The light blue arrows are directed along variable **widthDirection**. The **coordinate systems** in the figure represent frame\_a of the FixedShape component.

Additionally external shapes are specified as DXF-files (only 3-dim.Face is supported). External shapes must be named "1", "2" etc.. The corresponding definitions should be in files "1.dxf", "2.dxf" etc. Since the DXF-files contain color and dimensions for the individual faces, the corresponding information in the model is currently ignored. The DXF-files must be found in the current directory.

The sizes of any of the above components are specified by the **length**, **width** and **height** variables. Via variable **extra** additional data can be defined:

shapeType	Meaning of parameter extra
"cylinder"	if extra > 0, a black line is included in the cylinder to show the rotation of it.
"cone"	extra = diameter-left-side / diameter-right-side, i.e., extra = 1: cylinder extra = 0: "real" cone.
"pipe"	extra = outer-diameter / inner-diameter, i.e., extra = 1: cylinder that is completely hollow extra = 0: cylinder without a hole.
"gearwheel"	extra is the number of teeth of the gear.
"spring"	extra is the number of windings of the spring. Additionally, "height" is <b>not</b> the "height" but 2*coil-width.

Parameter **color** is a vector with 3 elements, {r, g, b}, and specifies the color of the shape. {r,g,b} are the "red", "green" and "blue" color parts. Note, r g, b are given as Integer[3] in the ranges 0 .. 255, respectively. The predefined type **MultiBody.Types.Color** contains a temporary menu definition of the colors used in the MultiBody library (this will be replaced by a color editor).

In the following figure the relationships between frame\_a and frame\_b are shown. The origin of frame\_b with respect to frame\_a is specified via parameter vector **r**.



## Parameters

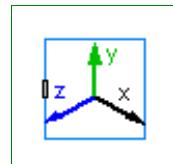
Name	Default	Description
animation	true	= true, if animation shall be enabled
if animation = true		
shapeType	"box"	Type of shape
r_shape[3]	{0,0,0}	Vector from frame_a to shape origin, resolved in frame_a [m]
lengthDirection	r - r_shape	Vector in length direction of shape, resolved in frame_a
widthDirection	{0,1,0}	Vector in width direction of shape, resolved in frame_a
length	Frames.length(r - r_shape)	Length of shape [m]
width	0.1	Width of shape [m]
height	width	Height of shape [m]
extra	0.0	Additional data for cylinder, cone, pipe, gearwheel and spring
color	{0,128,255}	Color of shape
specularCoefficient	world.defaultSpecularCoeffic...	Reflection of ambient light (= 0: light is completely absorbed)

## Connectors

Name	Description
frame_a	Coordinate system a (all shape definition vectors are resolved in this frame)
frame_b	Coordinate system b

## Modelica.Mechanics.MultiBody.Visualizers.FixedFrame

Visualizing a coordinate system including axes labels (visualization data may vary dynamically)



## Information

Model **FixedFrame** visualizes the three axes of its coordinate system **frame\_a** together with appropriate axes labels. A typical example is shown in the following figure:



The sizes of the axes, the axes colors and the specular coefficient (= reflection factor for ambient light) can vary dynamically by providing appropriate expressions in the input fields of the parameter menu.

## Parameters

Name	Default	Description
animation	true	= true, if animation shall be enabled
if animation = true		
showLabels	true	= true, if labels shall be shown

## 580 Modelica.Mechanics.MultiBody.Visualizers.FixedFrame

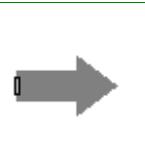
length	0.5	Length of axes arrows [m]
diameter	length/world.defaultFrameDia...	Diameter of axes arrows [m]
color_x	Modelica.Mechanics.MultiBody...	Color of x-arrow
color_y	color_x	Color of y-arrow
color_z	color_x	Color of z-arrow
specularCoefficient	world.defaultSpecularCoeffic...	Reflection of ambient light (= 0: light is completely absorbed)

### Connectors

Name	Description
frame_a	Coordinate system in which visualization data is resolved

## Modelica.Mechanics.MultiBody.Visualizers.FixedArrow

Visualizing an arrow with dynamically varying size in frame\_a



### Information

Model **FixedArrow** defines an arrow that is shown at the location of its frame\_a.



The direction of the arrow specified with vector **n** is with respect to frame\_a, i.e., the local frame to which the arrow component is attached. The direction and length of the arrow, its diameter and color can vary dynamically by providing appropriate expressions in the input fields of the parameter menu.

### Parameters

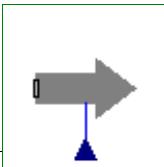
Name	Default	Description
animation	true	= true, if animation shall be enabled
if animation = true		
r_tail[3]	{0,0,0}	Vector from frame_a to arrow tail, resolved in frame_a [m]
n	{1,0,0}	Vector in arrow direction, resolved in frame_a
length	0.1	Length of complete arrow [m]
diameter	world.defaultArrowDiameter	Diameter of arrow line [m]
color	{0,0,255}	Color of arrow
specularCoefficient	world.defaultSpecularCoeffic...	Reflection of ambient light (= 0: light is completely absorbed)

### Connectors

Name	Description
frame_a	Coordinate system in which visualization data is resolved

## Modelica.Mechanics.MultiBody.Visualizers.SignalArrow

Visualizing an arrow with dynamically varying size in frame\_a based on input signal



## Information

Model **SignalArrow** defines an arrow that is dynamically visualized at the location where its frame\_a is attached. The position vector from the tail to the head of the arrow, resolved in frame\_a, is defined via the signal vector of the connector **r\_head** (Real  $r_{head}[3]$ ):



The tail of the arrow is defined with parameter **r\_tail** with respect to frame\_a (vector from the origin of frame\_a to the arrow tail).

## Parameters

Name	Default	Description
animation	true	= true, if animation shall be enabled
if animation = true		
r_tail[3]	{0,0,0}	Vector from frame_a to arrow tail, resolved in frame_a [m]
diameter	world.defaultArrowDiameter	Diameter of arrow line [m]
color	{0,0,255}	Color of arrow
specularCoefficient	world.defaultSpecularCoeffic...	Reflection of ambient light (= 0: light is completely absorbed)

## Connectors

Name	Description
frame_a	Coordinate system in which visualization data is resolved
r_head[3]	Position vector from origin of frame_a to head of arrow, resolved in frame_a

## Modelica.Mechanics.MultiBody.Visualizers.Advanced

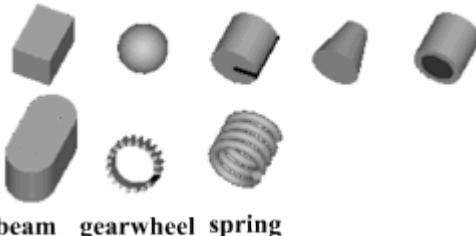
Visualizers that require basic knowledge about Modelica in order to use them

## Information

Package **Visualizers.Advanced** contains components to visualize 3-dimensional shapes with dynamical sizes. None of the components has a frame connector. The position and orientation is set via modifiers. Basic knowledge of Modelica is needed in order to utilize the components of this package. These components have also to be used for models, where the forces and torques in the frame connector are set via equations (in this case, the models of the Visualizers package cannot be used, since they all have frame connectors).

## Content

<b>Arrow</b>	Visualizing an arrow where all parts of the arrow can vary dynamically: 
<b>DoubleArrow</b>	Visualizing a double arrow where all parts of the arrow can vary dynamically:

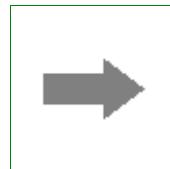
	
<b>Shape</b>	<p>Animation shape of a part with dynamically varying sizes. The following shape types are supported:</p> <p>box    sphere    cylinder    cone    pipe</p>  <p>beam    gearwheel    spring</p>

## Package Content

Name	Description
 Arrow	Visualizing an arrow with variable size; all data have to be set as modifiers (see info layer)
 DoubleArrow	Visualizing a double arrow with variable size; all data have to be set as modifiers (see info layer)
 Shape	Different visual shapes with variable size; all data have to be set as modifiers (see info layer)

## Modelica.Mechanics.MultiBody.Visualizers.Advanced.Arrow

Visualizing an arrow with variable size; all data have to be set as modifiers (see info layer)



### Information

Model **Arrow** defines an arrow that is dynamically visualized at the defined location (see variables below).



The variables under heading **Parameters** below are declared as (time varying) **input** variables. If the default equation is not appropriate, a corresponding modifier equation has to be provided in the model where an **Arrow** instance is used, e.g., in the form

```
Visualizers.Advanced.Arrow arrow(diameter = sin(time));
```

Variable **color** is an Integer vector with 3 elements, {r, g, b}, and specifies the color of the shape. {r,g,b} are the "red", "green" and "blue" color parts. Note, r g, b are given in the range 0 .. 255. The predefined type **MultiBody.Types.Color** contains a menu definition of the colors used in the MultiBody library (will be replaced by a color editor).

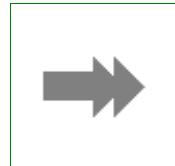
### Parameters

Name	Default	Description
R	Frames.nullRotation()	Orientation object to rotate the world frame into the arrow frame.

r[3]	{0,0,0}	Position vector from origin of world frame to origin of arrow frame, resolved in world frame [m]
r_tail[3]	{0,0,0}	Position vector from origin of arrow frame to arrow tail, resolved in arrow frame [m]
r_head[3]	{0,0,0}	Position vector from arrow tail to the head of the arrow, resolved in arrow frame [m]
diameter	world.defaultArrowDiameter	Diameter of arrow line [m]
color	Modelica.Mechanics.MultiBody.. .	Color of arrow
specularCoefficient	world.defaultSpecularCoeffic...	Material property describing the reflecting of ambient light (= 0 means, that light is completely absorbed)

## Modelica.Mechanics.MultiBody.Visualizers.Advanced.DoubleArrow

Visualizing a double arrow with variable size; all data have to be set as modifiers (see info layer)



### Information

Model **DoubleArrow** defines a double arrow that is dynamically visualized at the defined location (see variables below).



The variables under heading **Parameters** below are declared as (time varying) **input** variables. If the default equation is not appropriate, a corresponding modifier equation has to be provided in the model where an **Arrow** instance is used, e.g., in the form

```
Visualizers.Advanced.DoubleArrow doubleArrow(diameter = sin(time));
```

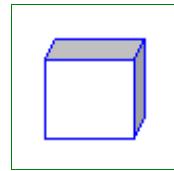
Variable **color** is an Integer vector with 3 elements, {r, g, b}, and specifies the color of the shape. {r,g,b} are the "red", "green" and "blue" color parts. Note, r g, b are given in the range 0 .. 255. The predefined type **MultiBody.Types.Color** contains a menu definition of the colors used in the MultiBody library (will be replaced by a color editor).

### Parameters

Name	Default	Description
R	Frames.nullRotation()	Orientation object to rotate the world frame into the arrow frame.
r[3]	{0,0,0}	Position vector from origin of world frame to origin of arrow frame, resolved in world frame [m]
r_tail[3]	{0,0,0}	Position vector from origin of arrow frame to double arrow tail, resolved in arrow frame [m]
r_head[3]	{0,0,0}	Position vector from double arrow tail to the head of the double arrow, resolved in arrow frame [m]
diameter	world.defaultArrowDiameter	Diameter of arrow line [m]
color	Modelica.Mechanics.MultiBody.. .	Color of double arrow
specularCoefficient	world.defaultSpecularCoeffic...	Material property describing the reflecting of ambient light (= 0 means, that light is completely absorbed)

## Modelica.Mechanics.MultiBody.Visualizers.Advanced.Shape

Different visual shapes with variable size; all data have to be set as modifiers (see info layer)



### Information

Model **Shape** defines a visual shape that is shown at the location of its reference coordinate system, called 'object frame' below. All describing variables such as size and color can vary dynamically (with the only exception of parameter **shapeType**). The default equations in the declarations should be modified by providing appropriate equations. Model **Shape** is usually used as a basic building block to implement simpler to use graphical components.

The following shapes are supported via parameter **shapeType** (e.g., **shapeType="box"**):



The dark blue arrows in the figure above are directed along variable **lengthDirection**. The light blue arrows are directed along variable **widthDirection**. The **coordinate systems** in the figure represent frame\_a of the Shape component.

Additionally, external shapes are specified as DXF-files (only 3-dim.Face is supported). External shapes must be named "1", "2" etc.. The corresponding definitions should be in files "1.dxf", "2.dxf" etc. Since the DXF-files contain color and dimensions for the individual faces, the corresponding information in the model is currently ignored. The DXF-files must be found either in the current directory or in the directory where the Shape instance is stored that references the DXF file.

Via input variable **extra** additional sizing data is defined according to:

<b>shapeType</b>	<b>Meaning of variable extra</b>
"cylinder"	if extra > 0, a black line is included in the cylinder to show the rotation of it.
"cone"	extra = diameter-left-side / diameter-right-side, i.e., extra = 1: cylinder extra = 0: "real" cone.
"pipe"	extra = outer-diameter / inner-diameter, i.e., extra = 1: cylinder that is completely hollow extra = 0: cylinder without a hole.
"gearwheel"	extra is the number of teeth of the gear.
"spring"	extra is the number of windings of the spring. Additionally, "height" is <b>not</b> the "height" but 2*coil-width.

Parameter **color** is an Integer vector with 3 elements, {r, g, b}, and specifies the color of the shape. {r,g,b} are the "red", "green" and "blue" color parts. Note, r g, b are given in the range 0 .. 255. The predefined type **MultiBody.Types.Color** contains a menu definition of the colors used in the MultiBody library (will be replaced by a color editor).

The variables under heading **Parameters** below are declared as (time varying) **input** variables. If the default equation is not appropriate, a corresponding modifier equation has to be provided in the model where a **Shape** instance is used, e.g., in the form

```
Visualizers.Advanced.Shape shape(length = sin(time));
```

### Parameters

Name	Default	Description
<b>shapeType</b>	"box"	Type of shape (box, sphere, cylinder, pipecylinder, cone, pipe,

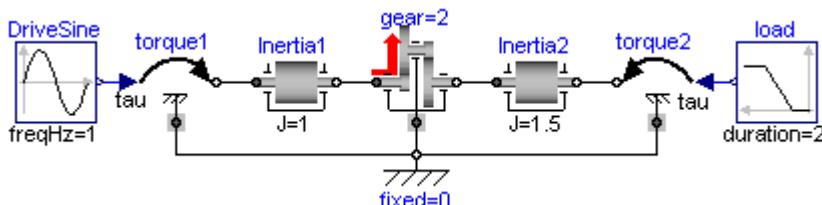
		beam, gearwheel, spring)
R	Frames.nullRotation()	Orientation object to rotate the world frame into the object frame
r[3]	{0,0,0}	Position vector from origin of world frame to origin of object frame, resolved in world frame [m]
r_shape[3]	{0,0,0}	Position vector from origin of object frame to shape origin, resolved in object frame [m]
lengthDirection[3]	{1,0,0}	Vector in length direction, resolved in object frame
widthDirection[3]	{0,1,0}	Vector in width direction, resolved in object frame
length	0	Length of visual object [m]
width	0	Width of visual object [m]
height	0	Height of visual object [m]
extra	0.0	Additional size data for some of the shape types
color[3]	{255,0,0}	Color of shape
specularCoefficient	0.7	Reflection of ambient light (= 0: light is completely absorbed)

## Modelica.Mechanics.Rotational

Library to model 1-dimensional, rotational mechanical systems

### Information

Library **Rotational** is a **free** Modelica package providing 1-dimensional, rotational mechanical components to model in a convenient way drive trains with frictional losses. A typical, simple example is shown in the next figure:



For an introduction, have especially a look at:

- [Rotational.UsersGuide](#) discusses the most important aspects how to use this library.
- [Rotational.Examples](#) contains examples that demonstrate the usage of this library.

Copyright © 1998-2007, Modelica Association and DLR.

*This Modelica package is **free** software; it can be redistributed and/or modified under the terms of the **Modelica license**, see the license conditions and the accompanying [disclaimer here](#).*

### Package Content

Name	Description
<a href="#">UsersGuide</a>	User's Guide of Rotational Library
<a href="#">Examples</a>	Demonstration examples of the components of this package
<a href="#">Sensors</a>	Sensors to measure variables in 1D rotational mechanical components
<a href="#">Interfaces</a>	Connectors and partial models for 1D rotational mechanical

	components
 Inertia	1D-rotational component with inertia
 IdealGear	Ideal gear without inertia
 IdealPlanetary	Ideal planetary gear box
 IdealGearR2T	Gearbox transforming rotational into translational motion
 Spring	Linear 1D rotational spring
 Damper	Linear 1D rotational damper
 SpringDamper	Linear 1D rotational spring and damper in parallel
 ElastoBacklash	Backlash connected in series to linear spring and damper (backlash is modeled with elasticity)
 BearingFriction	Coulomb friction in bearings
 Clutch	Clutch based on Coulomb friction
 OneWayClutch	Series connection of freewheel and clutch
 Brake	Brake based on Coulomb friction
 LossyGear	Gear with mesh efficiency and bearing friction (stuck/rolling possible)
 GearEfficiency	Obsolete component (use model LossyGear instead)
 Gear	Realistic model of a gearbox
 Gear2	Realistic model of a gearbox (based on LossyGear)
 Position	Forced movement of a flange according to a reference angle signal
 Speed	Forced movement of a flange according to a reference angular velocity signal
 Accelerate	Forced movement of a flange according to an acceleration signal
 Move	Forced movement of a flange according to an angle, speed and angular acceleration signal
 Fixed	Flange fixed in housing at a given angle
 Torque	Input signal acting as external torque on a flange
 Torque2	Input signal acting as torque on two flanges
 LinearSpeedDependentTorque	Linear dependency of torque versus speed
 QuadraticSpeedDependentTorque	Quadratic dependency of torque versus speed
 ConstantTorque	Constant torque, not dependent on speed
 ConstantSpeed	Constant speed, not dependent on torque
 TorqueStep	Constant torque, not dependent on speed
 RelativeStates	Definition of relative state variables
 InitializeFlange	Initializes a flange with pre-defined angle, speed and angular acceleration (usually, this is reference data from a control bus)
 Types	Constants and types with choices, especially to build menus

## Modelica.Mechanics.Rotational.UsersGuide



### User's Guide of Rotational Library

Library **Rotational** is a **free** Modelica package providing 1-dimensional, rotational mechanical components to model in a convenient way drive trains with frictional losses.

### Package Content

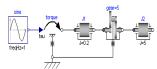
Name	Description
<a href="#">i Overview</a>	Overview
<a href="#">i FlangeConnectors</a>	Flange Connectors
<a href="#">i SupportTorques</a>	Support Torques
<a href="#">i SignConventions</a>	Sign Conventions
<a href="#">i UserDefinedComponents</a>	User Defined Components
<a href="#">i RequirementsForSimulationTool</a>	Requirements for Simulation Tools
<a href="#">i ReleaseNotes</a>	Release notes
<a href="#">i Contact</a>	Contact

## Modelica.Mechanics.Rotational.UsersGuide.Overview



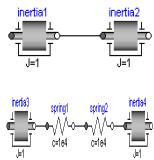
This package contains components to model **1-dimensional rotational mechanical** systems, including different types of gearboxes, shafts with inertia, external torques, spring/damper elements, frictional elements, backlash, elements to measure angle, angular velocity, angular acceleration and the cut-torque of a flange. In sublibrary **Examples** several examples are present to demonstrate the usage of the elements. Just open the corresponding example model and simulate the model according to the provided description.

A unique feature of this library is the **component-oriented** modeling of **Coulomb friction** elements, such as friction in bearings, clutches, brakes, and gear efficiency. Even (dynamically) coupled friction elements, e.g., as in automatic gearboxes, can be handled **without** introducing stiffness which leads to fast simulations. The underlying theory is new and is based on the solution of mixed continuous/discrete systems of equations, i.e., equations where the **unknowns** are of type **Real**, **Integer** or **Boolean**. Provided appropriate numerical algorithms for the solution of such types of systems are available in the simulation tool, the simulation of (dynamically) coupled friction elements of this library is **efficient** and **reliable**.



A simple example of the usage of this library is given in the figure above. This drive consists of a shaft with inertia  $J_1=0.2$  which is connected via an ideal gearbox with gear ratio=5 to a second shaft with inertia  $J_2=5$ . The left shaft is driven via an external, sinusoidal torque. The **filled** and **non-filled grey squares** at the left and right side of a component represent **mechanical flanges**. Drawing a line between such squares means that the corresponding flanges are **rigidly attached** to each other. By convention in this library, the connector characterized as a **filled** grey square is called **flange\_a** and placed at the left side of the component in the "design view" and the connector characterized as a **non-filled** grey square is called **flange\_b** and placed at the right side of the component in the "design view". The two connectors are completely **identical**, with the only exception that the graphical layout is a little bit different in order to distinguish them for easier access of the connector variables. For example,  $J_1.flange_a.tau$  is the cut-torque in the connector **flange\_a** of component  $J_1$ .

The components of this library can be **connected** together in an **arbitrary** way. E.g., it is possible to connect two springs or two shafts with inertia directly together, see figure below.



## Modelica.Mechanics.Rotational.UsersGuide.FlangeConnectors

A flange is described by the connector class Interfaces.**Flange\_a** or Interfaces.**Flange\_b**. As already noted, the two connector classes are completely identical. There is only a difference in the icons, in order to easier identify a flange variable in a diagram. Both connector classes contain the following variables:

```
SIunits.Angle phi "absolute rotation angle of flange";
flow SIunits.Torque tau "cut-torque in the flange";
```

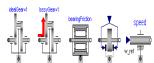
If needed, the angular velocity  $w$  and the angular acceleration  $a$  of a flange connector can be determined by differentiation of the flange angle  $\phi$ :

```
w = der(phi); a = der(w);
```

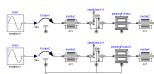


## Modelica.Mechanics.Rotational.UsersGuide.SupportTorques

The following figure shows examples of components equipped with a bearing flange (framed flange in the lower center), which can be used to fix components on the ground or on other rotating elements or to combine them with force elements. If the bearing flange is not connected, the components are assumed to be mounted on the ground. Otherwise, the bearing connector offers the possibility to consider, e.g., gearboxes mounted on the ground via spring-damper-systems (cf. example `ElasticBearing`). Independently, these components provide a variable `tau_support` stating the support torque exerted on the bearing.

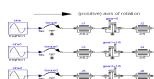


In general, it is not necessary to connect the bearing flange with a fixation, i.e., the two implementations in the following figure give identical results.



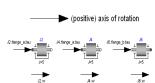
## Modelica.Mechanics.Rotational.UsersGuide.SignConventions

The variables of a component of this library can be accessed in the usual way. However, since most of these variables are basically elements of **vectors**, i.e., have a direction, the question arises how the signs of variables shall be interpreted. The basic idea is explained at hand of the following figure:

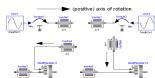


In the figure, three identical drive trains are shown. The only difference is that the gear of the middle drive train and the gear as well as the right inertia of the lower drive train are horizontally flipped with regards to the upper drive train. The signs of variables are now interpreted in the following way: Due to the 1-

dimensional nature of the model, all components are basically connected together along one line (more complicated cases are discussed below). First, one has to define a **positive** direction of this line, called **axis of rotation**. In the top part of the figure this is characterized by an arrow defined as **axis of rotation**. The simple rule is now: If a variable of a component is positive and can be interpreted as the element of a vector (e.g. torque or angular velocity vector), the corresponding vector is directed into the positive direction of the axis of rotation. In the following figure, the right-most inertias of the figure above are displayed with the positive vector direction displayed according to this rule:



The cut-torques  $J2.flange\_a.\tau$ ,  $J4.flange\_a.\tau$ ,  $J6.flange\_b.\tau$  of the right inertias are all identical and are directed into the direction of rotation if the values are positive. Similarly, the angular velocities  $J2.w$ ,  $J4.w$ ,  $J6.w$  of the right inertias are all identical and are also directed into the direction of rotation if the values are positive. Some special cases are shown in the next figure:



In the upper part of the figure, two variants of the connection of an external torque and an inertia are shown. In both cases, a positive signal input into the torque component accelerates the inertias `inertial1`, `inertial2` into the positive axis of rotation, i.e., the angular accelerations `inertial1.a`, `inertial2.a` are positive and are directed along the "axis of rotation" arrow. In the lower part of the figure the connection of inertias with a planetary gear is shown. Note, that the three flanges of the planetary gearbox are located along the axis of rotation and that the axis direction determines the positive rotation along these flanges. As a result, the positive rotation for `inertia4`, `inertia6` is as indicated with the additional grey arrows.

## Modelica.Mechanics.Rotational.UsersGuide.UserDefinedComponents

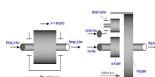
In this section some hints are given to define your own 1-dimensional rotational components which are compatible with the elements of this package. It is convenient to define a new component by inheritance from one of the following base classes, which are defined in sublibrary Interfaces:



Name	Description
<b>Rigid</b>	Rigid connection of two rotational 1D flanges (used for elements with inertia).
<b>Compliant</b>	Compliant connection of two rotational 1D flanges (used for force laws such as a spring or a damper).
<b>TwoFlanges</b>	General connection of two rotational 1D flanges (used for gearboxes).
<b>AbsoluteSensor</b>	Measure absolute flange variables.
<b>RelativeSensor</b>	Measure relative flange variables.

The difference between these base classes are the auxiliary variables defined in the model and the relations between the flange variables already defined in the base class. For example, in model **Rigid** the flanges `flange_a` and `flange_b` are rigidly connected, i.e.,  $flange\_a.\phi = flange\_b.\phi$ , whereas in model **Compliant** the cut-torques are the same, i.e.,  $flange\_a.\tau + flange\_b.\tau = 0$ .

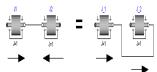
The equations of a mechanical component are vector equations, i.e., they need to be expressed in a common coordinate system. Therefore, for a component a **local axis of rotation** has to be defined. All vector quantities, such as cut-torques or angular velocities have to be expressed according to this definition. Examples for such a definition are given in the following figure for an inertia component and a planetary gearbox:



As can be seen, all vectors are directed into the direction of the rotation axis. The angles in the flanges are defined correspondingly. For example, the angle `sun.phi` in the flange of the sun wheel of the planetary

gearbox is positive, if rotated in mathematical positive direction (= counter clock wise) along the axis of rotation.

On first view, one may assume that the selected local coordinate system has an influence on the usage of the component. But this is not the case, as shown in the next figure:



In the figure the **local** axes of rotation of the components are shown. The connection of two inertias in the left and in the right part of the figure are completely equivalent, i.e., the right part is just a different drawing of the left part. This is due to the fact, that by a connection, the two local coordinate systems are made identical and the (automatically) generated connection equations (= angles are identical, cut-torques sum-up to zero) are also expressed in this common coordinate system. Therefore, even if in the left figure it seems to be that the angular velocity vector of  $J_2$  goes from right to left, in reality it goes from left to right as shown in the right part of the figure, where the local coordinate systems are drawn such that they are aligned. Note, that the simple rule stated in section 4 (Sign conventions) also determines that the angular velocity of  $J_2$  in the left part of the figure is directed from left to right.

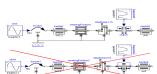
To summarize, the local coordinate system selected for a component is just necessary, in order that the equations of this component are expressed correctly. The selection of the coordinate system is arbitrary and has no influence on the usage of the component. Especially, the actual direction of, e.g., a cut-torque is most easily determined by the rule of section 4. A more strict determination by aligning coordinate systems and then using the vector direction of the local coordinate systems, often requires a re-drawing of the diagram and is therefore less convenient to use.

### Modelica.Mechanics.Rotational.UsersGuide.RequirementsForSimulationTool

This library is designed in a fully object oriented way in order that components can be connected together in every meaningful combination (e.g. direct connection of two springs or two inertias). As a consequence, most models lead to a system of differential-algebraic equations of **index 3** (= constraint equations have to be differentiated twice in order to arrive at a state space representation) and the Modelica translator or the simulator has to cope with this system representation. According to our present knowledge, this requires that the Modelica translator is able to symbolically differentiate equations (otherwise it is e.g. not possible to provide consistent initial conditions; even if consistent initial conditions are present, most numerical DAE integrators can cope at most with index 2 DAEs).



The elements of this library can be connected together in an arbitrary way. However, difficulties may occur, if the elements which can **lock** the **relative motion** between two flanges are connected **rigidly** together such that essentially the **same relative motion** can be locked. The reason is that the cut-torque in the locked phase is not uniquely defined if the elements are locked at the same time instant (i.e., there does not exist a unique solution) and some simulation systems may not be able to handle this situation, since this leads to a singularity during simulation. Currently, this type of problem can occur with the Coulomb friction elements **BearingFriction**, **Clutch**, **Brake**, **LossyGear** when the elements become stuck:



In the figure above two typical situations are shown: In the upper part of the figure, the series connection of rigidly attached BearingFriction and Clutch components are shown. This does not hurt, because the BearingFriction element can lock the relative motion between the element and the housing, whereas the clutch element can lock the relative motion between the two connected flanges. Contrary, the drive train in the lower part of the figure may rise to simulation problems, because the BearingFriction element and the Brake element can lock the relative motion between a flange and the housing and these flanges are rigidly connected together, i.e., essentially the same relative motion can be locked. These difficulties may be solved by either introducing a compliance between these flanges or by combining the BearingFriction and Brake element into one component and resolving the ambiguity of the frictional torque in the stuck mode. A tool may handle this situation also **automatically**, by picking one solution of the infinitely many, e.g., the one

where the difference to the value of the previous time instant is as small as possible.

---

### **Modelica.Mechanics.Rotational.UsersGuide.ReleaseNotes**



- *December 12, 2005* by [Martin Otter](#):  
Package documentation split into several parts and provided as User's Guide.
- *October 27, 2003* by [Martin Otter](#) and [Christian Schweiger](#):  
Bearing flanges added for mounted components and support torque computation implemented.  
New component Torque2 and new example ElasticBearing.
- *October 21, 2002* by [Martin Otter](#) and [Christian Schweiger](#):  
New components **LossyGear** (with corresponding examples) and **Gear2**.  
Interface **FrictionBase** adapted to new initialization.
- *June 19, 2000* by [Martin Otter](#):  
New elements:  
`IdealGearR2T`      Ideal gear transforming rotational in translational motion  
`Position`              Forced movement of a flange with a reference angle given as input signal  
`RelativeStates`        Definition of relative state variables  
Icon of Rotational.Torque changed. Elements Acceleration, Torque, Fixed, Sensors ordered according to the Translational library.
- *Nov. 4, 1999* by [Martin Otter](#):  
Improved documentation and improved graphical layout of the diagram level. Changes according to the Twente meeting introduced. Especially: Alias names, instead of extends. Model Shaft renamed to Inertia. Torque1D renamed to Torque. AccMotion renamed to Accelerate. LockedL, LockedR replaced by Fixed. SpeedSensor splitted into AngleSensor and SpeedSensor. RelSpeedSensor splitted into RelAngleSensor and RelSpeedSensor. Initialization of friction elements improved. Flanges renamed to flange\_a, flange\_b. MoveAngle renamed to KinematicPTP, vectorized and moved to Blocks.Sources.  
Advice given from P. Beater, H. Elmquist, S.E. Mattsson, H. Olsson is appreciated.
- *July 18, 1999* by [Martin Otter](#):  
Documentation and icons improved. Appropriate initial conditions introduced as start values in the demo models. Bearing model replaced by FixedRight and FixedLeft models; sensor elements replaced by TorqueSensor, SpeedSensor, AccSensor; new sensor elements RelSpeedSensor, RelAccSensor to measure relative kinematic quantitites. New elements GearEfficiency and Gear.
- *June 30, 1999* by [Martin Otter](#):  
Realized a first version based on an existing Dymola library of Martin Otter and Hilding Elmquist.

---

### **Modelica.Mechanics.Rotational.UsersGuide.Contact**



#### **Main Authors:**

[Martin Otter](#) and [Christian Schweiger](#)  
Deutsches Zentrum für Luft und Raumfahrt e.V. (DLR)  
Institut für Robotik und Mechatronik  
Abteilung für Entwurfsorientierte Regelungstechnik  
Postfach 1116  
D-82230 Wessling  
Germany  
email: [Martin.Otter@dlr.de](mailto:Martin.Otter@dlr.de), [Christian.Schweiger@dlr.de](mailto:Christian.Schweiger@dlr.de)

---

### **Modelica.Mechanics.Rotational.Examples**

#### **Demonstration examples of the components of this package**

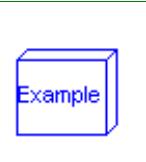
## Information

This package contains example models to demonstrate the usage of the Modelica.Mechanics.Rotational package. Open the models and simulate them according to the provided description in the models.

## Package Content

Name	Description
<a href="#">First</a>	First example: simple drive train
<a href="#">Friction</a>	Drive train with clutch and brake
<a href="#">CoupledClutches</a>	Drive train with 3 dynamically coupled clutches
<a href="#">LossyGearDemo1</a>	Example to show that gear efficiency may lead to stuck motion
<a href="#">LossyGearDemo2</a>	Example to show combination of LossyGear and BearingFriction
<a href="#">ElasticBearing</a>	Example to show possible usage of bearing flange

## Modelica.Mechanics.Rotational.Examples.First



### First example: simple drive train

#### Information

The drive train consists of a motor inertia which is driven by a sine-wave motor torque. Via a gearbox the rotational energy is transmitted to a load inertia. Elasticity in the gearbox is modeled by a spring element. A linear damper is used to model the damping in the gearbox bearing.

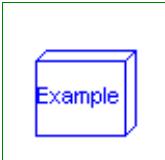
Note, that a force component (like the damper of this example) which is acting between a shaft and the housing has to be fixed in the housing on one side via component Fixed.

Simulate for 1 second and plot the following variables:  
angular velocities of inertias inertia2 and 3: inertia2.w, inertia3.w

## Parameters

Name	Default	Description
amplitude	10	
freqHz	5	[Hz]
Jmotor	0.1	[kg.m <sup>2</sup> ]
Jload	2	[kg.m <sup>2</sup> ]
ratio	10	
damping	10	

## Modelica.Mechanics.Rotational.Examples.Friction



### Drive train with clutch and brake

#### Information

This drive train contains a frictional **clutch** and a **brake**. Simulate the system for 1 second using the following initial values (defined already in the model):

```
inertial.w = 90 (or brake.w)
```

```
inertia2.w = 90
inertia3.w = 100
```

Plot the output signals

tMotor	Torque of motor
tClutch	Torque in clutch
tBrake	Torque in brake
tSpring	Torque in spring

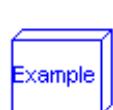
as well as the absolute angular velocities of the three inertia components (inertia1.w, inertia2.w, inertia3.w).

## Parameters

Name	Default	Description
startTime	0.5	Start time of step [s]

---

## Modelica.Mechanics.Rotational.Examples.CoupledClutches



Drive train with 3 dynamically coupled clutches

### Information

This example demonstrates how variable structure drive trains are handled. The drive train consists of 4 inertias and 3 clutches, where the clutches are controlled by input signals. The system has  $2^3=8$  different configurations and  $3^3 = 27$  different states (every clutch may be in forward sliding, backward sliding or locked mode when the relative angular velocity is zero). By invoking the clutches at different time instances, the switching of the configurations can be studied.

Simulate the system for 1.2 seconds with the following initial values:

J1.w = 10.

Plot the following variables:

angular velocities of inertias (J1.w, J2.w, J3.w, J4.w), frictional torques of clutches (clutchX.tau), frictional mode of clutches (clutchX.mode) where mode = -1/0/+1 means backward sliding, locked, forward sliding.

## Parameters

Name	Default	Description
freqHz	0.2	frequency of sine function to invoke clutch1 [Hz]
T2	0.4	time when clutch2 is invoked [s]
T3	0.9	time when clutch3 is invoked [s]

---

## Modelica.Mechanics.Rotational.Examples.LossyGearDemo1



Example to show that gear efficiency may lead to stuck motion

### Information

This model contains two inertias which are connected by an ideal gear where the friction between the teeth of the gear is modeled in a physical meaningful way (friction may lead to stuck mode which locks the motion of the gear). The friction is defined by an efficiency factor (= 0.5) for forward and backward driving condition leading to a torque dependent friction loss. Simulate for about 0.5 seconds. The friction in the gear will take all modes (forward and backward rolling, as well as stuck).

## 594 Modelica.Mechanics.Rotational.Examples.LossyGearDemo1

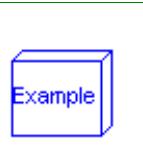
---

You may plot:

```
Inertial1.w,  
Inertia2.w : angular velocities of inertias  
powerLoss : power lost in the gear  
gear.mode : 1 = forward rolling  
            0 = stuck (w=0)  
           -1 = backward rolling
```

---

## Modelica.Mechanics.Rotational.Examples.LossyGearDemo2



Example to show combination of LossyGear and BearingFriction

### Information

This model contains bearing friction and gear friction (= efficiency). If both friction models are stuck, there is no unique solution. Still a reliable Modelica simulator, such as Dymola, should be able to handle this situation.

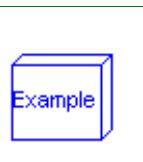
Simulate for about 0.5 seconds. The friction elements are in all modes (forward and backward rolling, as well as stuck).

You may plot:

```
Inertial1.w,  
Inertia2.w : angular velocities of inertias  
powerLoss : power lost in the gear  
bearingFriction.mode: 1 = forward rolling  
                      0 = stuck (w=0)  
                     -1 = backward rolling  
gear.mode : 1 = forward rolling  
            0 = stuck (w=0)  
           -1 = backward rolling
```

Note: This combination of LossyGear and BearingFriction is not recommended to use, as component LossyGear includes the functionality of component BearingFriction (only peak not supported).

---



## Modelica.Mechanics.Rotational.Examples.ElasticBearing

Example to show possible usage of bearing flange

### Information

This model demonstrates the usage of the bearing flange. The gearbox is not connected rigidly to the ground, but by a spring-damper-system. This allows examination of the gearbox housing dynamics.

Simulate for about 10 seconds and plot the angular velocities of the inertias `housing.w`, `shaft.w` and `load.w`.

---



## Modelica.Mechanics.Rotational.Sensors

Sensors to measure variables in 1D rotational mechanical components

## Information

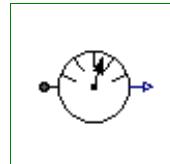
This package contains ideal sensor components that provide the connector variables as signals for further processing with the Modelica.Blocks library.

## Package Content

Name	Description
 AngleSensor	Ideal sensor to measure the absolute flange angle
 SpeedSensor	Ideal sensor to measure the absolute flange angular velocity
 AccSensor	Ideal sensor to measure the absolute flange angular acceleration
 RelAngleSensor	Ideal sensor to measure the relative angle between two flanges
 RelSpeedSensor	Ideal sensor to measure the relative angular velocity between two flanges
 RelAccSensor	Ideal sensor to measure the relative angular acceleration between two flanges
 TorqueSensor	Ideal sensor to measure the torque between two flanges (= flange_a.tau)
 PowerSensor	Ideal sensor to measure the power between two flanges (= flange_a.tau*der(flange_a.phi))

### Modelica.Mechanics.Rotational.Sensors.AngleSensor

Ideal sensor to measure the absolute flange angle



#### Information

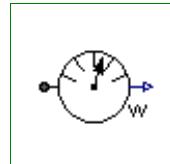
Measures the **absolute angle phi** of a flange in an ideal way and provides the result as output signal **phi** (to be further processed with blocks of the Modelica.Blocks library).

#### Connectors

Name	Description
flange_a	flange to be measured
phi	Absolute angle of flange

### Modelica.Mechanics.Rotational.Sensors.SpeedSensor

Ideal sensor to measure the absolute flange angular velocity

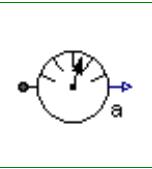


#### Information

Measures the **absolute angular velocity w** of a flange in an ideal way and provides the result as output signal **w** (to be further processed with blocks of the Modelica.Blocks library).

#### Connectors

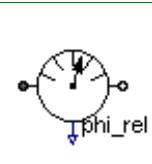
Name	Description
flange_a	flange to be measured
w	Absolute angular velocity of flange

**Modelica.Mechanics.Rotational.Sensors.AccSensor****Ideal sensor to measure the absolute flange angular acceleration****Information**

Measures the **absolute angular acceleration  $a$**  of a flange in an ideal way and provides the result as output signal  $a$  (to be further processed with blocks of the Modelica.Blocks library).

**Connectors**

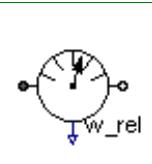
Name	Description
flange_a	flange to be measured
a	Absolute angular acceleration of flange

**Modelica.Mechanics.Rotational.Sensors.RelAngleSensor****Ideal sensor to measure the relative angle between two flanges****Information**

Measures the **relative angle  $\phi_{rel}$**  between two flanges in an ideal way and provides the result as output signal  $\phi_{rel}$  (to be further processed with blocks of the Modelica.Blocks library).

**Connectors**

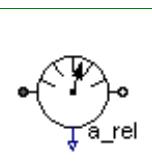
Name	Description
flange_a	driving flange (flange axis directed INTO cut plane)
flange_b	(right) driven flange (flange axis directed OUT OF cut plane)
phi_rel	Relative angle between two flanges (= flange_b.phi - flange_a.phi)

**Modelica.Mechanics.Rotational.Sensors.RelSpeedSensor****Ideal sensor to measure the relative angular velocity between two flanges****Information**

Measures the **relative angular velocity  $w_{rel}$**  between two flanges in an ideal way and provides the result as output signal  $w_{rel}$  (to be further processed with blocks of the Modelica.Blocks library).

**Connectors**

Name	Description
flange_a	driving flange (flange axis directed INTO cut plane)
flange_b	(right) driven flange (flange axis directed OUT OF cut plane)
w_rel	Relative angular velocity between two flanges (= der(flange_b.phi) - der(flange_a.phi))

**Modelica.Mechanics.Rotational.Sensors.RelAccSensor****Ideal sensor to measure the relative angular acceleration between two flanges**

## Information

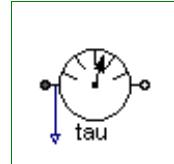
Measures the **relative angular acceleration  $a_{rel}$**  between two flanges in an ideal way and provides the result as output signal  $a_{rel}$  (to be further processed with blocks of the Modelica.Blocks library).

## Connectors

Name	Description
flange_a	driving flange (flange axis directed INTO cut plane)
flange_b	(right) driven flange (flange axis directed OUT OF cut plane)
$a_{rel}$	Relative angular acceleration between two flanges

## Modelica.Mechanics.Rotational.Sensors.TorqueSensor

Ideal sensor to measure the torque between two flanges (= flange\_a.tau)



## Information

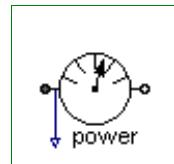
Measures the **cut-torque between two flanges** in an ideal way and provides the result as output signal  $\tau$  (to be further processed with blocks of the Modelica.Blocks library).

## Connectors

Name	Description
flange_a	
flange_b	
$\tau$	Torque in flange flange_a and flange_b (= flange_a.tau = -flange_b.tau)

## Modelica.Mechanics.Rotational.Sensors.PowerSensor

Ideal sensor to measure the power between two flanges (= flange\_a.tau\*der(flang\_a.phi))



## Information

Measures the **power between two flanges** in an ideal way and provides the result as output signal  $power$  (to be further processed with blocks of the Modelica.Blocks library).

## Connectors

Name	Description
flange_a	
flange_b	
power	Power in flange flange_a

## Modelica.Mechanics.Rotational.Interfaces

Connectors and partial models for 1D rotational mechanical components

## Information

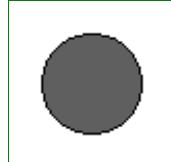
This package contains connectors and partial models for 1-dim. rotational mechanical components. The components of this package can only be used as basic building elements for models.

## Package Content

Name	Description
 Flange_a	1D rotational flange (filled square icon)
 Flange_b	1D rotational flange (non-filled square icon)
 Rigid	Base class for the rigid connection of two rotational 1D flanges
 Compliant	Base class for the compliant connection of two rotational 1D flanges
 TwoFlanges	Base class for a component with two rotational 1D flanges
 Bearing	Base class for interface classes with bearing connector
 TwoFlangesAndBearing	Base class for a equation-based component with two rotational 1D flanges and one rotational 1D bearing flange
 TwoFlangesAndBearingH	Base class for a hierarchically composed component with two rotational 1D flanges and one rotational bearing flange
FrictionBase	Base class of Coulomb friction elements
 AbsoluteSensor	Base class to measure a single absolute flange variable
 RelativeSensor	Base class to measure a single relative variable between two flanges
 PartialSpeedDependentTorque	Partial model of a torque acting at the flange (accelerates the flange)

## Modelica.Mechanics.Rotational.Interfaces.Flange\_a

1D rotational flange (filled square icon)



## Information

This is a connector for 1D rotational mechanical systems and models a mechanical flange. The following variables are defined in this connector:

**phi**: Absolute rotation angle of the flange in [rad].  
**tau**: Cut-torque in the flange in [Nm].

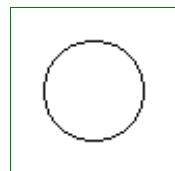
There is a second connector for flanges: Flange\_b. The connectors Flange\_a and Flange\_b are completely identical. There is only a difference in the icons, in order to easier identify a flange variable in a diagram. For a discussion on the actual direction of the cut-torque tau and of the rotation angle, see the information text of package Rotational (section 4. Sign conventions).

If needed, the absolute angular velocity w and the absolute angular acceleration a of the flange can be determined by differentiation of the flange angle phi:

$$w = \text{der}(\phi); \quad a = \text{der}(w)$$

## Contents

Name	Description
phi	Absolute rotation angle of flange [rad]
tau	Cut torque in the flange [N.m]

**Modelica.Mechanics.Rotational.Interfaces.Flange\_b****1D rotational flange (non-filled square icon)****Information**

This is a connector for 1D rotational mechanical systems and models a mechanical flange. The following variables are defined in this connector:

**phi**: Absolute rotation angle of the flange in [rad].  
**tau**: Cut-torque in the flange in [Nm].

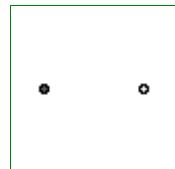
There is a second connector for flanges: Flange\_a. The connectors Flange\_a and Flange\_b are completely identical. There is only a difference in the icons, in order to easier identify a flange variable in a diagram. For a discussion on the actual direction of the cut-torque tau and of the rotation angle, see the information text of package Rotational (section 4. Sign conventions).

If needed, the absolute angular velocity w and the absolute angular acceleration a of the flange can be determined by differentiation of the flange angle phi:

$$w = \text{der}(\phi); \quad a = \text{der}(w)$$

**Contents**

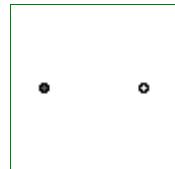
Name	Description
phi	Absolute rotation angle of flange [rad]
tau	Cut torque in the flange [N.m]

**Modelica.Mechanics.Rotational.Interfaces.Rigid****Base class for the rigid connection of two rotational 1D flanges****Information**

This is a 1D rotational component with two rigidly connected flanges, i.e., flange\_a.phi = flange\_b.phi. It is used e.g. to built up components with inertia.

**Connectors**

Name	Description
flange_a	(left) driving flange (flange axis directed INTO cut plane)
flange_b	(right) driven flange (flange axis directed OUT OF cut plane)

**Modelica.Mechanics.Rotational.Interfaces.Compliant****Base class for the compliant connection of two rotational 1D flanges****Information**

This is a 1D rotational component with a compliant connection of two rotational 1D flanges where inertial effects between the two flanges are neglected. The basic assumption is that the cut-torques of the two flanges sum-up to zero, i.e., they have the same absolute value but opposite sign: flange\_a.tau +

## 600 Modelica.Mechanics.Rotational.Interfaces.Compliant

---

flange\_b.tau = 0. This base class is used to built up force elements such as springs, dampers, friction.

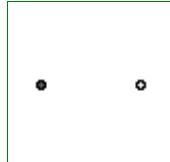
### Connectors

Name	Description
flange_a	(left) driving flange (flange axis directed INTO cut plane)
flange_b	(right) driven flange (flange axis directed OUT OF cut plane)

---

## Modelica.Mechanics.Rotational.Interfaces.TwoFlanges

Base class for a component with two rotational 1D flanges



### Information

This is a 1D rotational component with two flanges. It is used e.g. to build up parts of a drive train consisting of several base components. There are specialized versions of this base class for rigidly connected flanges (Interfaces.Rigid) and for a compliant connection of flanges (Interfaces.Compliant).

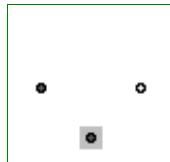
### Connectors

Name	Description
flange_a	
flange_b	

---

## Modelica.Mechanics.Rotational.Interfaces.Bearing

Base class for interface classes with bearing connector



### Information

This is a 1D rotational component with two flanges and an additional bearing flange. It is a superclass for the two components TwoFlangesAndBearing and TwoFlangesAndBearingH.

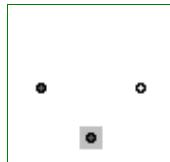
### Connectors

Name	Description
flange_a	
flange_b	
bearing	

---

## Modelica.Mechanics.Rotational.Interfaces.TwoFlangesAndBearing

Base class for a equation-based component with two rotational 1D flanges and one rotational 1D bearing flange



### Information

This is a 1D rotational component with two flanges and an additional bearing flange. It is used e.g. to build up equation-based parts of a drive train.

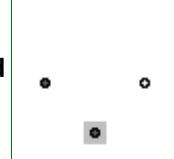
## Connectors

Name	Description
flange_a	
flange_b	
bearing	

---

## Modelica.Mechanics.Rotational.Interfaces.TwoFlangesAndBearingH

Base class for a hierarchically composed component with two rotational 1D flanges and one rotational bearing flange



## Information

This is a 1D rotational component with two flanges and an additional bearing flange. It is used e.g. to build up parts of a drive train consisting of several base components.

## Connectors

Name	Description
flange_a	
flange_b	
bearing	

---

## Modelica.Mechanics.Rotational.Interfaces.FrictionBase

Base class of Coulomb friction elements

## Information

Basic model for Coulomb friction that models the stuck phase in a reliable way.

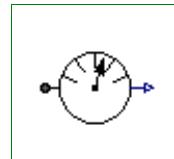
## Parameters

Name	Default	Description
<b>Advanced</b>		
w_smal_l	1e10	Relative angular velocity near to zero if jumps due to a reinit(..) of the velocity can occur (set to low value only if such impulses can occur) [rad/s]

---

## Modelica.Mechanics.Rotational.Interfaces.AbsoluteSensor

Base class to measure a single absolute flange variable



## Information

This is the base class of a 1D rotational component with one flange and one output signal  $y$  in order to measure an absolute kinematic quantity in the flange and to provide the measured signal as output signal for further processing with the blocks of package Modelica.Blocks.

## 602 Modelica.Mechanics.Rotational.Interfaces.AbsoluteSensor

---

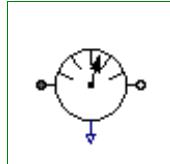
### Connectors

Name	Description
flange_a	(left) flange to be measured (flange axis directed INTO cut plane)
y	

---

## Modelica.Mechanics.Rotational.Interfaces.RelativeSensor

Base class to measure a single relative variable between two flanges



### Information

This is a base class for 1D rotational components with two rigidly connected flanges and one output signal  $y$  in order to measure relative kinematic quantities between the two flanges or the cut-torque in the flange and to provide the measured signal as output signal for further processing with the blocks of package Modelica.Blocks.

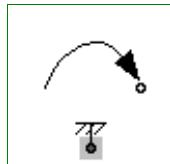
### Connectors

Name	Description
flange_a	(left) driving flange (flange axis directed INTO cut plane)
flange_b	(right) driven flange (flange axis directed OUT OF cut plane)
y	

---

## Modelica.Mechanics.Rotational.Interfaces.PartialSpeedDependentTorque

Partial model of a torque acting at the flange (accelerates the flange)



### Information

Partial model of torque dependent on speed that accelerates the flange.

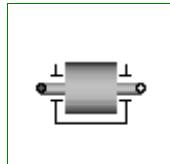
### Connectors

Name	Description
flange	Flange on which torque is acting
bearing	Bearing at which the reaction torque (i.e., -flange.tau) is acting

---

## Modelica.Mechanics.Rotational.Inertia

1D-rotational component with inertia



### Information

Rotational component with **inertia** and two rigidly connected flanges.

### Parameters

Name	Default	Description
J	1	Moment of inertia [kg.m <sup>2</sup> ]

---

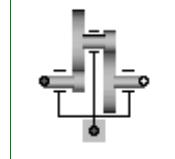
Initialization		
initType	Modelica.Mechanics.Rotationa...	Type of initialization (defines usage of start values below)
phi_start	0	Initial or guess value of rotor rotation angle phi [rad]
w_start	0	Initial or guess value of angular velocity w = der(phi) [rad/s]
a_start	0	Initial value of angular acceleration a = der(w) [rad/s <sup>2</sup> ]
phi.start	phi_start	Absolute rotation angle of component (= flange_a.phi = flange_b.phi) [rad]
Advanced		
stateSelection	Modelica.Blocks.Types.StateS...	Priority to use phi and w as states

## Connectors

Name	Description
flange_a	(left) driving flange (flange axis directed INTO cut plane)
flange_b	(right) driven flange (flange axis directed OUT OF cut plane)

## Modelica.Mechanics.Rotational.IdealGear

Ideal gear without inertia



## Information

This element characterizes any type of gear box which is fixed in the ground and which has one driving shaft and one driven shaft. The gear is **ideal**, i.e., it does not have inertia, elasticity, damping or backlash. If these effects have to be considered, the gear has to be connected to other elements in an appropriate way.

## Parameters

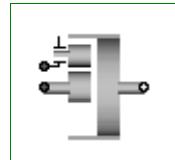
Name	Default	Description
ratio	1	Transmission ratio (flange_a.phi/flange_b.phi)

## Connectors

Name	Description
flange_a	
flange_b	
bearing	

## Modelica.Mechanics.Rotational.IdealPlanetary

Ideal planetary gear box



## Information

The IdealPlanetary gear box is an ideal gear without inertia, elasticity, damping or backlash consisting of an inner **sun** wheel, an outer **ring** wheel and a **planet** wheel located between sun and ring wheel. The bearing of the planet wheel shaft is fixed in the planet **carrier**. The component can be connected to other elements at the sun, ring and/or carrier flanges. It is not possible to connect to the planet wheel. If inertia shall not be neglected, the sun, ring and carrier inertias can be easily added by attaching inertias (= model Inertia) to the corresponding connectors. The inertias of the planet wheels are always neglected.

The icon of the planetary gear signals that the sun and carrier flanges are on the left side and the ring flange

## 604 Modelica.Mechanics.Rotational.IdealPlanetary

---

is on the right side of the gear box. However, this component is generic and is valid independantly how the flanges are actually placed (e.g. sun wheel may be placed on the right side instead on the left side in reality).

The ideal planetary gearbox is uniquely defined by the ratio of the number of ring teeth  $z_r$  with respect to the number of sun teeth  $z_s$ . For example, if there are 100 ring teeth and 50 sun teeth then ratio =  $z_r/z_s = 2$ . The number of planet teeth  $z_p$  has to fulfill the following relationship:

$$z_p := (z_r - z_s) / 2$$

Therefore, in the above example  $z_p = 25$  is required.

According to the overall convention, the positive direction of all vectors, especially the absolute angular velocities and cut-torques in the flanges, are along the axis vector displayed in the icon.

### Parameters

Name	Default	Description
ratio	100/50	number of ring_teeth/sun_teeth (e.g. ratio=100/50)

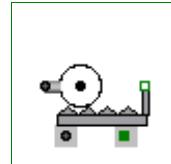
### Connectors

Name	Description
sun	sun flange (flange axis directed INTO cut plane)
carrier	carrier flange (flange axis directed INTO cut plane)
ring	ring flange (flange axis directed OUT OF cut plane)

---

## Modelica.Mechanics.Rotational.IdealGearR2T

Gearbox transforming rotational into translational motion



### Information

This is an ideal mass- and inertialess gearbox which transforms a 1D-rotational into a 1D-translational motion. If elasticity, damping or backlash has to be considered, this ideal gearbox has to be connected with corresponding elements.

### Parameters

Name	Default	Description
ratio	1	transmission ratio (flange_a.phi/flange_b.s) [rad/m]

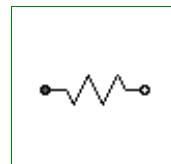
### Connectors

Name	Description
flange_a	
flange_b	
bearingR	
bearingT	

---

## Modelica.Mechanics.Rotational.Spring

Linear 1D rotational spring



## Information

A **linear 1D rotational spring**. The component can be connected either between two inertias/gears to describe the shaft elasticity, or between a inertia/gear and the housing (component Fixed), to describe a coupling of the element with the housing via a spring.

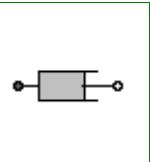
## Parameters

Name	Default	Description
c		Spring constant [N.m/rad]
phi_rel0	0	Unstretched spring angle [rad]
Initialization		
phi_rel.start	0	Relative rotation angle (= flange_b.phi - flange_a.phi) [rad]

## Connectors

Name	Description
flange_a	(left) driving flange (flange axis directed INTO cut plane)
flange_b	(right) driven flange (flange axis directed OUT OF cut plane)

## Modelica.Mechanics.Rotational.Damper



Linear 1D rotational damper

## Information

**Linear, velocity dependent damper** element. It can be either connected between an inertia or gear and the housing (component Fixed), or between two inertia/gear elements.

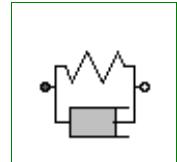
## Parameters

Name	Default	Description
d	0	Damping constant [N.m.s/rad]
Initialization		
phi_rel.start	0	Relative rotation angle (= flange_b.phi - flange_a.phi) [rad]

## Connectors

Name	Description
flange_a	(left) driving flange (flange axis directed INTO cut plane)
flange_b	(right) driven flange (flange axis directed OUT OF cut plane)

## Modelica.Mechanics.Rotational.SpringDamper



Linear 1D rotational spring and damper in parallel

## Information

A **spring and damper** element **connected in parallel**. The component can be connected either between two inertias/gears to describe the shaft elasticity and damping, or between an inertia/gear and the housing (component Fixed), to describe a coupling of the element with the housing via a spring/damper.

## 606 Modelica.Mechanics.Rotational.SpringDamper

---

### Parameters

Name	Default	Description
c		Spring constant [N.m/rad]
phi_rel0	0	Unstretched spring angle [rad]
d	0	Damping constant [N.m.s/rad]
<b>Initialization</b>		
initType	Modelica.Mechanics.Rotationa...	Type of initialization (defines usage of start values below)
phi_rel_start	0	Initial or guess value of relative rotation angle phi_rel [rad]
w_rel_start	0	Initial or guess value of relative angular velocity w_rel = der(phi_rel) [rad/s]
phi_rel.start	phi_rel_start	Relative rotation angle (= flange_b.phi - flange_a.phi) [rad]
<b>Advanced</b>		
stateSelection	Modelica.Blocks.Types.StateS...	Priority to use phi_rel and w_rel as states

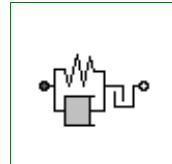
### Connectors

Name	Description
flange_a	(left) driving flange (flange axis directed INTO cut plane)
flange_b	(right) driven flange (flange axis directed OUT OF cut plane)

---

## Modelica.Mechanics.Rotational.ElastoBacklash

Backlash connected in series to linear spring and damper (backlash is modeled with elasticity)



### Information

This element consists of a **backlash** element **connected in series** to a **spring** and **damper** element which are **connected in parallel**. The spring constant shall be non-zero, otherwise the component cannot be used.

In combination with components IdealGear, the ElastoBacklash model can be used to model a gear box with backlash, elasticity and damping.

### Parameters

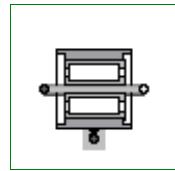
Name	Default	Description
b	0	Total backlash [rad]
c	1.e5	Spring constant ( $c > 0$ required) [N.m/rad]
phi_rel0	0	Unstretched spring angle [rad]
d	0	Damping constant [N.m.s/rad]
<b>Initialization</b>		
phi_rel.start	0	Relative rotation angle (= flange_b.phi - flange_a.phi) [rad]

### Connectors

Name	Description
flange_a	(left) driving flange (flange axis directed INTO cut plane)
flange_b	(right) driven flange (flange axis directed OUT OF cut plane)

## Modelica.Mechanics.Rotational.BearingFriction

### Coulomb friction in bearings



### Information

This element describes **Coulomb friction in bearings**, i.e., a frictional torque acting between a flange and the housing. The positive sliding friction torque "tau" has to be defined by table "tau\_pos" as function of the absolute angular velocity "w". E.g.

w		tau
0		0
1		2
2		5
3		8

gives the following table:

```
tau_pos = [0, 0; 1, 2; 2, 5; 3, 8];
```

Currently, only linear interpolation in the table is supported. Outside of the table, extrapolation through the last two table entries is used. It is assumed that the negative sliding friction force has the same characteristic with negative values. Friction is modelled in the following way:

When the absolute angular velocity "w" is not zero, the friction torque is a function of w and of a constant normal force. This dependency is defined via table tau\_pos and can be determined by measurements, e.g. by driving the gear with constant velocity and measuring the needed motor torque (= friction torque).

When the absolute angular velocity becomes zero, the elements connected by the friction element become stuck, i.e., the absolute angle remains constant. In this phase the friction torque is calculated from a torque balance due to the requirement, that the absolute acceleration shall be zero. The elements begin to slide when the friction torque exceeds a threshold value, called the maximum static friction torque, computed via:

```
maximum_static_friction = peak * sliding_friction(w=0) (peak >= 1)
```

This procedure is implemented in a "clean" way by state events and leads to continuous/discrete systems of equations if friction elements are dynamically coupled which have to be solved by appropriate numerical methods. The method is described in:

Otter M., Elmquist H., and Mattsson S.E. (1999):

**Hybrid Modeling in Modelica based on the Synchronous Data Flow Principle.** CACSD'99, Aug. 22.-26, Hawaii.

More precise friction models take into account the elasticity of the material when the two elements are "stuck", as well as other effects, like hysteresis. This has the advantage that the friction element can be completely described by a differential equation without events. The drawback is that the system becomes stiff (about 10-20 times slower simulation) and that more material constants have to be supplied which requires more sophisticated identification. For more details, see the following references, especially (Armstrong and Canudas de Witt 1996):

Armstrong B. (1991):

**Control of Machines with Friction.** Kluwer Academic Press, Boston MA.

Armstrong B., and Canudas de Wit C. (1996):

**Friction Modeling and Compensation.** The Control Handbook, edited by W.S.Levine, CRC Press, pp. 1369-1382.

Canudas de Wit C., Olsson H., Astrom K.J., and Lischinsky P. (1995):

**A new model for control of systems with friction.** IEEE Transactions on Automatic Control, Vol. 40,

No. 3, pp. 419-425.

## Parameters

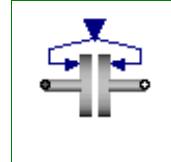
Name	Default	Description
tau_pos[:, :]	[0, 1]	[w,tau] Positive sliding friction characteristic ( $w \geq 0$ )
peak	1	peak*tau_pos[1,2] = Maximum friction torque for $w=0$
<b>Initialization</b>		
startForward.start	<b>false</b>	true, if $w_{rel}=0$ and start of forward sliding or $w_{rel} > w_{small}$
startBackward.start	<b>false</b>	true, if $w_{rel}=0$ and start of backward sliding or $w_{rel} < -w_{small}$
locked.start	false	true, if $w_{rel}=0$ and not sliding
<b>Advanced</b>		
w_small	1e10	Relative angular velocity near to zero if jumps due to a reinit(..) of the velocity can occur (set to low value only if such impulses can occur) [rad/s]

## Connectors

Name	Description
flange_a	
flange_b	
bearing	

## Modelica.Mechanics.Rotational.Clutch

Clutch based on Coulomb friction



## Information

This component models a **clutch**, i.e., a component with two flanges where friction is present between the two flanges and these flanges are pressed together via a normal force. The normal force  $f_n$  has to be provided as input signal  $f_{normalized}$  in a normalized form ( $0 \leq f_{normalized} \leq 1$ ),  $f_n = f_{n\_max} * f_{normalized}$ , where  $f_{n\_max}$  has to be provided as parameter. Friction in the clutch is modelled in the following way:

When the relative angular velocity is not zero, the friction torque is a function of the velocity dependent friction coefficient  $\mu_{friction}(w_{rel})$ , of the normal force " $f_n$ ", and of a geometry constant "cgeo" which takes into account the geometry of the device and the assumptions on the friction distributions:

$$\text{frictional\_torque} = \text{cgeo} * \mu_{friction}(w_{rel}) * f_n$$

Typical values of coefficients of friction:

dry operation :	$\mu_{friction} = 0.2 \dots 0.4$
operating in oil:	$\mu_{friction} = 0.05 \dots 0.1$

When plates are pressed together, where  $r_i$  is the inner radius,  $r_o$  is the outer radius and  $N$  is the number of friction interfaces, the geometry constant is calculated in the following way under the assumption of a uniform rate of wear at the interfaces:

$$\text{cgeo} = N * (r_o + r_i) / 2$$

The positive part of the friction characteristic  $\mu_{friction}(w_{rel})$ ,  $w_{rel} \geq 0$ , is defined via table  $\mu_{friction\_pos}$  (first column =  $w_{rel}$ , second column =  $\mu_{friction}$ ). Currently, only linear interpolation in the table is supported.

When the relative angular velocity becomes zero, the elements connected by the friction element become

stuck, i.e., the relative angle remains constant. In this phase the friction torque is calculated from a torque balance due to the requirement, that the relative acceleration shall be zero. The elements begin to slide when the friction torque exceeds a threshold value, called the maximum static friction torque, computed via:

$$\text{frictional\_torque} = \text{peak} * \text{cgeo} * \text{mue}(\text{w\_rel}=0) * \text{fn} \quad (\text{peak} \geq 1)$$

This procedure is implemented in a "clean" way by state events and leads to continuous/discrete systems of equations if friction elements are dynamically coupled. The method is described in:

Otter M., Elmquist H., and Mattsson S.E. (1999):

**Hybrid Modeling in Modelica based on the Synchronous Data Flow Principle.** CACSD'99, Aug. 22.-26, Hawaii.

More precise friction models take into account the elasticity of the material when the two elements are "stuck", as well as other effects, like hysteresis. This has the advantage that the friction element can be completely described by a differential equation without events. The drawback is that the system becomes stiff (about 10-20 times slower simulation) and that more material constants have to be supplied which requires more sophisticated identification. For more details, see the following references, especially (Armstrong and Canudas de Witt 1996):

Armstrong B. (1991):

**Control of Machines with Friction.** Kluwer Academic Press, Boston MA.

Armstrong B., and Canudas de Wit C. (1996):

**Friction Modeling and Compensation.** The Control Handbook, edited by W.S.Levine, CRC Press, pp. 1369-1382.

Canudas de Wit C., Olsson H., Astrom K.J., and Lischinsky P. (1995):

**A new model for control of systems with friction.** IEEE Transactions on Automatic Control, Vol. 40, No. 3, pp. 419-425.

## Parameters

Name	Default	Description
mu_pos[:, :]	[0, 0.5]	[w,mue] positive sliding friction coefficient ( $w_{\text{rel}} \geq 0$ )
peak	1	peak*mue_pos[1,2] = maximum value of mue for $w_{\text{rel}}=0$
cgeo	1	Geometry constant containing friction distribution assumption
fn_max	1	Maximum normal force [N]
<b>Initialization</b>		
phi_rel.start	0	Relative rotation angle (= flange_b.phi - flange_a.phi) [rad]
startForward.start	false	true, if $w_{\text{rel}}=0$ and start of forward sliding or $w_{\text{rel}} > w_{\text{small}}$
startBackward.start	false	true, if $w_{\text{rel}}=0$ and start of backward sliding or $w_{\text{rel}} < -w_{\text{small}}$
locked.start	false	true, if $w_{\text{rel}}=0$ and not sliding
<b>Advanced</b>		
w_small	1e10	Relative angular velocity near to zero if jumps due to a reinit(..) of the velocity can occur (set to low value only if such impulses can occur) [rad/s]

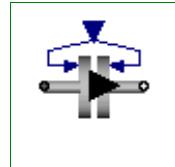
## Connectors

Name	Description
flange_a	(left) driving flange (flange axis directed INTO cut plane)
flange_b	(right) driven flange (flange axis directed OUT OF cut plane)

f\_normalized | Normalized force signal 0..1 (normal force = fn\_max\*f\_normalized; clutch is engaged if > 0)

## Modelica.Mechanics.Rotational.OneWayClutch

Series connection of freewheel and clutch



### Information

This component models a **one-way clutch**, i.e., a component with two flanges where friction is present between the two flanges and these flanges are pressed together via a normal force. These flanges maybe sliding with respect to each other Parallel connection of ClutchCombi and of FreeWheel. The element is introduced to resolve the ambiguity of the constraint torques of the elements.

A one-way-clutch is an element where a clutch is connected in parallel to a free wheel. This special element is provided, because such a parallel connection introduces an ambiguity into the model (the constraint torques are not uniquely defined when both elements are stuck) and this element resolves it by introducing **one** constraint torque and not two.

Note, initial values have to be chosen for the model, such that the relative speed of the one-way-clutch  $\geq 0$ . Otherwise, the configuration is physically not possible and an error occurs.

The normal force  $fn$  has to be provided as input signal  $f\_normalized$  in a normalized form ( $0 \leq f\_normalized \leq 1$ ),  $fn = fn\_max * f\_normalized$ , where  $fn\_max$  has to be provided as parameter. Friction in the clutch is modelled in the following way:

When the relative angular velocity is positive, the friction torque is a function of the velocity dependent friction coefficient  $mue(w\_rel)$ , of the normal force "fn", and of a geometry constant "cgeo" which takes into account the geometry of the device and the assumptions on the friction distributions:

$$\text{frictional\_torque} = cgeo * mue(w\_rel) * fn$$

Typical values of coefficients of friction:

dry operation :	<b>mue</b> = 0.2 .. 0.4
operating in oil:	<b>mue</b> = 0.05 .. 0.1

When plates are pressed together, where  $ri$  is the inner radius,  $ro$  is the outer radius and  $N$  is the number of friction interfaces, the geometry constant is calculated in the following way under the assumption of a uniform rate of wear at the interfaces:

$$cgeo = N * (ro + ri) / 2$$

The positive part of the friction characteristic  $mue(w\_rel)$ ,  $w\_rel \geq 0$ , is defined via table  $mue\_pos$  (first column =  $w\_rel$ , second column =  $mue$ ). Currently, only linear interpolation in the table is supported.

When the relative angular velocity becomes zero, the elements connected by the friction element become stuck, i.e., the relative angle remains constant. In this phase the friction torque is calculated from a torque balance due to the requirement, that the relative acceleration shall be zero. The elements begin to slide when the friction torque exceeds a threshold value, called the maximum static friction torque, computed via:

$$\text{frictional\_torque} = peak * cgeo * mue(w\_rel=0) * fn \quad (peak \geq 1)$$

This procedure is implemented in a "clean" way by state events and leads to continuous/discrete systems of equations if friction elements are dynamically coupled. The method is described in:

Otter M., Elmquist H., and Mattsson S.E. (1999):

**Hybrid Modeling in Modelica based on the Synchronous Data Flow Principle.** CACSD'99, Aug. 22.-26, Hawaii.

## Parameters

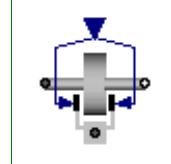
Name	Default	Description
mue_pos[:, :]	[0, 0.5]	[w,mue] positive sliding friction coefficient (w_rel>=0)
peak	1	peak*mue_pos[1,2] = maximum value of mue for w_rel==0
cgeo	1	Geometry constant containing friction distribution assumption
fn_max	1	Maximum normal force [N]
Initialization		
phi_rel.start	0	Relative rotation angle (= flange_b.phi - flange_a.phi) [rad]
Advanced		
w_small	1e10	Relative angular velocity near to zero if jumps due to a reinit(..) of the velocity can occur (set to low value only if such impulses can occur) [rad/s]

## Connectors

Name	Description
flange_a	(left) driving flange (flange axis directed INTO cut plane)
flange_b	(right) driven flange (flange axis directed OUT OF cut plane)
f_normalized	Normalized force signal 0..1 (normal force = fn_max*f_normalized; clutch is engaged if > 0)

## Modelica.Mechanics.Rotational.Brake

Brake based on Coulomb friction



## Information

This component models a **brake**, i.e., a component where a frictional torque is acting between the housing and a flange and a controlled normal force presses the flange to the housing in order to increase friction. The normal force *fn* has to be provided as input signal *f\_normalized* in a normalized form ( $0 \leq f_{\text{normalized}} \leq 1$ ),  $fn = fn_{\text{max}} * f_{\text{normalized}}$ , where *fn\_max* has to be provided as parameter. Friction in the brake is modelled in the following way:

When the absolute angular velocity "w" is not zero, the friction torque is a function of the velocity dependent friction coefficient *mue(w)*, of the normal force "fn", and of a geometry constant "cgeo" which takes into account the geometry of the device and the assumptions on the friction distributions:

$$\text{frictional\_torque} = cgeo * \text{muc}(w) * fn$$

Typical values of coefficients of friction:

dry operation : *muc* = 0.2 .. 0.4  
operating in oil: *muc* = 0.05 .. 0.1

When plates are pressed together, where *ri* is the inner radius, *ro* is the outer radius and *N* is the number of friction interfaces, the geometry constant is calculated in the following way under the assumption of a uniform rate of wear at the interfaces:

$$cgeo = N * (ro + ri) / 2$$

The positive part of the friction characteristic *muc(w)*,  $w \geq 0$ , is defined via table *mue\_pos* (first column = *w*, second column = *muc*). Currently, only linear interpolation in the table is supported.

When the absolute angular velocity becomes zero, the elements connected by the friction element become stuck, i.e., the absolute angle remains constant. In this phase the friction torque is calculated from a torque

## 612 Modelica.Mechanics.Rotational.Brake

---

balance due to the requirement, that the absolute acceleration shall be zero. The elements begin to slide when the friction torque exceeds a threshold value, called the maximum static friction torque, computed via:

$$\text{frictional\_torque} = \text{peak} * \text{cgeo} * \text{mue}(\omega=0) * \text{fn} \quad (\text{peak} \geq 1)$$

This procedure is implemented in a "clean" way by state events and leads to continuous/discrete systems of equations if friction elements are dynamically coupled. The method is described in:

Otter M., Elmquist H., and Mattsson S.E. (1999):

**Hybrid Modeling in Modelica based on the Synchronous Data Flow Principle.** CACSD'99, Aug. 22.-26, Hawaii.

More precise friction models take into account the elasticity of the material when the two elements are "stuck", as well as other effects, like hysteresis. This has the advantage that the friction element can be completely described by a differential equation without events. The drawback is that the system becomes stiff (about 10-20 times slower simulation) and that more material constants have to be supplied which requires more sophisticated identification. For more details, see the following references, especially (Armstrong and Canudas de Witt 1996):

Armstrong B. (1991):

**Control of Machines with Friction.** Kluwer Academic Press, Boston MA.

Armstrong B., and Canudas de Wit C. (1996):

**Friction Modeling and Compensation.** The Control Handbook, edited by W.S.Levine, CRC Press, pp. 1369-1382.

Canudas de Wit C., Olsson H., Astrom K.J., and Lischinsky P. (1995):

**A new model for control of systems with friction.** IEEE Transactions on Automatic Control, Vol. 40, No. 3, pp. 419-425.

## Parameters

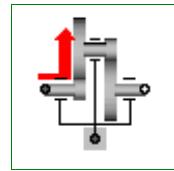
Name	Default	Description
mue_pos[:, :]	[0, 0.5]	[w,mue] positive sliding friction coefficient ( $w_{\text{rel}} \geq 0$ )
peak	1	$\text{peak} * \text{mue\_pos}[1,2] = \text{maximum value of mue for } w_{\text{rel}} == 0$
cgeo	1	Geometry constant containing friction distribution assumption
fn_max	1	Maximum normal force [N]
Initialization		
startForward.start	<b>false</b>	true, if $w_{\text{rel}}=0$ and start of forward sliding or $w_{\text{rel}} > w_{\text{small}}$
startBackward.start	<b>false</b>	true, if $w_{\text{rel}}=0$ and start of backward sliding or $w_{\text{rel}} < -w_{\text{small}}$
locked.start	false	true, if $w_{\text{rel}}=0$ and not sliding
Advanced		
w_small	1e10	Relative angular velocity near to zero if jumps due to a reinit(..) of the velocity can occur (set to low value only if such impulses can occur) [rad/s]

## Connectors

Name	Description
flange_a	
flange_b	
bearing	
f_normalized	Normalized force signal 0..1 (normal force = $\text{fn\_max} * \text{f\_normalized}$ ; brake is active if $> 0$ )

## Modelica.Mechanics.Rotational.LossyGear

Gear with mesh efficiency and bearing friction (stuck/rolling possible)



### Information

This component models the gear ratio and the **losses** of a standard gear box in a **reliable** way including the stuck phases that may occur at zero speed. The gear boxes that can be handled are fixed in the ground, have one input and one output shaft, and are essentially described by the equations:

$$\begin{aligned} \text{flange\_a.phi} &= i * \text{flange\_b.phi} \\ (-\text{flange\_b.tau}) &= i * (\eta_{mf} * \text{flange\_a.tau} - \tau_{bf}) \end{aligned}$$

where

- $i$  is the constant **gear ratio**,
- $\eta_{mf} = \eta_{mf}(w)$  is the **mesh efficiency** due to the friction between the teeth of the gear wheels,
- $\tau_{bf} = \tau_{bf}(w)$  is the **bearing friction torque**, and
- $w_a = \text{der}(\text{flange\_a.phi})$  is the speed of flange\_a

The loss terms " $\eta_{mf}$ " and " $\tau_{bf}$ " are functions of the *absolute value* of the input shaft speed  $w_a$  and of the energy flow direction. They are defined by parameter **lossTable[:,5]** where the columns of this table have the following meaning:

$ w_a $	$\eta_{mf1}$	$\eta_{mf2}$	$ \tau_{bf1} $	$ \tau_{bf2} $
...	...	...	...	...
...	...	...	...	...

with

$ w_a $	Absolute value of angular velocity of input shaft flange_a
$\eta_{mf1}$	Mesh efficiency in case of input shaft driving
$\eta_{mf2}$	Mesh efficiency in case of output shaft driving
$ \tau_{bf1} $	Absolute bearing friction torque in case of input shaft driving
$ \tau_{bf2} $	Absolute bearing friction torque in case of output shaft driving

With these variables, the mesh efficiency and the bearing friction are formally defined as:

```

if flange_a.tau*w_a > 0 or flange_a.tau==0 and w_a > 0 then
    eta_mf := eta_mf1
    tau_bf := tau_bf1
elseif flange_a.tau*w_a < 0 or flange_a.tau==0 and w_a < 0 then
    eta_mf := 1/eta_mf2
    tau_bf := tau_bf2
else // w_a == 0
    eta_mf and tau_bf are computed such that der(w_a) = 0
end if;

```

Note, that the losses are modeled in a physically meaningful way taking into account that at zero speed the movement may be locked due to the friction in the gear teeth and/or in the bearings. Due to this important property, this component can be used in situations where the combination of the components Modelica.Mechanics.Rotational.IdealGear and Modelica.Mechanics.Rotational.GearEfficiency will fail because, e.g., chattering occurs when using the Modelica.Mechanics.Rotational.GearEfficiency model.

**Acknowledgement:** The essential idea to model efficiency in this way is from Christoph Pelchen, ZF Friedrichshafen.

### For detailed information:

Pelchen C., Schweiger C., and Otter M.: "Modeling and Simulating the Efficiency of Gearboxes and of Planetary Gearboxes," in *Proceedings of the 2nd International Modelica Conference, Oberpfaffenhofen, Germany*, pp. 257-266, The Modelica Association and Institute of Robotics and Mechatronics, Deutsches

## 614 Modelica.Mechanics.Rotational.LossyGear

Zentrum für Luft- und Raumfahrt e. V., March 18-19, 2002.

### Parameters

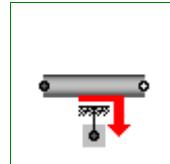
Name	Default	Description
i	1	Transmission ratio (flange_a.phi/flange_b.phi)
lossTable[:, 5]	[0, 1, 1, 0, 0]	Array for mesh efficiencies and bearing friction depending on speed

### Connectors

Name	Description
flange_a	
flange_b	
bearing	

## Modelica.Mechanics.Rotational.GearEfficiency

Obsolete component (use model LossyGear instead)



### Information

THIS COMPONENT IS **OBSOLETE** and should **no longer be used**. It is only kept for **backward compatibility** purposes. Use model Modelica.Mechanics.Rotational.LossyGear instead which implements gear efficiency in a much more reliable way.

This component consists of two rigidly connected flanges flange\_a and flange\_b without inertia where an **efficiency** coefficient **eta** reduces the driven torque as function of the driving torque depending on the direction of the energy flow, i.e., energy is always lost. This can be seen as a simple model of the Coulomb friction acting between the teeth of a gearbox.

Note, that most gearbox manufacturers provide tables of the efficiency of a gearbox as function of the angular velocity (efficiency becomes zero, if the angular velocity is zero). However, such a table is practically useless for simulation purposes, because in gearboxes always two types of friction is present: (1) Friction in the **bearings** and (2) friction between the teeth of the gear. (1) leads to a velocity dependent, additive loss-torque, whereas (2) leads to a torque-dependent reduction of the driving torque. The gearbox manufacturers measure both effects together and determine the gear efficiency from it, although for simulation purposes the two effects need to be separated. Assume for example that only constant bearing friction, i.e., bearingTorque=const., is present, i.e.,

$$(1) \quad \text{loadTorque} = \text{motorTorque} - \text{sign}(w) * \text{bearingTorque}$$

Gearbox manufacturers use the loss-formula

$$(2) \quad \text{loadTorque} = \text{eta} * \text{motorTorque}$$

Comparing (1) and (2) gives a formula for the efficiency eta:

$$\text{eta} = (1 - \text{sign}(w) * \text{bearingTorque}) / \text{motorTorque}$$

When the motorTorque becomes smaller as the bearingTorque, (2) is useless, because the efficiency is zero. To summarize, be careful to determine the gear **efficiency** of this element from tables of the gear manufacturers.

### Parameters

Name	Default	Description
------	---------	-------------

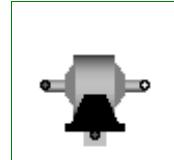
eta	1	Efficiency
-----	---	------------

## Connectors

Name	Description
flange_a	
flange_b	
bearing	

## Modelica.Mechanics.Rotational.Gear

Realistic model of a gearbox



## Information

This component models the essential effects of a gearbox, in particular gear **efficiency** due to friction between the teeth, **bearing friction**, gear **elasticity** and **damping**, **backlash**. The inertia of the gear wheels is not modeled. If necessary, inertia has to be taken into account by connecting components of model Inertia to the left and/or the right flange.

## Parameters

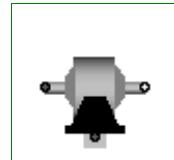
Name	Default	Description
ratio	1	transmission ratio (flange_a.phi/flange_b.phi)
eta	1	Gear efficiency
friction_pos[:, :]	[0, 1]	[w,tau] positive sliding friction characteristic (w>=0)
peak	1	peak*friction_pos[1,2] = maximum friction torque at zero velocity
c	1.e5	Gear elasticity (spring constant) [N.m/rad]
d	0	(relative) gear damping [N.m.s/rad]
b	0	Total backlash [rad]

## Connectors

Name	Description
flange_a	
flange_b	
bearing	

## Modelica.Mechanics.Rotational.Gear2

Realistic model of a gearbox (based on LossyGear)



## Information

This component models the essential effects of a gearbox, in particular

- in component **lossyGear**
  - gear **efficiency** due to friction between the teeth
  - **bearing friction**
- in component **elastoBacklash**
  - gear **elasticity**
  - **damping**

- **backlash**

The inertia of the gear wheels is not modeled. If necessary, inertia has to be taken into account by connecting components of model Inertia to the left and/or the right flange of component GearNew.

## Parameters

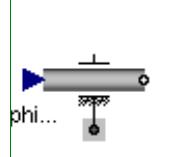
Name	Default	Description
i	1	transmission ratio (flange_a.phi/flange_b.phi)
lossTable[:, 5]	[0, 1, 1, 0, 0]	Array for mesh efficiencies and bearing friction depending on speed (see docu of LossyGear)
c	1.e5	Gear elasticity (spring constant) [N.m/rad]
d	0	(relative) gear damping [N.m.s/rad]
b	0	Total backlash [rad]

## Connectors

Name	Description
flange_a	
flange_b	
bearing	

## Modelica.Mechanics.Rotational.Position

Forced movement of a flange according to a reference angle signal



## Information

The input signal **phi\_ref** defines the **reference angle** in [rad]. Flange **flange\_b** is **forced** to move according to this reference motion. According to parameter **exact** (default = **false**), this is done in the following way:

1. **exact=true**

The reference angle is treated **exactly**. This is only possible, if the input signal is defined by an analytical function which can be differentiated at least twice. If this prerequisite is fulfilled, the Modelica translator will differentiate the input signal twice in order to compute the reference acceleration of the flange.

2. **exact=false**

The reference angle is **filtered** and the second derivative of the filtered curve is used to compute the reference acceleration of the flange. This second derivative is **not** computed by numerical differentiation but by an appropriate realization of the filter. For filtering, a second order Bessel filter is used. The critical frequency (also called cut-off frequency) of the filter is defined via parameter **f\_crit** in [Hz]. This value should be selected in such a way that it is higher as the essential low frequencies in the signal.

The input signal can be provided from one of the signal generator blocks of the block library Modelica.Blocks.Sources.

## Parameters

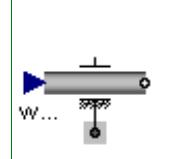
Name	Default	Description
exact	false	true/false exact treatment/filtering the input signal
f_crit	50	if exact=false, critical frequency of filter to filter input signal [Hz]

## Connectors

Name	Description
phi_ref	reference angle of flange_b as input signal
flange_b	
bearing	

## Modelica.Mechanics.Rotational.Speed

Forced movement of a flange according to a reference angular velocity signal



## Information

The input signal **w\_ref** defines the **reference angle** in [rad]. Flange **flange\_b** is **forced** to move according to this reference motion. According to parameter **exact** (default = **false**), this is done in the following way:

### 1. exact=true

The reference angle is treated **exactly**. This is only possible, if the input signal is defined by an analytical function which can be differentiated at least twice. If this prerequisite is fulfilled, the Modelica translator will differentiate the input signal twice in order to compute the reference acceleration of the flange.

### 2. exact=false

The reference angle is **filtered** and the second derivative of the filtered curve is used to compute the reference acceleration of the flange. This second derivative is **not** computed by numerical differentiation but by an appropriate realization of the filter. For filtering, a second order Bessel filter is used. The critical frequency (also called cut-off frequency) of the filter is defined via parameter **f\_crit** in [Hz]. This value should be selected in such a way that it is higher as the essential low frequencies in the signal.

The input signal can be provided from one of the signal generator blocks of the block library Modelica.Blocks.Sources.

## Release Notes:

- October 27, 2003 by Christian Schweiger.  
Realized based on component **Position** (implemented by Martin Otter).

## Parameters

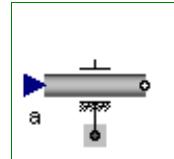
Name	Default	Description
exact	false	true/false exact treatment/filtering the input signal
f_crit	50	if exact=false, critical frequency of filter to filter input signal [Hz]

## Connectors

Name	Description
flange_b	
w_ref	Reference angular velocity of flange_b as input signal
bearing	

## Modelica.Mechanics.Rotational.Accelerate

Forced movement of a flange according to an acceleration signal



## Information

The input signal **a** defines an **angular acceleration** in [rad/s<sup>2</sup>]. Flange **flange\_b** is **forced** to move with this acceleration. The angular velocity **w** and the rotation angle **phi** of the flange are automatically determined by integration of the acceleration.

The input signal can be provided from one of the signal generator blocks of the block library Modelica.Blocks.Sources.

## Parameters

Name	Default	Description
phi_start	0	Start angle [rad]
w_start	0	Start angular velocity [rad/s]

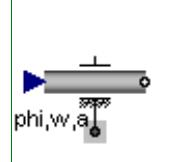
## Connectors

Name	Description
flange_b	
a	absolute angular acceleration of flange_b as input signal
bearing	

---

## Modelica.Mechanics.Rotational.Move

Forced movement of a flange according to an angle, speed and angular acceleration signal



## Information

Flange **flange\_b** is **forced** to move with a predefined motion according to the input signals:

- u[1]: angle of flange
- u[2]: angular velocity of flange
- u[3]: angular acceleration of flange

The user has to guarantee that the input signals are consistent to each other, i.e., that u[2] is the derivative of u[1] and that u[3] is the derivative of u[2]. There are, however, also applications where by purpose these conditions do not hold. For example, if only the position dependent terms of a mechanical system shall be calculated, one may provide angle = angle(t) and set the angular velocity and the angular acceleration to zero.

The input signals can be provided from one of the signal generator blocks of the block library Modelica.Blocks.Sources.

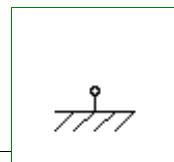
## Connectors

Name	Description
u[3]	angle, angular velocity and angular acceleration of flange_b as input signals
flange_b	Flange that is forced to move according to input signals u
bearing	Bearing flange (if not connected, it is assumed that it is fixed on ground)

---

## Modelica.Mechanics.Rotational.Fixed

Flange fixed in housing at a given angle



## Information

The **flange** of a 1D rotational mechanical system is **fixed** at an angle phi0 in the **housing**. May be used:

- to connect a compliant element, such as a spring or a damper, between an inertia or gearbox component and the housing.
- to fix a rigid element, such as an inertia, with a specific angle to the housing.

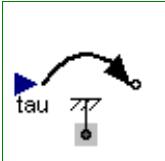
## Parameters

Name	Default	Description
phi0	0	Fixed offset angle of housing [rad]

## Connectors

Name	Description
flange_b	(right) flange fixed in housing

## Modelica.Mechanics.Rotational.Torque



Input signal acting as external torque on a flange

## Information

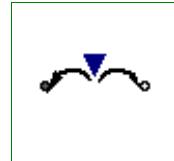
The input signal **tau** defines an external torque in [Nm] which acts (with negative sign) at a flange connector, i.e., the component connected to this flange is driven by torque **tau**.

The input signal can be provided from one of the signal generator blocks of Modelica.Blocks.Sources.

## Connectors

Name	Description
tau	Torque driving the flange (a positive value accelerates the flange)
flange_b	(Right) flange
bearing	

## Modelica.Mechanics.Rotational.Torque2



Input signal acting as torque on two flanges

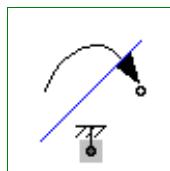
## Information

The input signal **tau** defines an external torque in [Nm] which acts at both flange connectors, i.e., the components connected to these flanges are driven by torque **tau**.

The input signal can be provided from one of the signal generator blocks of Modelica.Blocks.Sources.

## Connectors

Name	Description
flange_a	
flange_b	
tau	Torque driving the two flanges (a positive value accelerates the flange)

**Modelica.Mechanics.Rotational.LinearSpeedDependentTorque****Linear dependency of torque versus speed****Information**

Model of torque, linearly dependent on angular velocity of flange.

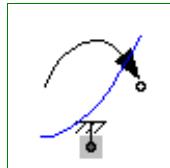
Parameter TorqueDirection chooses whether direction of torque is the same in both directions of rotation or not.

**Parameters**

Name	Default	Description
tau_nominal		nominal torque (if negative, torque is acting as load) [N.m]
TorqueDirection	true	same direction of torque in both directions of rotation
w_nominal		nominal speed [rad/s]

**Connectors**

Name	Description
flange	Flange on which torque is acting
bearing	Bearing at which the reaction torque (i.e., -flange.tau) is acting

**Modelica.Mechanics.Rotational.QuadraticSpeedDependentTorque****Quadratic dependency of torque versus speed****Information**

Model of torque, quadratic dependent on angular velocity of flange.

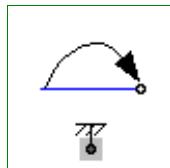
Parameter TorqueDirection chooses whether direction of torque is the same in both directions of rotation or not.

**Parameters**

Name	Default	Description
tau_nominal		nominal torque (if negative, torque is acting as load) [N.m]
TorqueDirection	true	same direction of torque in both directions of rotation
w_nominal		nominal speed [rad/s]

**Connectors**

Name	Description
flange	Flange on which torque is acting
bearing	Bearing at which the reaction torque (i.e., -flange.tau) is acting

**Modelica.Mechanics.Rotational.ConstantTorque****Constant torque, not dependent on speed**

## Information

Model of constant torque, not dependent on angular velocity of flange.  
Positive torque acts accelerating.

## Parameters

Name	Default	Description
tau_constant		constant torque (if negative, torque is acting as load) [N.m]

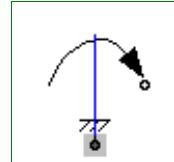
## Connectors

Name	Description
flange	Flange on which torque is acting
bearing	Bearing at which the reaction torque (i.e., -flange.tau) is acting

---

## Modelica.Mechanics.Rotational.ConstantSpeed

Constant speed, not dependent on torque



## Information

Model of **fixed** angular velocity of flange, not dependent on torque.

## Parameters

Name	Default	Description
w_fixed		fixed speed (if negative, torque is acting as load) [rad/s]

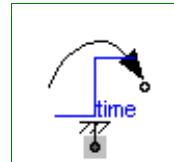
## Connectors

Name	Description
flange	Flange on which torque is acting
bearing	Bearing at which the reaction torque (i.e., -flange.tau) is acting

---

## Modelica.Mechanics.Rotational.TorqueStep

Constant torque, not dependent on speed



## Information

Model of a torque step at time .  
Positive torque acts accelerating.

## Parameters

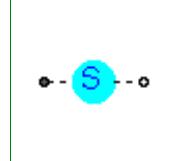
Name	Default	Description
stepTorque	1	height of torque step (if negative, torque is acting as load) [N.m]
offsetTorque	0	offset of torque [N.m]
startTime	0	output = offset for time < startTime [s]

## Connectors

Name	Description
flange	Flange on which torque is acting
bearing	Bearing at which the reaction torque (i.e., -flange.tau) is acting

## Modelica.Mechanics.Rotational.RelativeStates

### Definition of relative state variables

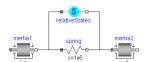


### Information

Usually, the absolute angle and the absolute angular velocity of Modelica.Mechanics.Rotational.Inertia models are used as state variables. In some circumstances, relative quantities are better suited, e.g., because it may be easier to supply initial values. In such cases, model **RelativeStates** allows the definition of state variables in the following way:

- Connect an instance of this model between two flange connectors.
- The **relative rotation angle** and the **relative angular velocity** between the two connectors are used as **state variables**.

An example is given in the next figure



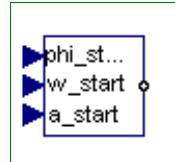
Here, the relative angle and the relative angular velocity between the two inertias are used as state variables. Additionally, the simulator selects either the absolute angle and absolute angular velocity of model inertia1 or of model inertia2 as state variables.

## Connectors

Name	Description
flange_a	
flange_b	

## Modelica.Mechanics.Rotational.InitializeFlange

Initializes a flange with pre-defined angle, speed and angular acceleration (usually, this is reference data from a control bus)



### Information

This component is used to optionally initialize the angle, speed, and/or angular acceleration of the flange to which this component is connected. Via parameters use\_phi\_start, use\_w\_start, use\_a\_start the corresponding input signals phi\_start, w\_start, a\_start are conditionally activated. If an input is activated, the corresponding flange property is initialized with the input value at start time.

For example, if "use\_phi\_start = true", then flange.phi is initialized with the value of the input signal "phi\_start" at the start time.

Additionally, it is optionally possible to define the "StateSelect" attribute of the flange angle and the flange speed via parameter "stateSelection".

This component is especially useful when the initial values of a flange shall be set according to reference signals of a controller that are provided via a signal bus.

## Parameters

Name	Default	Description
use_phi_start	true	= true, if initial angle is defined by input phi_start, otherwise not initialized
use_w_start	true	= true, if initial speed is defined by input w_start, otherwise not initialized
use_a_start	true	= true, if initial angular acceleration is defined by input a_start, otherwise not initialized
stateSelection	Modelica.Blocks.Types.StateS...	Priority to use flange angle and speed as states

## Connectors

Name	Description
phi_start	Initial angle of flange
w_start	Initial speed of flange
a_start	Initial angular acceleration of flange
flange	Flange that is initialized

## Modelica.Mechanics.Rotational.Types

Constants and types with choices, especially to build menus

## Information

In this package **types** and **constants** are defined that are used in library Modelica.Blocks. The types have additional annotation choices definitions that define the menus to be built up in the graphical user interface when the type is used as parameter in a declaration.

## Package Content

Name	Description
(e) Init	Type, constants and menu choices to define initialization of absolute rotational quantities
(e) InitRel	Type, constants and menu choices to define initialization of relative rotational quantities

## Modelica.Mechanics.Rotational.Types.Init

Type, constants and menu choices to define initialization of absolute rotational quantities

## Information

Type **Init** defines initialization of absolute rotational quantities.

## Package Content

Name	Description
NoInit=1	no initialization (phi_start, w_start are guess values)
SteadyState=2	steady state initialization ( $\text{der}(\phi)=\text{der}(w)=0$ )
InitialState=3	initialization with phi_start, w_start
InitialAngle=4	initialization with phi_start

InitialSpeed=5	initialization with w_start
InitialAcceleration=6	initialization with a_start
InitialAngleAcceleration=7	initialization with phi_start, a_start
InitialSpeedAcceleration=8	initialization with w_start, a_start
InitialAngleSpeedAcceleration=9	initialization with phi_start, w_start, a_start
 Temp	Temporary type of absolute initialization with choices for menus (until enumerations are available)

## Types and constants

```

constant Integer NoInit=1
"no initialization (phi_start, w_start are guess values)";

constant Integer SteadyState=2
"steady state initialization (der(phi)=der(w)=0)";

constant Integer InitialState=3 "initialization with phi_start, w_start";

constant Integer InitialAngle=4 "initialization with phi_start";

constant Integer InitialSpeed=5 "initialization with w_start";

constant Integer InitialAcceleration=6 "initialization with a_start";

constant Integer InitialAngleAcceleration=7
"initialization with phi_start, a_start";

constant Integer InitialSpeedAcceleration=8
"initialization with w_start, a_start";

constant Integer InitialAngleSpeedAcceleration=9
"initialization with phi_start, w_start, a_start";

type Temp
"Temporary type of absolute initialization with choices for menus (until
enumerations are available)"
  extends Modelica.Icons.TypeInteger(min=1,max=9);

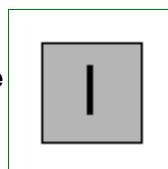
end Temp;

```

---

## Modelica.Mechanics.Rotational.Types.Init.Temp

Temporary type of absolute initialization with choices for menus (until enumerations are available)



## Parameters

Name	Default	Description
min	1	
max	9	

## Modelica.Mechanics.Rotational.Types.InitRel

Type, constants and menu choices to define initialization of relative rotational quantities

### Information

Type **Init** defines initialization of relative rotational quantities.

### Package Content

Name	Description
NoInit=1	no initialization (phi_rel_start, w_rel_start are guess values)
SteadyState=2	steady state initialization (der(phi_rel)=der(w_rel)=0)
InitialState=3	initialization with phi_rel_start, w_rel_start
InitialAngle=4	initialization with phi_rel_start
InitialSpeed=5	initialization with w_rel_start
 Temp	Temporary type of absolute initialization with choices for menus (until enumerations are available)

### Types and constants

```

constant Integer NoInit=1
"no initialization (phi_rel_start, w_rel_start are guess values)";

constant Integer SteadyState=2
"steady state initialization (der(phi_rel)=der(w_rel)=0)";

constant Integer InitialState=3
"initialization with phi_rel_start, w_rel_start";

constant Integer InitialAngle=4 "initialization with phi_rel_start";

constant Integer InitialSpeed=5 "initialization with w_rel_start";

type Temp
"Temporary type of absolute initialization with choices for menus (until
enumerations are available)"
extends Modelica.Icons.TypeInteger(min=1,max=5);

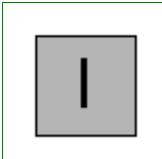
end Temp;

```

---

## Modelica.Mechanics.Rotational.Types.InitRel.Temp

Temporary type of absolute initialization with choices for menus (until enumerations are available)



### Parameters

Name	Default	Description
min	1	
max	5	

## Modelica.Mechanics.Translational

Library to model 1-dimensional, translational mechanical systems

### Information

This package contains components to model *1-dimensional translational mechanical systems*.

The *filled and non-filled green squares* at the left and right side of a component represent *mechanical flanges*. Drawing a line between such squares means that the corresponding flanges are *rigidly attached* to each other. The components of this library can be usually connected together in an arbitrary way. E.g. it is possible to connect two springs or two sliding masses with inertia directly together.

The only *connection restriction* is that the Coulomb friction elements (Stop) should be only connected together provided a compliant element, such as a spring, is in between. The reason is that otherwise the frictional force is not uniquely defined if the elements are stuck at the same time instant (i.e., there does not exist a unique solution) and some simulation systems may not be able to handle this situation, since this leads to a singularity during simulation. It can only be resolved in a "clean way" by combining the two connected friction elements into one component and resolving the ambiguity of the frictional force in the stuck mode.

Another restriction arises if the hard stops in model Stop are used, i. e. the movement of the mass is limited by a stop at smax or smin. **This requires the states Stop.s and Stop.v**. If these states are eliminated during the index reduction the model will not work. To avoid this any inertias should be connected via springs to the Stop element, other sliding masses, dampers or hydraulic chambers must be avoided.

In the *icon* of every component an *arrow* is displayed in grey color. This arrow characterizes the coordinate system in which the vectors of the component are resolved. It is directed into the positive translational direction (in the mathematical sense). In the flanges of a component, a coordinate system is rigidly attached to the flange. It is called *flange frame* and is directed in parallel to the component coordinate system. As a result, e.g., the positive cut-force of a "left" flange (flange\_a) is directed into the flange, whereas the positive cut-force of a "right" flange (flange\_b) is directed out of the flange. A flange is described by a Modelica connector containing the following variables:

```
SIunits.Position s      "absolute position of flange";  
flow Force f           "cut-force in the flange";
```

This library is designed in a fully object oriented way in order that components can be connected together in every meaningful combination (e.g. direct connection of two springs or two shafts with inertia). As a consequence, most models lead to a system of differential-algebraic equations of *index 3* (= constraint equations have to be differentiated twice in order to arrive at a state space representation) and the Modelica translator or the simulator has to cope with this system representation. According to our present knowledge, this requires that the Modelica translator is able to symbolically differentiate equations (otherwise it is e.g. not possible to provide consistent initial conditions; even if consistent initial conditions are present, most numerical DAE integrators can cope at most with index 2 DAEs).

#### Main Author:

Peter Beater  
Universität Paderborn, Abteilung Soest  
Fachbereich Maschinenbau/Automatisierungstechnik  
Lübecker Ring 2  
D 59494 Soest  
Germany  
email: [Beater@mailso.uni-paderborn.de](mailto:Beater@mailso.uni-paderborn.de)

Copyright © 1998-2007, Modelica Association and Universität Paderborn, FB 12.

*This Modelica package is free software; it can be redistributed and/or modified under the terms of the Modelica license, see the license conditions and the accompanying disclaimer here.*

## Package Content

Name	Description
Examples	Demonstration examples of the components of this package
Sensors	Sensors for 1-dim. translational mechanical quantities
Interfaces	Interfaces for 1-dim. translational mechanical components
SlidingMass	Sliding mass with inertia
Stop	Sliding mass with hard stop and Stribeck friction
Rod	Rod without inertia
Spring	Linear 1D translational spring
Damper	Linear 1D translational damper
SpringDamper	Linear 1D translational spring and damper in parallel
ElastoGap	1D translational spring damper combination with gap
Position	Forced movement of a flange according to a reference position
Speed	Forced movement of a flange according to a reference speed
Accelerate	Forced movement of a flange according to an acceleration signal
Move	Forced movement of a flange according to a position, velocity and acceleration signal
Fixed	Fixed flange
Force	External force acting on a drive train element as input signal
RelativeStates	Definition of relative state variables

## Modelica.Mechanics.Translational.Examples

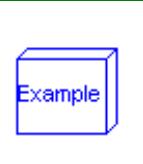
### Demonstration examples of the components of this package

## Information

This package contains example models to demonstrate the usage of the Translational package. Open the models and simulate them according to the provided description in the models.

## Package Content

Name	Description
SignConvention	Examples for the used sign conventions.
InitialConditions	Setting of initial conditions
WhyArrows	Use of arrows in Mechanics.Translational
Accelerate	Use of model accelerate.
Damper	Use of damper models.
Oscillator	Oscillator demonstrates the use of initial conditions.
Sensors	Sensors for translational systems.
Friction	Use of model Stop
PreLoad	Preload of a spool using ElastoGap models.

**Modelica.Mechanics.Translational.Examples.SignConvention**

**Examples for the used sign conventions.**

**Information**

If all arrows point in the same direction a positive force results in a positive acceleration a, velocity v and position s.

For a force of 1 N and a mass of 1 Kg this leads to

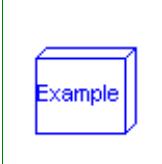
$$\begin{aligned} a &= 1 \text{ m/s}^2 \\ v &= 1 \text{ m/s after } 1 \text{ s (SlidingMass1.v)} \\ s &= 0.5 \text{ m after } 1 \text{ s (SlidingMass1.s)} \end{aligned}$$

The acceleration is not available for plotting.

System 1) and 2) are equivalent. It doesn't matter whether the force pushes at flange\_a in system 1 or pulls at flange\_b in system 2.

It is of course possible to ignore the arrows and connect the models in an arbitrary way. But then it is hard see in what direction the force acts.

In the third system the two arrows are opposed which means that the force acts in the opposite direction (in the same direction as in the two other examples).

**Modelica.Mechanics.Translational.Examples.InitialConditions**

**Setting of initial conditions**

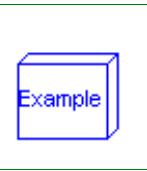
**Information**

There are several ways to set initial conditions. In the first system the position of the sliding mass m3 was defined by using the modifier s(start=4.5), the position of m5 by s(start=12.5). These positions were chosen such that the system is a rest. To calculate these values start at the left (Fixed1) with a value of 1 m. The spring has an unstretched length of 2 m and m3 an length of 3 m, which leads to

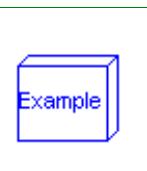
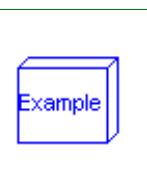
$$\begin{aligned} 1 &\text{ m (Fixed1)} \\ + 2 &\text{ m (Spring S2)} \\ + 3/2 &\text{ m (half of the length of SlidingMass m3)} \\ \hline &4,5 \text{ m} = s(\text{start} = 4.5) \text{ for m3} \\ + 3/2 &\text{ m (half of the length of SlidingMass m3)} \\ + 4 &\text{ m (SpringDamper 4)} \\ + 5/2 &\text{ m (half of length of SlidingMass m5)} \\ \hline &12,5 \text{ m} = s(\text{start} = 12.5) \text{ for m5} \end{aligned}$$

This selection of initial conditions has the effect that Dymola selects those variables (m3.s and m5.s) as state variables. In the second example the length of the springs are given as start values but they cannot be used as state for pure springs (only for the spring/damper combination). In this case the system is not at rest.



**Modelica.Mechanics.Translational.Examples.WhyArrows****Use of arrows in Mechanics.Translational****Information**

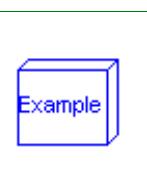
When using the models of the translational sublibrary it is recommended to make sure that all arrows point in the same direction because then all component have the same reference system. In the example the distance from flange\_a of Rod1 to flange\_b of Rod2 is 2 m. The distance from flange\_a of Rad1 to flange\_b of Rod3 is also 2 m though it is difficult to see that. Without the arrows it would be almost impossible to notice. That all arrows point in the same direction is a sufficient condition for an easy use of the library. There are cases where horizontally flipped models can be used without problems.

**Modelica.Mechanics.Translational.Examples.Accelerate****Use of model accelerate.****Information****Modelica.Mechanics.Translational.Examples.Damper****Use of damper models.****Information****Modelica.Mechanics.Translational.Examples.Oscillator****Oscillator demonstrates the use of initial conditions.****Information**

A spring - mass system is a mechanical oscillator. If no damping is included and the system is excited at resonance frequency infinite amplitudes will result. The resonant frequency is given by  $\omega_{res} = \sqrt{c/m}$  with:

```
c spring stiffness
m mass
```

To make sure that the system is initially at rest the initial conditions  $s(start=0)$  and  $v(start=0)$  for the SlidingMass are set. If damping is added the amplitudes are bounded.

**Modelica.Mechanics.Translational.Examples.Sensors****Sensors for translational systems.**

## Information

These sensors measure

force f in N  
position s in m  
velocity v in m/s  
acceleration a in m/s<sup>2</sup>

The measured velocity and acceleration is independent on the flange the sensor is connected to. The position depends on the flange (flange\_a or flange\_b) and the length L of the component. Plot PositionSensor1.s, PositionSensor2.s and SlidingMass1.s to see the difference.

*Error:Found no end-tag in HTML-documentation*

---

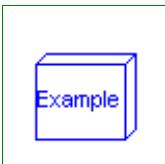


## Modelica.Mechanics.Translational.Examples.Friction

### Use of model Stop

## Information

1. Simulate and then plot Stop1.f as a function of Stop1.v. This gives the Stribeck curve.
  2. This model gives an example for a hard stop. However there can arise some problems with the used modeling approach (use of Reinit, convergence problems). In this case use the ElastoGap to model a stop (see example Preload).
- 

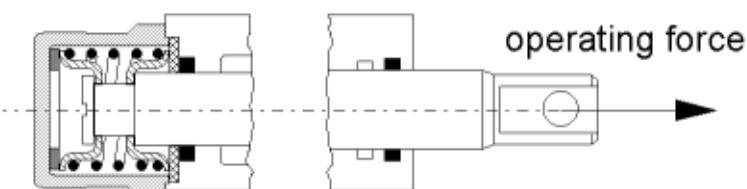


## Modelica.Mechanics.Translational.Examples.PreLoad

Preload of a spool using ElastoGap models.

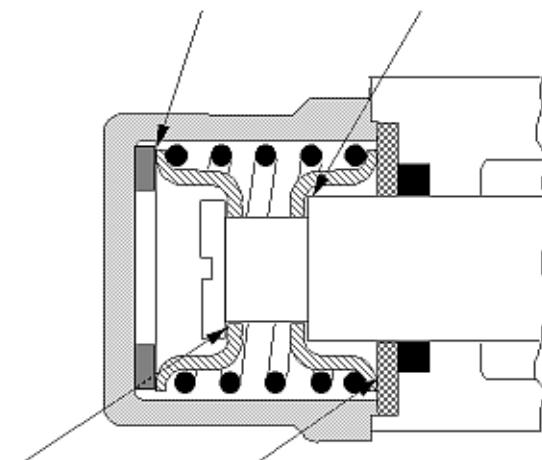
## Information

When designing hydraulic valves it is often necessary to hold the spool in a certain position as long as an external force is below a threshold value. If this force exceeds the threshold value a linear relation between force and position is desired. There are designs that need only one spring to accomplish this task. Using the ElastoGap elements this design can be modelled easily. Drawing of spool.

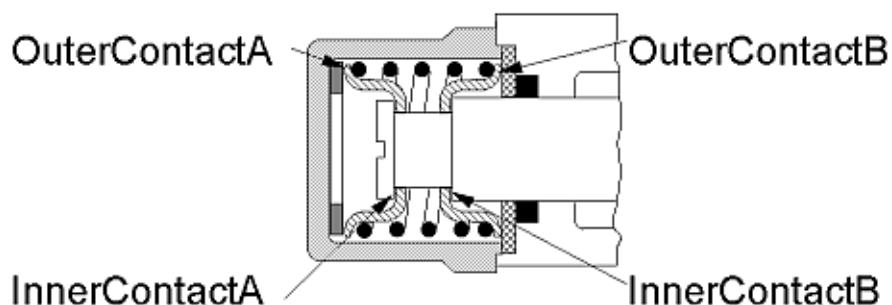


<

gap, if spool moved to the right

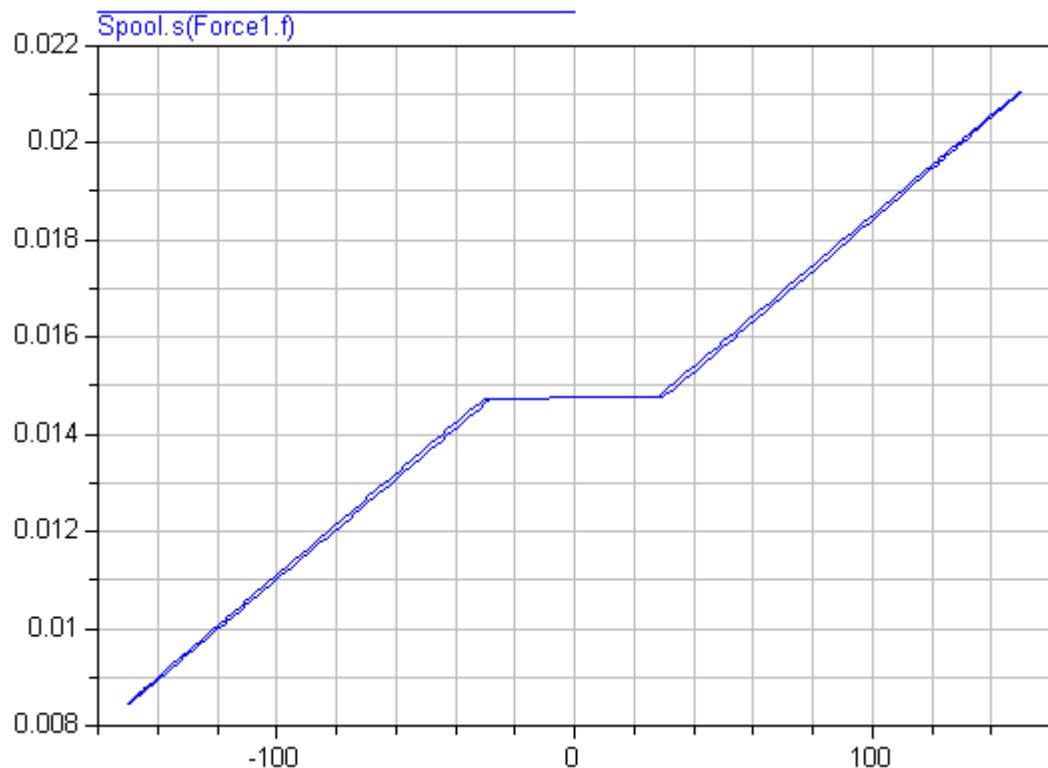


contact, if spool moved to the right



contact, if spool moved to the right

Spool position s as a function of working force f.



## Modelica.Mechanics.Translational.Sensors

Sensors for 1-dim. translational mechanical quantities

### Information

### Package Content

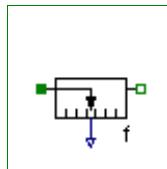
Name	Description
ForceSensor	Ideal sensor to measure the force between two flanges
PositionSensor	Ideal sensor to measure the absolute position
SpeedSensor	Ideal sensor to measure the absolute velocity
AccSensor	Ideal sensor to measure the absolute acceleration

### Modelica.Mechanics.Translational.Sensors.ForceSensor

Ideal sensor to measure the force between two flanges

### Information

Measures the *cut-force between two flanges* in an ideal way and provides the result as output signal (to be further processed with blocks of the Modelica.Blocks library).

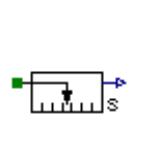


## Connectors

Name	Description
flange_a	(left) driving flange (flange axis directed INTO cut plane, e. g. from left to right)
flange_b	(right) driven flange (flange axis directed OUT OF cut plane)
f	force in flange_a and flange_b ( $f = \text{flange\_a}.f = -\text{flange\_b}.f$ )

## Modelica.Mechanics.Translational.Sensors.PositionSensor

Ideal sensor to measure the absolute position



## Information

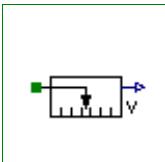
Measures the *absolute position*  $s$  of a flange in an ideal way and provides the result as output signals (to be further processed with blocks of the Modelica.Blocks library).

## Connectors

Name	Description
flange_a	flange to be measured (flange axis directed INTO cut plane, e. g. from left to right)
s	Absolute position of flange as output signal

## Modelica.Mechanics.Translational.Sensors.SpeedSensor

Ideal sensor to measure the absolute velocity



## Information

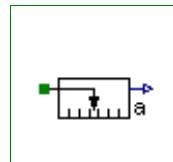
Measures the *absolute velocity*  $v$  of a flange in an ideal way and provides the result as output signals (to be further processed with blocks of the Modelica.Blocks library).

## Connectors

Name	Description
flange_a	flange to be measured (flange axis directed INTO cut plane, e. g. from left to right)
v	Absolute velocity of flange as output signal

## Modelica.Mechanics.Translational.Sensors.AccSensor

Ideal sensor to measure the absolute acceleration



## Information

Measures the *absolute acceleration*  $a$  of a flange in an ideal way and provides the result as output signals (to be further processed with blocks of the Modelica.Blocks library).

## Connectors

Name	Description
flange_a	flange to be measured (flange axis directed INTO cut plane, e. g. from left to right)

a	Absolute acceleration of flange as output signal
---	--

## Modelica.Mechanics.Translational.Interfaces

Interfaces for 1-dim. translational mechanical components

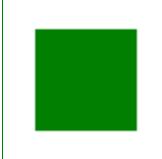
### Information

### Package Content

Name	Description
 Flange_a	(left) 1D translational flange (flange axis directed INTO cut plane, e. g. from left to right)
 Flange_b	right 1D translational flange (flange axis directed OUT OF cut plane)
 Rigid	Rigid connection of two translational 1D flanges
 Compliant	Compliant connection of two translational 1D flanges
 TwoFlanges	Component with two translational 1D flanges
 AbsoluteSensor	Device to measure a single absolute flange variable
 RelativeSensor	Device to measure a single relative variable between two flanges
 FrictionBase	Base class of Coulomb friction elements

## Modelica.Mechanics.Translational.Interfaces.Flange\_a

(left) 1D translational flange (flange axis directed INTO cut plane, e. g. from left to right)



### Information

This is a flange for 1D translational mechanical systems. In the cut plane of the flange a unit vector  $n$ , called flange axis, is defined which is directed INTO the cut plane, i. e. from left to right. All vectors in the cut plane are resolved with respect to this unit vector. E.g. force  $f$  characterizes a vector which is directed in the direction of  $n$  with value equal to  $f$ . When this flange is connected to other 1D translational flanges, this means that the axes vectors of the connected flanges are identical.

The following variables are transported through this connector:

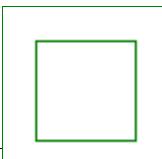
- s: Absolute position of the flange in [m]. A positive translation means that the flange is translated along the flange axis.
- f: Cut-force in direction of the flange axis in [N].

### Contents

Name	Description
s	absolute position of flange [m]
f	cut force directed into flange [N]

## Modelica.Mechanics.Translational.Interfaces.Flange\_b

right 1D translational flange (flange axis directed OUT OF cut plane)



## Information

This is a flange for 1D translational mechanical systems. In the cut plane of the flange a unit vector  $n$ , called flange axis, is defined which is directed OUT OF the cut plane. All vectors in the cut plane are resolved with respect to this unit vector. E.g. force  $f$  characterizes a vector which is directed in the direction of  $n$  with value equal to  $f$ . When this flange is connected to other 1D translational flanges, this means that the axes vectors of the connected flanges are identical.

The following variables are transported through this connector:

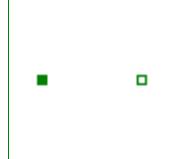
- $s$ : Absolute position of the flange in [m]. A positive translation means that the flange is translated along the flange axis.
- $f$ : Cut-force in direction of the flange axis in [N].

## Contents

Name	Description
$s$	absolute position of flange [m]
$f$	cut force directed into flange [N]

## Modelica.Mechanics.Translational.Interfaces.Rigid

### Rigid connection of two translational 1D flanges



## Information

This is a 1D translational component with two *rigidly* connected flanges. The distance between the left and the right flange is always constant, i. e.  $L$ . The forces at the right and left flange can be different. It is used e.g. to built up sliding masses.

## Parameters

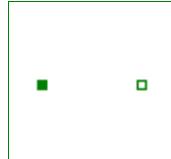
Name	Default	Description
$L$	0	length of component from left flange to right flange (= flange_b.s - flange_a.s) [m]

## Connectors

Name	Description
flange_a	(left) driving flange (flange axis directed INTO cut plane, i. e. from left to right)
flange_b	(right) driven flange (flange axis directed OUT OF cut plane, i. e. from right to left)

## Modelica.Mechanics.Translational.Interfaces.Compliant

### Compliant connection of two translational 1D flanges



## Information

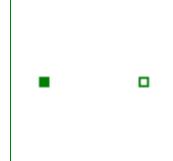
This is a 1D translational component with a *compliant* connection of two translational 1D flanges where inertial effects between the two flanges are not included. The absolute value of the force at the left and the right flange is the same. It is used to built up springs, dampers etc.

## Connectors

Name	Description
flange_a	(left) driving flange (flange axis directed INTO cut plane, e. g. from left to right)
flange_b	(right) driven flange (flange axis directed OUT OF cut plane)

## Modelica.Mechanics.Translational.Interfaces.TwoFlanges

Component with two translational 1D flanges



## Information

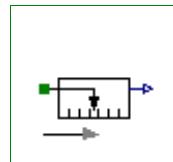
This is a 1D translational component with two flanges. It is used e.g. to built up parts of a drive train consisting of several base components.

## Connectors

Name	Description
flange_a	(left) driving flange (flange axis directed INTO cut plane, e. g. from left to right)
flange_b	(right) driven flange (flange axis directed OUT OF cut plane)

## Modelica.Mechanics.Translational.Interfaces.AbsoluteSensor

Device to measure a single absolute flange variable



## Information

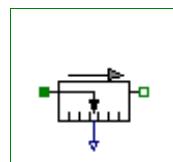
This is the superclass of a 1D translational component with one flange and one output signal in order to measure an absolute kinematic quantity in the flange and to provide the measured signal as output signal for further processing with the Modelica.Blocks blocks.

## Connectors

Name	Description
flange_a	flange to be measured (flange axis directed INTO cut plane, e. g. from left to right)
y	

## Modelica.Mechanics.Translational.Interfaces.RelativeSensor

Device to measure a single relative variable between two flanges



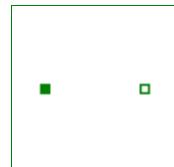
## Information

This is a superclass for 1D translational components with two rigidly connected flanges and one output signal in order to measure relative kinematic quantities between the two flanges or the cut-force in the flange and to provide the measured signal as output signal for further processing with the Modelica.Blocks blocks.

## Connectors

Name	Description
flange_a	(left) driving flange (flange axis directed INTO cut plane, e. g. from left to right)

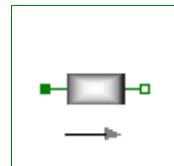
flange_b	(right) driven flange (flange axis directed OUT OF cut plane)
y	

**Modelica.Mechanics.Translational.Interfaces.FrictionBase****Base class of Coulomb friction elements****Information****Parameters**

Name	Default	Description
L	0	length of component from left flange to right flange (= flange_b.s - flange_a.s) [m]
smax	25	right stop for (right end of) sliding mass [m]
smin	-25	left stop for (left end of) sliding mass [m]
v_small	1e-3	Relative velocity near to zero (see model info text) [m/s]

**Connectors**

Name	Description
flange_a	(left) driving flange (flange axis directed INTO cut plane, i. e. from left to right)
flange_b	(right) driven flange (flange axis directed OUT OF cut plane, i. e. from right to left)

**Modelica.Mechanics.Translational.SlidingMass****Sliding mass with inertia****Information**Sliding mass with *inertia*, *without friction* and two rigidly connected flanges.

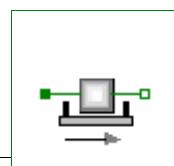
The sliding mass has the length L, the position coordinate s is in the middle. Sign convention: A positive force at flange flange\_a moves the sliding mass in the positive direction. A negative force at flange flange\_a moves the sliding mass to the negative direction.

**Parameters**

Name	Default	Description
L	0	length of component from left flange to right flange (= flange_b.s - flange_a.s) [m]
m	1	mass of the sliding mass [kg]

**Connectors**

Name	Description
flange_a	(left) driving flange (flange axis directed INTO cut plane, i. e. from left to right)
flange_b	(right) driven flange (flange axis directed OUT OF cut plane, i. e. from right to left)

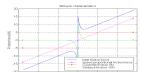
**Modelica.Mechanics.Translational.Stop****Sliding mass with hard stop and Stribeck friction**

## Information

This element describes the *Stribeck friction characteristics* of a sliding mass, i. e. the frictional force acting between the sliding mass and the support. Included is a *hard stop* for the position.

The surface is fixed and there is friction between sliding mass and surface. The frictional force  $f$  is given for positive velocity  $v$  by:

$$f = F_{\text{Coulomb}} + F_{\text{prop}} * v + F_{\text{Stribeck}} * \exp(-f_{\text{exp}} * v)$$



The distance between the left and the right connector is given by parameter L. The position of the center of gravity, coordinate s, is in the middle between the two flanges.

There are hard stops at smax and smin, i. e. if

$$\begin{aligned} \text{flange\_a.s} &\geq s_{\min} \\ \text{and} \\ \text{flange\_b.s} &\leq s_{\max} \end{aligned}$$

the sliding mass can move freely.

When the absolute velocity becomes zero, the sliding mass becomes stuck, i.e., the absolute position remains constant. In this phase the friction force is calculated from a force balance due to the requirement that the absolute acceleration shall be zero. The elements begin to slide when the friction force exceeds a threshold value, called the maximum static friction force, computed via:

$$\text{maximum\_static\_friction} = F_{\text{Coulomb}} + F_{\text{Stribeck}}$$

**This requires the states Stop.s and Stop.v**. If these states are eliminated during the index reduction the model will not work. To avoid this any inertias should be connected via springs to the Stop element, other sliding masses, dampers or hydraulic chambers must be avoided.

For more details of the used friction model see the following reference:

Beater P. (1999):

[Entwurf hydraulischer Maschinen](#). Springer Verlag Berlin Heidelberg New York.

The friction model is implemented in a "clean" way by state events and leads to continuous/discrete systems of equations which have to be solved by appropriate numerical methods. The method is described in:

Otter M., Elmquist H., and Mattsson S.E. (1999):

[Hybrid Modeling in Modelica based on the Synchronous Data Flow Principle](#). CACSD'99, Aug. 22.-26, Hawaii.

More precise friction models take into account the elasticity of the material when the two elements are "stuck", as well as other effects, like hysteresis. This has the advantage that the friction element can be completely described by a differential equation without events. The drawback is that the system becomes stiff (about 10-20 times slower simulation) and that more material constants have to be supplied which requires more sophisticated identification. For more details, see the following references, especially (Armstrong and Canudas de Witt 1996):

Armstrong B. (1991):

[Control of Machines with Friction](#). Kluwer Academic Press, Boston MA.

Armstrong B., and Canudas de Wit C. (1996):

[Friction Modeling and Compensation](#). The Control Handbook, edited by W.S.Levine, CRC Press, pp. 1369-1382.

Canudas de Wit C., Olsson H., Astrom K.J., and Lischinsky P. (1995):

[A new model for control of systems with friction](#). IEEE Transactions on Automatic Control, Vol. 40, No.

3, pp. 419-425.

## Parameters

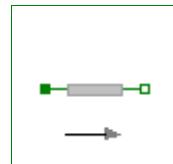
Name	Default	Description
L	0	length of component from left flange to right flange (= flange_b.s - flange_a.s) [m]
smax	25	right stop for (right end of) sliding mass [m]
smin	-25	left stop for (left end of) sliding mass [m]
v_small	1e-3	Relative velocity near to zero (see model info text) [m/s]
m	1	mass [kg]
F_prop	1	velocity dependent friction [N/ (m/s)]
F_Coulomb	5	constant friction: Coulomb force [N]
F_Stribeck	10	Stribeck effect [N]
fexp	2	exponential decay [1/ (m/s)]
Initialization		
s.start		absolute position of center of component ( $s = flange\_a.s + L/2 = flange\_b.s - L/2$ ) [m]
startForward.start	false	true, if $v_{rel}=0$ and start of forward sliding or $v_{rel} > v_{small}$
startBackward.start	false	true, if $v_{rel}=0$ and start of backward sliding or $v_{rel} < -v_{small}$
locked.start	false	true, if $v_{rel}=0$ and not sliding

## Connectors

Name	Description
flange_a	(left) driving flange (flange axis directed INTO cut plane, i. e. from left to right)
flange_b	(right) driven flange (flange axis directed OUT OF cut plane, i. e. from right to left)

## Modelica.Mechanics.Translational.Rod

Rod without inertia



## Information

Rod *without inertia* and two rigidly connected flanges.

## Parameters

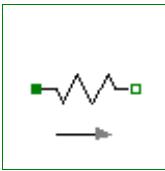
Name	Default	Description
L	0	length of component from left flange to right flange (= flange_b.s - flange_a.s) [m]

## Connectors

Name	Description
flange_a	(left) driving flange (flange axis directed INTO cut plane, i. e. from left to right)
flange_b	(right) driven flange (flange axis directed OUT OF cut plane, i. e. from right to left)

## Modelica.Mechanics.Translational.Spring

Linear 1D translational spring



### Information

A *linear 1D translational spring*. The component can be connected either between two sliding masses, or between a sliding mass and the housing (model Fixed), to describe a coupling of the sliding mass with the housing via a spring.

### Parameters

Name	Default	Description
s_rel0	0	unstretched spring length [m]
c	1	spring constant [N/m]

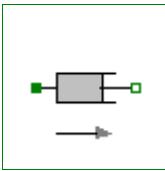
### Connectors

Name	Description
flange_a	(left) driving flange (flange axis directed INTO cut plane, e. g. from left to right)
flange_b	(right) driven flange (flange axis directed OUT OF cut plane)

---

## Modelica.Mechanics.Translational.Damper

Linear 1D translational damper



### Information

*Linear, velocity dependent damper* element. It can be either connected between a sliding mass and the housing (model Fixed), or between two sliding masses.

### Parameters

Name	Default	Description
d	0	damping constant [N/ (m/s)] [N/ (m/s)]

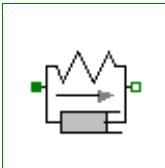
### Connectors

Name	Description
flange_a	(left) driving flange (flange axis directed INTO cut plane, e. g. from left to right)
flange_b	(right) driven flange (flange axis directed OUT OF cut plane)

---

## Modelica.Mechanics.Translational.SpringDamper

Linear 1D translational spring and damper in parallel



### Information

A *spring and damper element connected in parallel*. The component can be connected either between two sliding masses to describe the elasticity and damping, or between a sliding mass and the housing (model Fixed), to describe a coupling of the sliding mass with the housing via a spring/damper.

## Parameters

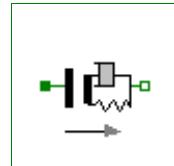
Name	Default	Description
s_rel0	0	unstretched spring length [m]
c	1	spring constant [N/m]
d	1	damping constant [N/(m/s)]

## Connectors

Name	Description
flange_a	(left) driving flange (flange axis directed INTO cut plane, e. g. from left to right)
flange_b	(right) driven flange (flange axis directed OUT OF cut plane)

## Modelica.Mechanics.Translational.ElastoGap

1D translational spring damper combination with gap



## Information

A *linear translational spring damper combination that can lift off*. The component can be connected between a sliding mass and the housing (model Fixed), to describe the contact of a sliding mass with the housing.

## Parameters

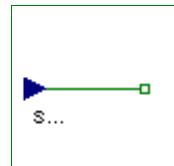
Name	Default	Description
s_rel0	0	unstretched spring length [m]
c	1	spring constant [N/m]
d	1	damping constant [N/ (m/s)]

## Connectors

Name	Description
flange_a	(left) driving flange (flange axis directed INTO cut plane, e. g. from left to right)
flange_b	(right) driven flange (flange axis directed OUT OF cut plane)

## Modelica.Mechanics.Translational.Position

Forced movement of a flange according to a reference position



## Information

The input signal **s\_ref** defines the **reference position** in [m]. Flange **flange\_b** is **forced** to move according to this reference motion. According to parameter **exact** (default = **false**), this is done in the following way:

### 1. **exact=true**

The reference position is treated **exactly**. This is only possible, if the input signal is defined by an analytical function which can be differentiated at least twice. If this prerequisite is fulfilled, the Modelica translator will differentiate the input signal twice in order to compute the reference acceleration of the flange.

### 2. **exact=false**

The reference position is **filtered** and the second derivative of the filtered curve is used to compute the reference acceleration of the flange. This second derivative is **not** computed by numerical differentiation but by an appropriate realization of the filter. For filtering, a second order Bessel filter

## 642 Modelica.Mechanics.Translational.Position

is used. The critical frequency (also called cut-off frequency) of the filter is defined via parameter **f\_crit** in [Hz]. This value should be selected in such a way that it is higher as the essential low frequencies in the signal.

The input signal can be provided from one of the signal generator blocks of the block library Modelica.Blocks.Sources.

### Parameters

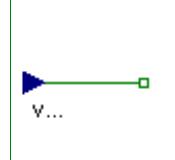
Name	Default	Description
exact	false	true/false exact treatment/filtering the input signal
f_crit	50	if exact=false, critical frequency of filter to filter input signal [Hz]

### Connectors

Name	Description
flange_b	
s_ref	reference position of flange as input signal

## Modelica.Mechanics.Translational.Speed

Forced movement of a flange according to a reference speed



### Information

The input signal **v\_ref** defines the **reference speed** in [m/s]. Flange **flange\_b** is **forced** to move according to this reference motion. According to parameter **exact** (default = **false**), this is done in the following way:

#### 1. **exact=true**

The reference speed is treated **exactly**. This is only possible, if the input signal is defined by an analytical function which can be differentiated at least once. If this prerequisite is fulfilled, the Modelica translator will differentiate the input signal once in order to compute the reference acceleration of the flange.

#### 2. **exact=false**

The reference speed is **filtered** and the first derivative of the filtered curve is used to compute the reference acceleration of the flange. This first derivative is **not** computed by numerical differentiation but by an appropriate realization of the filter. For filtering, a first order filter is used. The critical frequency (also called cut-off frequency) of the filter is defined via parameter **f\_crit** in [Hz]. This value should be selected in such a way that it is higher as the essential low frequencies in the signal.

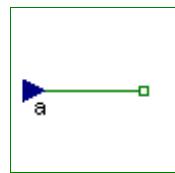
The input signal can be provided from one of the signal generator blocks of the block library Modelica.Blocks.Sources.

### Parameters

Name	Default	Description
exact	false	true/false exact treatment/filtering the input signal
f_crit	50	if exact=false, critical frequency of filter to filter input signal [Hz]
s_start	0	Start position of flange_b [m]

### Connectors

Name	Description
v_ref	reference speed of flange as input signal
flange_b	Flange that is forced to move according to input signals u

**Modelica.Mechanics.Translational.Accelerate****Forced movement of a flange according to an acceleration signal****Information**

The input signal **a** in [m/s<sup>2</sup>] moves the 1D translational flange connector **flange\_b** with a predefined *acceleration*, i.e., the flange is *forced* to move with this acceleration. The velocity and the position of the flange are also predefined and are determined by integration of the acceleration.

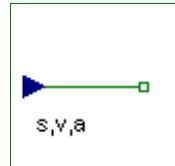
The acceleration "a(t)" can be provided from one of the signal generator blocks of the block library Modelica.Blocks.Source.

**Parameters**

Name	Default	Description
s_start	0	Start position [m]
v_start	0	Start velocity [m/s]

**Connectors**

Name	Description
a	absolute acceleration of flange as input signal
flange_b	

**Modelica.Mechanics.Translational.Move****Forced movement of a flange according to a position, velocity and acceleration signal****Information**

Flange **flange\_b** is **forced** to move with a predefined motion according to the input signals:

- u[1]: position of flange
- u[2]: velocity of flange
- u[3]: acceleration of flange

The user has to guarantee that the input signals are consistent to each other, i.e., that u[2] is the derivative of u[1] and that u[3] is the derivative of u. There are, however, also applications where by purpose these conditions do not hold. For example, if only the position dependent terms of a mechanical system shall be calculated, one may provide position = position(t) and set the velocity and the acceleration to zero.

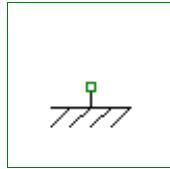
The input signals can be provided from one of the signal generator blocks of the block library Modelica.Blocks.Sources.

**Connectors**

Name	Description
u[3]	position, velocity and acceleration of flange as input signals
flange_b	Flange that is forced to move according to input signals u

## Modelica.Mechanics.Translational.Fixed

### Fixed flange



### Information

The *flange* of a 1D translational mechanical system *fixed* at an position *s0* in the *housing*. May be used:

- to connect a compliant element, such as a spring or a damper, between a sliding mass and the housing.
- to fix a rigid element, such as a sliding mass, at a specific position.

### Parameters

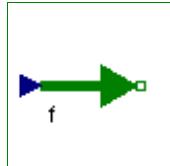
Name	Default	Description
s0	0	fixed offset position of housing [m]

### Connectors

Name	Description
flange_b	

## Modelica.Mechanics.Translational.Force

External force acting on a drive train element as input signal



### Information

The input signal "s" in [N] characterizes an *external force* which acts (with positive sign) at a flange, i.e., the component connected to the flange is driven by force f.

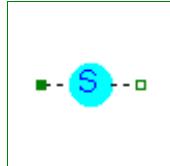
Input signal s can be provided from one of the signal generator blocks of Modelica.Blocks.Source.

### Connectors

Name	Description
flange_b	
f	driving force as input signal

## Modelica.Mechanics.Translational.RelativeStates

Definition of relative state variables



### Information

Usually, the absolute position and the absolute velocity of Modelica.Mechanics.Translational.Inertia models are used as state variables. In some circumstances, relative quantities are better suited, e.g., because it may be easier to supply initial values. In such cases, model **RelativeStates** allows the definition of state variables in the following way:

- Connect an instance of this model between two flange connectors.
- The **relative position** and the **relative velocity** between the two connectors are used as **state variables**.

An example is given in the next figure



Here, the relative position and the relative velocity between the two masses are used as state variables. Additionally, the simulator selects either the absolute position and absolute velocity of model mass1 or of model mass2 as state variables.

## Connectors

Name	Description
flange_a	(left) driving flange (flange axis directed INTO cut plane, e. g. from left to right)
flange_b	(right) driven flange (flange axis directed OUT OF cut plane)

## Modelica.Media

### Library of media property models

#### Information

This library contains [interface](#) definitions for media and the following [property](#) models for single and multiple substance fluids with one and multiple phases:

- [Ideal gases](#):  
1241 high precision gas models based on the NASA Glenn coefficients, plus ideal gas mixture models based on the same data.
- [Water models](#):  
ConstantPropertyLiquidWater, WaterIF97 (high precision water model according to the IAPWS/IF97 standard)
- [Air models](#):  
SimpleAir, DryAirNasa, and MoistAir
- [Incompressible media](#):  
TableBased (template medium for user defined properties from tables rho(T), HeatCapacity\_cp(T), etc.)

The following parts are useful, when newly starting with this library:

- [Modelica.Media.UsersGuide](#).
- [Modelica.Media.UsersGuide.MediumUsage](#) describes how to use a medium model in a component model.
- [Modelica.Media.UsersGuide.MediumDefinition](#) describes how a new fluid medium model has to be implemented.
- [Modelica.Media.UsersGuide.ReleaseNotes](#) summarizes the changes of the library releases.
- [Modelica.Media.Examples](#) contains examples that demonstrate the usage of this library.

Copyright © 1998-2007, Modelica Association.

*This Modelica package is free software; it can be redistributed and/or modified under the terms of the [Modelica license](#), see the license conditions and the accompanying [disclaimer](#) [here](#).*

## Package Content

Name	Description
<a href="#">UsersGuide</a>	User's Guide of Media Library
<a href="#">Examples</a>	Demonstrate usage of property models (currently: simple tests)

<input type="checkbox"/> Interfaces	Interfaces for media models
<input type="checkbox"/> Common	data structures and fundamental functions for fluid properties
<input type="checkbox"/> Air	Medium models for air
<input type="checkbox"/> CompressibleLiquids	compressible liquid models
<input type="checkbox"/> IdealGases	Data and models of ideal gases (single, fixed and dynamic mixtures) from NASA source
<input type="checkbox"/> Incompressible	Medium model for T-dependent properties, defined by tables or polynomials
<input type="checkbox"/> Water	Medium models for water

## Modelica.Media.UsersGuide

Library **Modelica.Media** is a **free** Modelica package providing a standardized interface to fluid media models and specific media models based on this interface. A fluid medium model defines **algebraic** equations for the intensive thermodynamic variables used in the **mass** and **energy** balance of component models. Optionally, additional medium properties can be computed such as dynamic viscosity or thermal conductivity. Medium models are defined for **single** and **multiple substance** fluids with **one** and **multiple phases**.



A large part of the library provides specific medium models that can be directly utilized. This library can be used in all types of Modelica fluid libraries that may have different connectors and design philosophies. It is particularly utilized in the Modelica\_Fluid library (the Modelica\_Fluid library is currently under development to provide 1D thermo-fluid flow components for single and multiple substance flow with one and multiple phases). The Modelica.Media library has the following main features:

- Balance equations and media model equations are decoupled. This means that the used medium model does usually not have an influence on how the balance equations are formulated. For example, the same balance equations are used for media that use pressure and temperature, or pressure and specific enthalpy as independent variables, as well as for incompressible and compressible media models. A Modelica tool will have enough information to generate as efficient code as a traditional (coupled) definition. This feature is described in more detail in section [Static State Selection](#).
- Optional variables, such as dynamic viscosity, are only computed if needed in the corresponding component.
- The independent variables of a medium model do not influence the definition of a fluid connector port. Especially, the media models are implemented in such a way that a connector may have the minimum number of independent medium variables in a connector and still get the same efficiency as if all medium variables are passed by the connector from one component to the next one (the latter approach has the restriction that a fluid port can only connect two components and not more). Note, the Modelica\_Fluid library uses the first approach, i.e., having a set of independent medium variables in a connector.
- The medium models are implemented with regards to efficient dynamic simulation. For example, two phase medium models trigger state events at phase boundaries (because the medium variables are not differentiable at this point).

This User's Guide has the following main parts:

- [Medium usage](#) describes how to use a medium model from this library in a component model.
- [Medium definition](#) describes how a new fluid medium model has to be implemented.
- [ReleaseNotes](#) summarizes the changes of the library releases.
- [Contact](#) provides information about the authors of the library as well as acknowledgements.

## Modelica.Media.Examples

Demonstrate usage of property models (currently: simple tests)

### Information

#### Examples

Physical properties for fluids are needed in so many different variants that a library can only provide models for the most common situations. With the following examples we are going to demonstrate how to use the existing packages and functions in Modelica.Media to customize these models for advanced applications. The high level functions try to abstract as much as possible from the fact that different media are based on different variables, e.g. ideal gases need pressure and temperature, while many refrigerants are based on Helmholtz functions of density and temperature, and many water properties are based on pressure and specific enthalpy. Medium properties are needed in control volumes in the dynamic state equations and in many thermodynamic state locations that are independent of the dynamic states of a control volume, e.g. at a wall temperature, an isentropic reference state or at a phase boundary. The general structure of the library is such that:

- Each medium has a model called BaseProperties. BaseProperties contains the minimum set of medium properties needed in a dynamic control volume model.
- Each instance of BaseProperties contains a "state" record that is an input to all the functions to compute properties. If these functions need further inputs, like e.g. the molarMass, these are accessible as constants in the package.
- The simplest way to compute properties at any other reference point is to declare an instance of ThermodynamicState and use that as input to arbitrary property functions.

A small library of generic volume, pipe, pump and ambient models is provided in Modelica.Media.Examples.Tests.Components to demonstrate how fluid components should be implemented that are using Modelica.Media models. This library is also used to test all media models in Modelica.Media.Examples.Tests.MediaTestModels.

### Package Content

Name	Description
 SimpleLiquidWater	Example for Water.SimpleLiquidWater medium model
 IdealGasH2O	IdealGas H2O medium model
 WaterIF97	WaterIF97 medium model
 MixtureGases	Test gas mixtures
 MoistAir	Ideal gas flue gas model
 TwoPhaseWater	extension of the StandardWater package
 TestOnly	examples for testing purposes: move for final version
 Tests	Library to test that all media models simulate and fulfill the expected structural properties
 SolveOneNonlinearEquation	Demonstrate how to solve one non-linear algebraic equation in one unknown

### Modelica.Media.Examples.SimpleLiquidWater

Example for Water.SimpleLiquidWater medium model



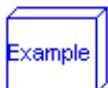
## Information

### Parameters

Name	Default	Description
V	1	Volume [m3]
H_flow_ext	1.e6	Constant enthalpy flow rate into the volume [W]

---

## Modelica.Media.Examples.IdealGasH2O



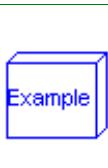
IdealGas H2O medium model

## Information

An example for using ideal gas properties and how to compute isentropic enthalpy changes. The function that is implemented is approximate, but usually very good: the second medium record medium2 is given to compare the approximation.

---

## Modelica.Media.Examples.WaterIF97



WaterIF97 medium model

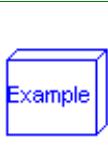
## Information

### Parameters

Name	Default	Description
dV	0.0	[m3/s]
m_flow_ext	0	[kg/s]
H_flow_ext	10000	[W]

---

## Modelica.Media.Examples.MixtureGases



Test gas mixtures

## Information

### Parameters

Name	Default	Description
V	1	
m_flow_ext	0.01	
H_flow_ext	5000	

---

## Modelica.Media.Examples.MoistAir



Ideal gas flue gas model

## Information

An example for using ideal gas properties and how to compute isentropic enthalpy changes. The function that is implemented is approximate, but usually very good: the second medium record medium2 is given to compare the approximation.

## Parameters

Name	Default	Description
MMx[2]	{Medium.dryair.MM,Medium.ste...}	

## Modelica.Media.Examples.TwoPhaseWater

extension of the StandardWater package

## Information

### Example: TwoPhaseWater

The TwoPhaseWater package demonstrates how to extend the parsimonious BaseProperties with a minimal set of properties from the standard water package with most properties that are needed in two-phase situations. The model also demonstrates how to compute additional properties for the medium model. In this scenario, that builds a new medium model with many more properties than the default, the standard BaseProperties is used as a basis. For additional properties, a user has to:

1. Declare a new variable of the wanted type, e.g. "DynamicViscosity eta".
2. Compute that variable by calling the function form the package, e.g. `eta = dynamicViscosity(state)`. Note that the instance of ThermodynamicState is used as an input to the function. This instance "state" is declared in PartialMedium and thus available in every medium model. A user does not have to know what actual variables are required to compute the dynamic viscosity, because the state instance is guaranteed to contain what is needed.
3. **Attention:** Many properties are not well defined in the two phase region and the functions might return undesired values if called there. It is the user's responsibility to take care of such situations. The example uses one of several possible models to compute an averaged viscosity for two-phase flows.

In two phase models, properties are often needed on the phase boundary just outside the two phase dome, right on the border.. To compute the thermodynamic state there, two auxiliary functions are provided: `setDewState(sat)` and `setBubbleState(sat)`. They take an instance of SaturationProperties as input. By default they are in one-phase, but with the optional phase argument set to 2, the output is forced to be just inside the phase boundary. This is only needed when derivatives like cv are computed with are different on both sides of the boundaries. The ususal steps to compute properties on the phase boundary are:

1. Declare an instance of ThermodynamicState, e.g. "ThermodynamicState dew".
2. Compute the state, using an instance of SaturationProperties, e.g. `dew = setDewState(sat)`
3. Compute properties on the phase boundary to your full desire, e.g. "`cp_d = specificHeatCapacityCp(dew)`".

The sample model TestTwoPhaseStates test the extended properties

The same procedure can be used to compute properties at other state points, e.g. when an isentropic reference state is computed.

## Package Content

Name	Description
<code>ExtendedProperties</code>	plenty of two-phase properties

TestTwoPhaseStates	Test the above model
<b>Inherited</b>	
 ThermodynamicState	thermodynamic state
ph_explicit	true if explicit in pressure and specific enthalpy
dT_explicit	true if explicit in density and temperature
pT_explicit	true if explicit in pressure and temperature
 BaseProperties	Base properties of water
 density_ph	Computes density as a function of pressure and specific enthalpy
 temperature_ph	Computes temperature as a function of pressure and specific enthalpy
 temperature_ps	Compute temperature from pressure and specific enthalpy
 density_ps	Computes density as a function of pressure and specific enthalpy
 pressure_dT	Computes pressure as a function of density and temperature
 specificEnthalpy_dT	Computes specific enthalpy as a function of density and temperature
 specificEnthalpy_pT	Computes specific enthalpy as a function of pressure and temperature
 specificEnthalpy_ps	Computes specific enthalpy as a function of pressure and temperature
 density_pT	Computes density as a function of pressure and temperature
 setDewState	set the thermodynamic state on the dew line
 setBubbleState	set the thermodynamic state on the bubble line
 dynamicViscosity	Dynamic viscosity of water
 thermalConductivity	Thermal conductivity of water
 surfaceTension	Surface tension in two phase region of water
 pressure	return pressure of ideal gas
 temperature	return temperature of ideal gas
 density	return density of ideal gas
 specificEnthalpy	Return specific enthalpy
 specificInternalEnergy	Return specific internal energy
 specificGibbsEnergy	Return specific Gibbs energy
 specificHelmholtzEnergy	Return specific Helmholtz energy
 specificEntropy	specific entropy of water
 specificHeatCapacityCp	specific heat capacity at constant pressure of water
 specificHeatCapacityCv	specific heat capacity at constant volume of water
 isentropicExponent	Return isentropic exponent
 isothermalCompressibility	Isothermal compressibility of water
 isobaricExpansionCoefficient	isobaric expansion coefficient of water
 velocityOfSound	

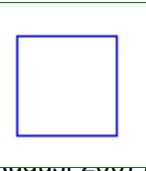
(f) isentropicEnthalpy	compute h(p,s)
(f) density_derh_p	density derivative by specific enthalpy
(f) density_derh_h	density derivative by pressure
(f) bubbleEnthalpy	boiling curve specific enthalpy of water
(f) dewEnthalpy	dew curve specific enthalpy of water
(f) bubbleEntropy	boiling curve specific entropy of water
(f) dewEntropy	dew curve specific entropy of water
(f) bubbleDensity	boiling curve specific density of water
(f) dewDensity	dew curve specific density of water
(f) saturationTemperature	saturation temperature of water
(f) saturationTemperature_derh	derivative of saturation temperature w.r.t. pressure
(f) saturationPressure	saturation pressure of water
(f) dBubbleDensity_dPressure	bubble point density derivative
(f) dDewDensity_dPressure	dew point density derivative
(f) dBubbleEnthalpy_dPressure	bubble point specific enthalpy derivative
(f) dDewEnthalpy_dPressure	dew point specific enthalpy derivative
(f) setState_dTX	
(f) setState_phX	
(f) setState_psX	
(f) setState_pTX	
smoothModel	true if the (derived) model should not generate state events
onePhase	true if the (derived) model should never be called with two-phase inputs
[FluidLimits]	validity limits for fluid model
[FluidConstants]	extended fluid constants
fluidConstants	constant data for the fluid
[SaturationProperties]	Saturation properties of two phase medium
FixedPhase	phase of the fluid: 1 for 1-phase, 2 for two-phase, 0 for not known, e.g. interactive use
(f) setSat_T	Return saturation property record from temperature
(f) setSat_p	Return saturation property record from pressure
(f) saturationPressure_sat	Return saturation temperature
(f) saturationTemperature_sat	Return saturation temperature
(f) saturationTemperature_derh_sat	Return derivative of saturation temperature w.r.t. pressure
(f) molarMass	Return the molar mass of the medium
(f) specificEnthalpy_pTX	Return specific enthalpy from pressure, temperature and mass fraction
(f) temperature_phX	Return temperature from p, h, and X or Xi
(f) density_phX	Return density from p, h, and X or Xi

<code>temperature_psX</code>	Return temperature from p, s, and X or Xi
<code>density_psX</code>	Return density from p, s, and X or Xi
<code>specificEnthalpy_psX</code>	Return specific enthalpy from p, s, and X or Xi
<code>setState_pT</code>	Return thermodynamic state from p and T
<code>setState_ph</code>	Return thermodynamic state from p and h
<code>setState_ps</code>	Return thermodynamic state from p and s
<code>setState_dT</code>	Return thermodynamic state from d and T
<code>setState_px</code>	Return thermodynamic state from pressure and vapour quality
<code>setState_Tx</code>	Return thermodynamic state from temperature and vapour quality
<code>vapourQuality</code>	Return vapour quality
<code>mediumName="unusablePartialMedium"</code>	Name of the medium
<code>substanceNames={mediumName}</code>	Names of the mixture substances. Set <code>substanceNames={mediumName}</code> if only one substance.
<code>extraPropertiesNames=fill("", 0)</code>	Names of the additional (extra) transported properties. Set <code>extraPropertiesNames=fill("",0)</code> if unused
<code>singleState</code>	= true, if u and d are not a function of pressure
<code>reducedX=true</code>	= true if medium contains the equation $\sum(X) = 1.0$ ; set <code>reducedX=true</code> if only one substance (see docu for details)
<code>fixedX=false</code>	= true if medium contains the equation $X = \text{reference\_}X$
<code>reference_p=101325</code>	Reference pressure of Medium: default 1 atmosphere
<code>reference_T=298.15</code>	Reference temperature of Medium: default 25 deg Celsius
<code>reference_X=if nX == 0 then fill(0, nX) else fill(1/nX, nX)</code>	Default mass fractions of medium
<code>p_default=101325</code>	Default value for pressure of medium (for initialization)
<code>T_default=Modelica.Slunits.Conversions.from_degC(20)</code>	Default value for temperature of medium (for initialization)
<code>h_default=specificEnthalpy_pTX(p_default, T_default, X_default)</code>	Default value for specific enthalpy of medium (for initialization)
<code>X_default=reference_X</code>	Default value for mass fractions of medium (for initialization)
<code>nS=size(substanceNames, 1)</code>	Number of substances
<code>nX=if nS == 1 then 0 else nS</code>	Number of mass fractions (= 0, if only one substance)
<code>nXi=if fixedX then 0 else if reducedX then nS - 1 else nX</code>	Number of structurally independent mass fractions (see docu for details)
<code>nC=size(extraPropertiesNames, 1)</code>	Number of extra (outside of standard mass-balance) transported properties
<code>BasePropertiesRecord</code>	Variables contained in every instance of BaseProperties
<code>prandtlNumber</code>	Return the Prandtl number
<code>heatCapacity_cp</code>	alias for deprecated name
<code>heatCapacity_cv</code>	alias for deprecated name
<code>beta</code>	alias for isobaricExpansionCoefficient for user convenience
<code>kappa</code>	alias of isothermalCompressibility for user convenience
<code>density_derP_T</code>	Return density derivative wrt pressure at const temperature

 density_derT_p	Return density derivative wrt temperature at constant pressure
 density_derX	Return density derivative wrt mass fraction
 density_pTX	Return density from p, T, and X or Xi
AbsolutePressure	Type for absolute pressure with medium specific attributes
Density	Type for density with medium specific attributes
DynamicViscosity	Type for dynamic viscosity with medium specific attributes
EnthalpyFlowRate	Type for enthalpy flow rate with medium specific attributes
MassFlowRate	Type for mass flow rate with medium specific attributes
MassFraction	Type for mass fraction with medium specific attributes
MoleFraction	Type for mole fraction with medium specific attributes
MolarMass	Type for molar mass with medium specific attributes
MolarVolume	Type for molar volume with medium specific attributes
IsentropicExponent	Type for isentropic exponent with medium specific attributes
SpecificEnergy	Type for specific energy with medium specific attributes
SpecificInternalEnergy	Type for specific internal energy with medium specific attributes
SpecificEnthalpy	Type for specific enthalpy with medium specific attributes
SpecificEntropy	Type for specific entropy with medium specific attributes
SpecificHeatCapacity	Type for specific heat capacity with medium specific attributes
SurfaceTension	Type for surface tension with medium specific attributes
Temperature	Type for temperature with medium specific attributes
ThermalConductivity	Type for thermal conductivity with medium specific attributes
PrandtlNumber	Type for Prandtl number with medium specific attributes
VelocityOfSound	Type for velocity of sound with medium specific attributes
ExtraProperty	Type for unspecified, mass-specific property transported by flow
CumulativeExtraProperty	Type for conserved integral of unspecified, mass specific property
ExtraPropertyFlowRate	Type for flow rate of unspecified, mass-specific property
IsobaricExpansionCoefficient	Type for isobaric expansion coefficient with medium specific attributes
DipoleMoment	Type for dipole moment with medium specific attributes
DerDensityByPressure	Type for partial derivative of density with respect to pressure with medium specific attributes
DerDensityByEnthalpy	Type for partial derivative of density with respect to enthalpy with medium specific attributes
DerEnthalpyByPressure	Type for partial derivative of enthalpy with respect to pressure with medium specific attributes
DerDensityByTemperature	Type for partial derivative of density with respect to temperature with medium specific attributes
 Choices	Types, constants to define menu choices

**Modelica.Media.Examples.TwoPhaseWater.ExtendedProperties**

plenty of two-phase properties



## Parameters

Name	Default	Description
standardOrderComponents	true	if true, last element in components is computed from 1-sum(Xi)
<b>Initialization</b>		
phase.start	1	2 for two-phase, 1 for one-phase, 0 if not known
<b>Advanced</b>		
preferredMediumStates	false	= true if StateSelect.prefer shall be used for the independent property variables of the medium

## Modelica.Media.Examples.TwoPhaseWater.TestTwoPhaseStates

Test the above model

## Information

For details see the documentation of the example package TwoPhaseWater

## Parameters

Name	Default	Description
dh	80000.0	80 kJ/second
dp	1.0e6	10 bars per second

## Modelica.Media.Examples.TestOnly

examples for testing purposes: move for final version

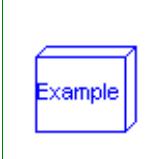
## Information

## Package Content

Name	Description
 MixIdealGasAir	Ideal gas air medium model
 FlueGas	Ideal gas flue gas model
 TestMedia	
 IdealGasAir	Test IdealGas.SingleMedia.Air medium model

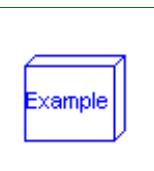
## Modelica.Media.Examples.TestOnly.MixIdealGasAir

Ideal gas air medium model



## Information

An example for using ideal gas properties and how to compute isentropic enthalpy changes. The function that is implemented is approximate, but usually very good: the second medium record medium2 is given to compare the approximation.

**Modelica.Media.Examples.TestOnly.FlueGas****Ideal gas flue gas model****Information**

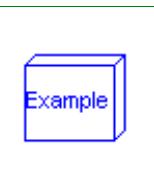
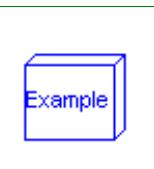
An example for using ideal gas properties and how to compute isentropic enthalpy changes. The function that is implemented is approximate, but usually very good: the second medium record medium2 is given to compare the approximation.

**Parameters**

Name	Default	Description
MMx[4]	Medium.data.MM	

**Modelica.Media.Examples.TestOnly.TestMedia****Information****Package Content**

Name	Description
<input type="checkbox"/> TemplateMedium	Test Interfaces.TemplateMedium

**Modelica.Media.Examples.TestOnly.TestMedia.TemplateMedium****Test Interfaces.TemplateMedium****Information****Modelica.Media.Examples.TestOnly.IdealGasAir****Test IdealGas.SingleMedia.Air medium model****Information****Parameters**

Name	Default	Description
V	1	
m_flow_ext	0.01	
H_flow_ext	5000	

**Modelica.Media.Examples.Tests****Library to test that all media models simulate and fulfill the expected structural properties**

## Information

### Package Content

Name	Description
 Components	Functions, connectors and models needed for the media model tests
 MediaTestModels	Test models to test all media

### Modelica.Media.Examples.Tests.Components

Functions, connectors and models needed for the media model tests

## Information

### Package Content

Name	Description
 FluidPort	Interface for quasi one-dimensional fluid flow in a piping network (incompressible or compressible, one or more phases, one or more substances)
 FluidPort_a	Fluid connector with filled icon
 FluidPort_b	Fluid connector with outlined icon
 PortVolume	Fixed volume associated with a port by the finite volume method
 FixedMassFlowRate	Ideal pump that produces a constant mass flow rate from a large reservoir at fixed temperature and mass fraction
 FixedAmbient	Ambient pressure, temperature and mass fraction source
 ShortPipe	Simple pressure loss in pipe
 PartialTestModel	Basic test model to test a medium
 PartialTestModel2	slightly larger test model to test a medium

### Modelica.Media.Examples.Tests.Components.FluidPort

Interface for quasi one-dimensional fluid flow in a piping network (incompressible or compressible, one or more phases, one or more substances)

## Information

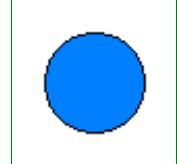
### Contents

Name	Description
p	Pressure in the connection point [Pa]
m_flow	Mass flow rate from the connection point into the component [kg/s]
h	Specific mixture enthalpy in the connection point [J/kg]
H_flow	Enthalpy flow rate into the component (if m_flow > 0, H_flow = m_flow*h) [W]
Xi[Medium.nXi]	Independent mixture mass fractions m_i/m in the connection point [kg/kg]
mXi_flow[Medium.nXi]	Mass flow rates of the independent substances from the connection point into the component (if m_flow > 0, mX_flow = m_flow*X) [kg/s]

C[Medium.nC]	properties $c_i/m$ in the connection point
mC_flow[Medium.nC]	Flow rates of auxiliary properties from the connection point into the component (if $m_{flow} > 0$ , $mC_{flow} = m_{flow} * C$ )

**Modelica.Media.Examples.Tests.Components.FluidPort\_a**

Fluid connector with filled icon

**Information**

Modelica.Media.Examples.Tests.Components.FluidPort\_a

**Parameters**

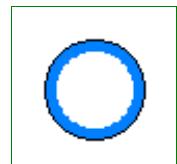
Name	Default	Description
replaceable package Medium	PartialMedium	Medium model

**Contents**

Name	Description
p	Pressure in the connection point [Pa]
m_flow	Mass flow rate from the connection point into the component [kg/s]
h	Specific mixture enthalpy in the connection point [J/kg]
H_flow	Enthalpy flow rate into the component (if $m_{flow} > 0$ , $H_{flow} = m_{flow} * h$ ) [W]
Xi[Medium.nXi]	Independent mixture mass fractions $m_i/m$ in the connection point [kg/kg]
mXi_flow[Medium.nXi]	Mass flow rates of the independent substances from the connection point into the component (if $m_{flow} > 0$ , $mX_{flow} = m_{flow} * X$ ) [kg/s]
C[Medium.nC]	properties $c_i/m$ in the connection point
mC_flow[Medium.nC]	Flow rates of auxiliary properties from the connection point into the component (if $m_{flow} > 0$ , $mC_{flow} = m_{flow} * C$ )

**Modelica.Media.Examples.Tests.Components.FluidPort\_b**

Fluid connector with outlined icon

**Information****Parameters**

Name	Default	Description
replaceable package Medium	PartialMedium	Medium model

**Contents**

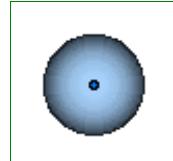
Name	Description
p	Pressure in the connection point [Pa]
m_flow	Mass flow rate from the connection point into the component [kg/s]
h	Specific mixture enthalpy in the connection point [J/kg]
H_flow	Enthalpy flow rate into the component (if $m_{flow} > 0$ , $H_{flow} = m_{flow} * h$ ) [W]
Xi[Medium.nXi]	Independent mixture mass fractions $m_i/m$ in the connection point [kg/kg]

## 658 Modelica.Media.Examples.Tests.Components.FluidPort\_b

mXi_flow[Medium.nXi]	Mass flow rates of the independent substances from the connection point into the component (if $m_{\text{flow}} > 0$ , $mX_{\text{flow}} = m_{\text{flow}} \cdot X$ ) [kg/s]
C[Medium.nC]	properties $c_i/m$ in the connection point
mC_flow[Medium.nC]	Flow rates of auxiliary properties from the connection point into the component (if $m_{\text{flow}} > 0$ , $mC_{\text{flow}} = m_{\text{flow}} \cdot C$ )

## Modelica.Media.Examples.Tests.Components.PortVolume

Fixed volume associated with a port by the finite volume method



### Information

This component models the **volume of fixed size** that is associated with the **fluid port** to which it is connected. This means that all medium properties inside the volume, are identical to the port medium properties. In particular, the specific enthalpy inside the volume (= medium.h) is always identical to the specific enthalpy in the port (port.h = medium.h). Usually, this model is used when discretizing a component according to the finite volume method into volumes in internal ports that only store energy and mass and into transport elements that just transport energy, mass and momentum between the internal ports without storing these quantities during the transport.

### Parameters

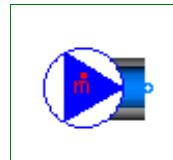
Name	Default	Description
V	1e-6	Fixed size of junction volume [m <sup>3</sup> ]
Initial pressure or initial density		
use_p_start	true	select p_start or d_start
p_start	101325	Initial pressure [Pa]
d_start	1	Initial density [kg/m <sup>3</sup> ]
Initial temperature or initial specific enthalpy		
use_T_start	true	select T_start or h_start
T_start	Modelica.Slunits.Conversions...	Initial temperature [K]
h_start	1.e4	Initial specific enthalpy [J/kg]
Only for multi-substance flow		
X_start[Medium.nX]		Initial mass fractions $m_i/m$ [kg/kg]

### Connectors

Name	Description
port	

## Modelica.Media.Examples.Tests.Components.FixedMassFlowRate

Ideal pump that produces a constant mass flow rate from a large reservoir at fixed temperature and mass fraction



## Information

## Parameters

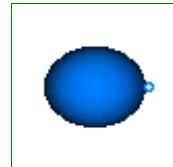
Name	Default	Description
m_flow		Fixed mass flow rate from an infinite reservoir to the fluid port [kg/s]
X_ambient[Medium.nX]		Ambient mass fractions m_i/m of reservoir [kg/kg]
Ambient temperature or ambient specific enthalpy		
use_T_ambient	true	select T_ambient or h_ambient
T_ambient	Modelica.Slunits.Conversions...	Ambient temperature [K]
h_ambient	1.e4	Ambient specific enthalpy [J/kg]

## Connectors

Name	Description
port	

## Modelica.Media.Examples.Tests.Components.FixedAmbient

Ambient pressure, temperature and mass fraction source



## Information

Model **FixedAmbient\_pt** defines constant values for ambient conditions:

- Ambient pressure.
- Ambient temperature.
- Ambient mass fractions (only for multi-substance flow).

Note, that ambient temperature and mass fractions have only an effect if the mass flow is from the ambient into the port. If mass is flowing from the port into the ambient, the ambient definitions, with exception of ambient pressure, do not have an effect.

## Parameters

Name	Default	Description
Ambient pressure or ambient density		
use_p_ambient	true	select p_ambient or d_ambient
p_ambient	101325	Ambient pressure [Pa]
d_ambient	1	Ambient density [kg/m <sup>3</sup> ]
Ambient temperature or ambient specific enthalpy		
use_T_ambient	true	select T_ambient or h_ambient
T_ambient	Modelica.Slunits.Conversions...	Ambient temperature [K]
h_ambient	1.e4	Ambient specific enthalpy [J/kg]
Only for multi-substance flow		
X_ambient[Medium.nX]		Ambient mass fractions m_i/m [kg/kg]

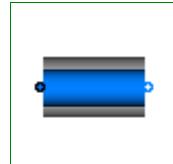
## Connectors

Name	Description
port	

---

## Modelica.Media.Examples.Tests.Components.ShortPipe

Simple pressure loss in pipe



## Information

Model **ShortPipe** defines a simple pipe model with pressure loss due to friction. It is assumed that no mass or energy is stored in the pipe. The details of the pipe friction model are described [here](#).

## Parameters

Name	Default	Description
dp_nominal		Nominal pressure drop [Pa]
m_flow_nominal		Nominal mass flow rate at nominal pressure drop [kg/s]

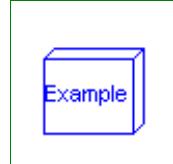
## Connectors

Name	Description
port_a	
port_b	

---

## Modelica.Media.Examples.Tests.Components.PartialTestModel

Basic test model to test a medium



## Information

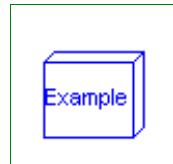
## Parameters

Name	Default	Description
p_start	Medium.p_default	Initial value of pressure [Pa]
T_start	Medium.T_default	Initial value of temperature [K]
h_start	Medium.h_default	Initial value of specific enthalpy [J/kg]
X_start[Medium.nX ]	Medium.X_default	Initial value of mass fractions

---

## Modelica.Media.Examples.Tests.Components.PartialTestModel2

slightly larger test model to test a medium



## Information

## Parameters

Name	Default	Description

p_start	1.0e5	Initial value of pressure [Pa]
T_start	300	Initial value of temperature [K]
h_start	1	Initial value of specific enthalpy [J/kg]
X_start[Medium.nX ]	Medium.reference_X	Initial value of mass fractions

## Modelica.Media.Examples.Tests.MediaTestModels

Test models to test all media

### Information

#### Package Content

Name	Description
<input type="checkbox"/> Air	Test models of library Modelica.Media.Air
<input type="checkbox"/> IdealGases	Test models of library Modelica.Media.IdealGases
<input type="checkbox"/> Incompressible	Test models of library Modelica.Media.Incompressible
<input type="checkbox"/> Water	Test models of library Modelica.Media.Water
<input type="checkbox"/> LinearFluid	Test models of library Modelica.Media.Incompressible

## Modelica.Media.Examples.Tests.MediaTestModels.Air

Test models of library Modelica.Media.Air

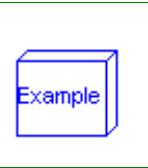
### Information

#### Package Content

Name	Description
<input type="checkbox"/> SimpleAir	Test Modelica.Media.Air.SimpleAir
<input type="checkbox"/> DryAirNasa	Test Modelica.Media.Air.DryAirNasa
<input type="checkbox"/> MoistAir	Test Modelica.Media.Air.MoistAir

## Modelica.Media.Examples.Tests.MediaTestModels.Air.SimpleAir

Test Modelica.Media.Air.SimpleAir



### Information

#### Parameters

Name	Default	Description
replaceable package Medium	PartialMedium	Medium model
p_start	Medium.p_default	Initial value of pressure [Pa]

## 662 Modelica.Media.Examples.Tests.MediaTestModels.Air.SimpleAir

---

T_start	Medium.T_default	Initial value of temperature [K]
h_start	Medium.h_default	Initial value of specific enthalpy [J/kg]
X_start[Medium.nX]	Medium.X_default	Initial value of mass fractions

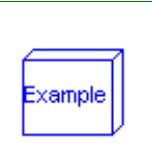
### Connectors

Name	Description
replaceable package Medium	Medium model

---

## Modelica.Media.Examples.Tests.MediaTestModels.Air.DryAirNasa

Test Modelica.Media.Air.DryAirNasa



### Information

### Parameters

Name	Default	Description
replaceable package Medium	PartialMedium	Medium model
p_start	Medium.p_default	Initial value of pressure [Pa]
T_start	Medium.T_default	Initial value of temperature [K]
h_start	Medium.h_default	Initial value of specific enthalpy [J/kg]
X_start[Medium.nX]	Medium.X_default	Initial value of mass fractions

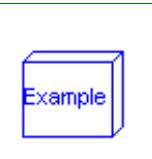
### Connectors

Name	Description
replaceable package Medium	Medium model

---

## Modelica.Media.Examples.Tests.MediaTestModels.Air.MoistAir

Test Modelica.Media.Air.MoistAir



### Information

### Parameters

Name	Default	Description
replaceable package Medium	PartialMedium	Medium model
p_start	Medium.p_default	Initial value of pressure [Pa]
T_start	Medium.T_default	Initial value of temperature [K]
h_start	Medium.h_default	Initial value of specific enthalpy [J/kg]
X_start[Medium.nX]	Medium.X_default	Initial value of mass fractions

### Connectors

Name	Description
replaceable package Medium	Medium model

**Modelica.Media.Examples.Tests.MediaTestModels.IdealGases**

Test models of library Modelica.Media.IdealGases

**Package Content**

Name	Description
Air	Test single gas Modelica.Media.IdealGases.SingleGases.Air
SimpleNaturalGas	Test mixture gas Modelica.Media.IdealGases.MixtureGases.SimpleNaturalGas
SimpleNaturalGasFixedComposition	Test mixture gas Modelica.Media.IdealGases.MixtureGases.SimpleNaturalGas

**Modelica.Media.Examples.Tests.MediaTestModels.IdealGases.Air**

Test single gas Modelica.Media.IdealGases.SingleGases.Air

**Information****Parameters**

Name	Default	Description
replaceable package Medium	PartialMedium	Medium model
p_start	Medium.p_default	Initial value of pressure [Pa]
T_start	Medium.T_default	Initial value of temperature [K]
h_start	Medium.h_default	Initial value of specific enthalpy [J/kg]
X_start[Medium.nX]	Medium.X_default	Initial value of mass fractions

**Connectors**

Name	Description
replaceable package Medium	Medium model

**Modelica.Media.Examples.Tests.MediaTestModels.IdealGases.SimpleNaturalGas**

Test mixture gas Modelica.Media.IdealGases.MixtureGases.SimpleNaturalGas

**Information****Parameters**

Name	Default	Description
replaceable package Medium	PartialMedium	Medium model
p_start	Medium.p_default	Initial value of pressure [Pa]
T_start	Medium.T_default	Initial value of temperature [K]
h_start	Medium.h_default	Initial value of specific enthalpy [J/kg]
X_start[Medium.nX]	Medium.X_default	Initial value of mass fractions

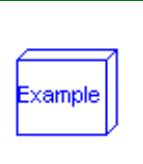
## Connectors

Name	Description
replaceable package Medium	Medium model

---

## Modelica.Media.Examples.Tests.MediaTestModels.IdealGases.SimpleNaturalGasFixedComposition

Test mixture gas Modelica.Media.IdealGases.MixtureGases.SimpleNaturalGas



## Parameters

Name	Default	Description
replaceable package Medium	PartialMedium	Medium model
p_start	Medium.p_default	Initial value of pressure [Pa]
T_start	Medium.T_default	Initial value of temperature [K]
h_start	Medium.h_default	Initial value of specific enthalpy [J/kg]
X_start[Medium.nX]	Medium.X_default	Initial value of mass fractions

## Connectors

Name	Description
replaceable package Medium	Medium model

---

## Modelica.Media.Examples.Tests.MediaTestModels.Incompressible

Test models of library Modelica.Media.Incompressible

## Information

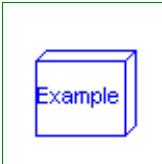
### Package Content

Name	Description
Glycol47	Test Modelica.Media.Incompressible.Examples.Glycol47
Essotherm650	Test Modelica.Media.Incompressible.Examples.Essotherm65

---

## Modelica.Media.Examples.Tests.MediaTestModels.Incompressible.Glycol47

Test Modelica.Media.Incompressible.Examples.Glycol47



## Information

### Parameters

Name	Default	Description
replaceable package Medium	PartialMedium	Medium model
p_start	Medium.p_default	Initial value of pressure [Pa]
T_start	Medium.T_default	Initial value of temperature [K]

h_start	Medium.h_default	Initial value of specific enthalpy [J/kg]
X_start[Medium.nX]	Medium.X_default	Initial value of mass fractions

## Connectors

Name	Description
replaceable package Medium	Medium model

## Modelica.Media.Examples.Tests.MediaTestModels.Incompressible.Essotherm650

Test Modelica.Media.Incompressible.Examples.Essotherm65



## Information

## Parameters

Name	Default	Description
replaceable package Medium	PartialMedium	Medium model
p_start	Medium.p_default	Initial value of pressure [Pa]
T_start	Medium.T_default	Initial value of temperature [K]
h_start	Medium.h_default	Initial value of specific enthalpy [J/kg]
X_start[Medium.nX]	Medium.X_default	Initial value of mass fractions

## Connectors

Name	Description
replaceable package Medium	Medium model

## Modelica.Media.Examples.Tests.MediaTestModels.Water

Test models of library Modelica.Media.Water

## Information

## Package Content

Name	Description
<input type="checkbox"/> ConstantPropertyLiquidWater	Test Modelica.Media.Water.ConstantPropertyLiquidWater
<input type="checkbox"/> IdealSteam	Test Modelica.Media.Water.IdealSteam
<input type="checkbox"/> WaterIF97OnePhase_ph	Test Modelica.Media.Water.WaterIF97OnePhase_ph
<input type="checkbox"/> WaterIF97_pT	Test Modelica.Media.Water.WaterIF97_pT
<input type="checkbox"/> WaterIF97_ph	Test Modelica.Media.Water.WaterIF97_ph

## Modelica.Media.Examples.Tests.MediaTestModels.Water.ConstantPropertyLiquidWater

Test Modelica.Media.Water.ConstantPropertyLiquidWater



## Information

## Parameters

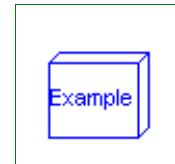
Name	Default	Description
replaceable package Medium	PartialMedium	Medium model
p_start	Medium.p_default	Initial value of pressure [Pa]
T_start	Medium.T_default	Initial value of temperature [K]
h_start	Medium.h_default	Initial value of specific enthalpy [J/kg]
X_start[Medium.nX]	Medium.X_default	Initial value of mass fractions

## Connectors

Name	Description
replaceable package Medium	Medium model

## Modelica.Media.Examples.Tests.MediaTestModels.Water.IdealSteam

Test Modelica.Media.Water.IdealSteam



## Information

## Parameters

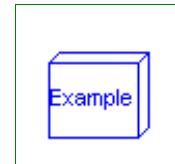
Name	Default	Description
replaceable package Medium	PartialMedium	Medium model
p_start	Medium.p_default	Initial value of pressure [Pa]
T_start	Medium.T_default	Initial value of temperature [K]
h_start	Medium.h_default	Initial value of specific enthalpy [J/kg]
X_start[Medium.nX]	Medium.X_default	Initial value of mass fractions

## Connectors

Name	Description
replaceable package Medium	Medium model

## Modelica.Media.Examples.Tests.MediaTestModels.Water.WaterIF97OnePhase\_ph

Test Modelica.Media.Water.WaterIF97OnePhase\_ph



## Information

## Parameters

Name	Default	Description
replaceable package Medium	PartialMedium	Medium model
p_start	Medium.p_default	Initial value of pressure [Pa]
T_start	Medium.T_default	Initial value of temperature [K]
h_start	Medium.h_default	Initial value of specific enthalpy [J/kg]

X_start[Medium.nX]	Medium.X_default	Initial value of mass fractions
--------------------	------------------	---------------------------------

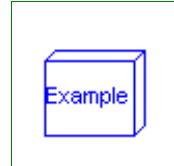
## Connectors

Name	Description
replaceable package Medium	Medium model

---

## **Modelica.Media.Examples.Tests.MediaTestModels.Water.WaterIF97\_pT**

Test Modelica.Media.Water.WaterIF97\_pT



## Information

## Parameters

Name	Default	Description
replaceable package Medium	PartialMedium	Medium model
p_start	Medium.p_default	Initial value of pressure [Pa]
T_start	Medium.T_default	Initial value of temperature [K]
h_start	Medium.h_default	Initial value of specific enthalpy [J/kg]
X_start[Medium.nX]	Medium.X_default	Initial value of mass fractions

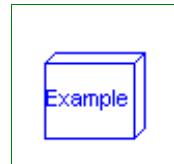
## Connectors

Name	Description
replaceable package Medium	Medium model

---

## **Modelica.Media.Examples.Tests.MediaTestModels.Water.WaterIF97\_ph**

Test Modelica.Media.Water.WaterIF97\_ph



## Information

## Parameters

Name	Default	Description
replaceable package Medium	PartialMedium	Medium model
p_start	Medium.p_default	Initial value of pressure [Pa]
T_start	Medium.T_default	Initial value of temperature [K]
h_start	Medium.h_default	Initial value of specific enthalpy [J/kg]
X_start[Medium.nX]	Medium.X_default	Initial value of mass fractions

## Connectors

Name	Description
replaceable package Medium	Medium model

**Modelica.Media.Examples.Tests.MediaTestModels.LinearFluid**

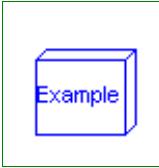
Test models of library Modelica.Media.Incompressible

**Information****Package Content**

Name	Description
 LinearColdWater	Test Modelica.Media.Incompressible.Examples.Glycol47
 LinearWater_pT	Test Modelica.Media.Incompressible.Examples.Essotherm65

**Modelica.Media.Examples.Tests.MediaTestModels.LinearFluid.LinearColdWater**

Test Modelica.Media.Incompressible.Examples.Glycol47

**Information****Parameters**

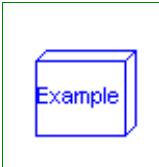
Name	Default	Description
replaceable package Medium	PartialMedium	Medium model
p_start	Medium.p_default	Initial value of pressure [Pa]
T_start	Medium.T_default	Initial value of temperature [K]
h_start	Medium.h_default	Initial value of specific enthalpy [J/kg]
X_start[Medium.nX]	Medium.X_default	Initial value of mass fractions

**Connectors**

Name	Description
replaceable package Medium	Medium model

**Modelica.Media.Examples.Tests.MediaTestModels.LinearFluid.LinearWater\_pT**

Test Modelica.Media.Incompressible.Examples.Essotherm65

**Information****Parameters**

Name	Default	Description
replaceable package Medium	PartialMedium	Medium model
p_start	Medium.p_default	Initial value of pressure [Pa]
T_start	Medium.T_default	Initial value of temperature [K]
h_start	Medium.h_default	Initial value of specific enthalpy [J/kg]
X_start[Medium.nX]	Medium.X_default	Initial value of mass fractions

## Connectors

Name	Description
replaceable package Medium	Medium model

## Modelica.Media.Examples.SolveOneNonlinearEquation

Demonstrate how to solve one non-linear algebraic equation in one unknown

## Information

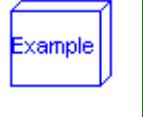
This package demonstrates how to solve one non-linear algebraic equation in one unknown with function Modelica.Media.Common.OneNonLinearEquation.

## Package Content

Name	Description
<input type="checkbox"/> Inverse_sine	Solve $y = A \cdot \sin(w \cdot x)$ for $x$ , given $y$
<input type="checkbox"/> Inverse_sh_T	Solve $h = h_T(T)$ , $s = s_T(T)$ for $T$ , if $h$ or $s$ is given for ideal gas NASA
<input type="checkbox"/> InverselIncompressible_sh_T	inverse computation for incompressible media
<input type="checkbox"/> Inverse_sh_TX	Solve $h = h_{TX}(TX)$ for $T$ , if $h$ is given for ideal gas NASA

## Modelica.Media.Examples.SolveOneNonlinearEquation.Inverse\_sine

Solve  $y = A \cdot \sin(w \cdot x)$  for  $x$ , given  $y$



## Information

This models solves the following non-linear equation

$y = A \cdot \sin(w \cdot x)$ ; -> determine  $x$  for given  $y$

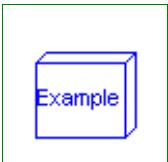
Translate model "Inverse\_sine" and simulate for 0 sec. The result is printed to the output window.

## Parameters

Name	Default	Description
y_zero	0.5	Desired value of $A \cdot \sin(w \cdot x)$
x_min	-1.7	Minimum value of $x$ _zero
x_max	1.7	Maximum value of $x$ _zero
A	1	
w	1	
data	Inverse_sine_definition.f_no...	

## Modelica.Media.Examples.SolveOneNonlinearEquation.Inverse\_sh\_T

Solve  $h = h_T(T)$ ,  $s = s_T(T)$  for  $T$ , if  $h$  or  $s$  is given for ideal gas NASA



## 670 Modelica.Media.Examples.SolveOneNonlinearEquation.Inverse\_sh\_T

---

### Information

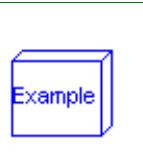
### Parameters

Name	Default	Description
T_min	300	[K]
T_max	500	[K]
p	1.0e5	[Pa]
h_min	Medium.h_T(Medium.data, T_min)	[J/kg]
h_max	Medium.h_T(Medium.data, T_max)	[J/kg]
s_min	Medium.specificEntropy(Medium.data, T_min)	[J/(kg.K)]
s_max	Medium.specificEntropy(Medium.data, T_max)	[J/(kg.K)]

---

## Modelica.Media.Examples.SolveOneNonlinearEquation.InverseIncompressible\_sh\_T

inverse computation for incompressible media



### Information

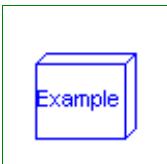
### Parameters

Name	Default	Description
T_min	Medium.T_min	[K]
T_max	Medium.T_max	[K]
p	1.0e5	[Pa]
h_min	Medium.h_T(Medium.T_min)	[J/kg]
h_max	Medium.h_T(Medium.T_max)	[J/kg]
s_min	Medium.specificEntropy(Medium.data, T_min)	[J/(kg.K)]
s_max	Medium.specificEntropy(Medium.data, T_max)	[J/(kg.K)]

---

## Modelica.Media.Examples.SolveOneNonlinearEquation.Inverse\_sh\_TX

Solve  $h = h_{TX}(T)$  for  $T$ , if  $h$  is given for ideal gas NASA



### Information

### Parameters

Name	Default	Description
T_min	300	[K]
T_max	500	[K]
p	1.0e5	[Pa]
s_min	Medium.specificEntropy(Medium.data, T_min)	[J/(kg.K)]
s_max	Medium.specificEntropy(Medium.data, T_max)	[J/(kg.K)]

---

## Modelica.Media.Interfaces

### Interfaces for media models

#### Information

This package provides basic interfaces definitions of media models for different kind of media.

#### Package Content

Name	Description
 TemplateMedium	Template for media models
 PartialMedium	Partial medium properties (base package of all media packages)
 PartialPureSubstance	base class for pure substances of one chemical substance
 PartialLinearFluid	Generic pure liquid model with constant cp, compressibility and thermal expansion coefficients
 PartialMixtureMedium	base class for pure substances of several chemical substances
 PartialCondensingGases	Base class for mixtures of condensing and non-condensing gases
 PartialTwoPhaseMedium	
 PartialSimpleMedium	Medium model with linear dependency of u, h from temperature. All other quantities, especially density, are constant.
 PartialSimpleIdealGasMedium	Medium model of Ideal gas with constant cp and cv. All other quantities, e.g. transport properties, are constant.

## Modelica.Media.Interfaces.TemplateMedium

### Template for media models

#### Information

This package is a **template** for **new medium** models. For a new medium model just make a copy of this package, remove the "partial" keyword from the package and provide the information that is requested in the comments of the Modelica source.

#### Package Content

Name	Description
 BaseProperties	Base properties of medium
 ThermodynamicState	a selection of variables that uniquely defines the thermodynamic state
 dynamicViscosity	Return dynamic viscosity
 thermalConductivity	Return thermal conductivity
 specificEntropy	Return specific entropy
 specificHeatCapacityCp	Return specific heat capacity at constant pressure
 specificHeatCapacityCv	Return specific heat capacity at constant volume
 isentropicExponent	Return isentropic exponent
 velocityOfSound	Return velocity of sound

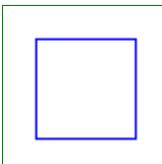
<b>Inherited</b>	
mediumName="unusablePartialMedium"	Name of the medium
substanceNames={mediumName}	Names of the mixture substances. Set substanceNames={mediumName} if only one substance.
extraPropertiesNames=fill("", 0)	Names of the additional (extra) transported properties. Set extraPropertiesNames=fill("",0) if unused
singleState	= true, if u and d are not a function of pressure
reducedX=true	= true if medium contains the equation sum(X) = 1.0; set reducedX=true if only one substance (see docu for details)
fixedX=false	= true if medium contains the equation X = reference_X
reference_p=101325	Reference pressure of Medium: default 1 atmosphere
reference_T=298.15	Reference temperature of Medium: default 25 deg Celsius
reference_X=if nX == 0 then fill(0, nX) else fill(1/nX, nX)	Default mass fractions of medium
p_default=101325	Default value for pressure of medium (for initialization)
T_default=Modelica.SIunits.Conversions.from_degC(20)	Default value for temperature of medium (for initialization)
h_default=specificEnthalpy_pTX(p_default, T_default, X_default)	Default value for specific enthalpy of medium (for initialization)
X_default=reference_X	Default value for mass fractions of medium (for initialization)
nS=size(substanceNames, 1)	Number of substances
nX=if nS == 1 then 0 else nS	Number of mass fractions (= 0, if only one substance)
nXi=if fixedX then 0 else if reducedX then nS - 1 else nX	Number of structurally independent mass fractions (see docu for details)
nC=size(extraPropertiesNames, 1)	Number of extra (outside of standard mass-balance) transported properties
 FluidConstants	critical, triple, molecular and other standard data of fluid
 BasePropertiesRecord	Variables contained in every instance of BaseProperties
 setState_pTX	Return thermodynamic state as function of p, T and composition X or Xi
 setState_phX	Return thermodynamic state as function of p, h and composition X or Xi
 setState_psX	Return thermodynamic state as function of p, s and composition X or Xi
 setState_dTX	Return thermodynamic state as function of d, T and composition X or Xi
 prandtlNumber	Return the Prandtl number
 pressure	Return pressure
 temperature	Return temperature
 density	Return density
 specificEnthalpy	Return specific enthalpy
 specificInternalEnergy	Return specific internal energy
 specificGibbsEnergy	Return specific Gibbs energy
 specificHelmholtzEnergy	Return specific Helmholtz energy
 heatCapacity_cp	alias for deprecated name

<a href="#"> heatCapacity_cv</a>	alias for deprecated name
<a href="#"> isentropicEnthalpy</a>	Return isentropic enthalpy
<a href="#"> isobaricExpansionCoefficient</a>	Return overall the isobaric expansion coefficient beta
<a href="#"> beta</a>	alias for isobaricExpansionCoefficient for user convenience
<a href="#"> isothermalCompressibility</a>	Return overall the isothermal compressibility factor
<a href="#"> kappa</a>	alias of isothermalCompressibility for user convenience
<a href="#"> density_derP_h</a>	Return density derivative wrt pressure at const specific enthalpy
<a href="#"> density_derH_p</a>	Return density derivative wrt specific enthalpy at constant pressure
<a href="#"> density_derP_T</a>	Return density derivative wrt pressure at const temperature
<a href="#"> density_derT_p</a>	Return density derivative wrt temperature at constant pressure
<a href="#"> density_derX</a>	Return density derivative wrt mass fraction
<a href="#"> molarMass</a>	Return the molar mass of the medium
<a href="#"> specificEnthalpy_pTX</a>	Return specific enthalpy from p, T, and X or Xi
<a href="#"> density_pTX</a>	Return density from p, T, and X or Xi
<a href="#"> temperature_phX</a>	Return temperature from p, h, and X or Xi
<a href="#"> density_phX</a>	Return density from p, h, and X or Xi
<a href="#"> temperature_psX</a>	Return temperature from p,s, and X or Xi
<a href="#"> density_psX</a>	Return density from p, s, and X or Xi
<a href="#"> specificEnthalpy_psX</a>	Return specific enthalpy from p, s, and X or Xi
AbsolutePressure	Type for absolute pressure with medium specific attributes
Density	Type for density with medium specific attributes
DynamicViscosity	Type for dynamic viscosity with medium specific attributes
EnthalpyFlowRate	Type for enthalpy flow rate with medium specific attributes
MassFlowRate	Type for mass flow rate with medium specific attributes
MassFraction	Type for mass fraction with medium specific attributes
MoleFraction	Type for mole fraction with medium specific attributes
MolarMass	Type for molar mass with medium specific attributes
MolarVolume	Type for molar volume with medium specific attributes
IsentropicExponent	Type for isentropic exponent with medium specific attributes
SpecificEnergy	Type for specific energy with medium specific attributes
SpecificInternalEnergy	Type for specific internal energy with medium specific attributes
SpecificEnthalpy	Type for specific enthalpy with medium specific attributes
SpecificEntropy	Type for specific entropy with medium specific attributes
SpecificHeatCapacity	Type for specific heat capacity with medium specific attributes
SurfaceTension	Type for surface tension with medium specific attributes
Temperature	Type for temperature with medium specific attributes
ThermalConductivity	Type for thermal conductivity with medium specific attributes
PrandtlNumber	Type for Prandtl number with medium specific attributes

VelocityOfSound	Type for velocity of sound with medium specific attributes
ExtraProperty	Type for unspecified, mass-specific property transported by flow
CumulativeExtraProperty	Type for conserved integral of unspecified, mass specific property
ExtraPropertyFlowRate	Type for flow rate of unspecified, mass-specific property
IsobaricExpansionCoefficient	Type for isobaric expansion coefficient with medium specific attributes
DipoleMoment	Type for dipole moment with medium specific attributes
DerDensityByPressure	Type for partial derivative of density with respect to pressure with medium specific attributes
DerDensityByEnthalpy	Type for partial derivative of density with respect to enthalpy with medium specific attributes
DerEnthalpyByPressure	Type for partial derivative of enthalpy with respect to pressure with medium specific attributes
DerDensityByTemperature	Type for partial derivative of density with respect to temperature with medium specific attributes
<input checked="" type="checkbox"/> Choices	Types, constants to define menu choices

## Modelica.Media.Interfaces.TemplateMedium.BaseProperties

Base properties of medium



### Parameters

Name	Default	Description
standardOrderComponents	true	if true, last element in components is computed from $1 - \sum(X_i)$
<b>Advanced</b>		
preferredMediumStates	false	= true if StateSelect.prefer shall be used for the independent property variables of the medium

## Modelica.Media.Interfaces.TemplateMedium.ThermodynamicState

a selection of variables that uniquely defines the thermodynamic state

### Modelica definition

```
redeclare replaceable record ThermodynamicState
  "a selection of variables that uniquely defines the thermodynamic state"
  AbsolutePressure p "Absolute pressure of medium";
  Temperature T "Temperature of medium";
end ThermodynamicState;
```



## Modelica.Media.Interfaces.TemplateMedium.dynamicViscosity

Return dynamic viscosity

**Inputs**

Name	Default	Description
state		

**Outputs**

Name	Description
eta	Dynamic viscosity [Pa.s]

**Modelica.Media.Interfaces.TemplateMedium.thermalConductivity**

Return thermal conductivity

**Inputs**

Name	Default	Description
state		

**Outputs**

Name	Description
lambda	Thermal conductivity [W/(m.K)]

**Modelica.Media.Interfaces.TemplateMedium.specificEntropy**

Return specific entropy

**Inputs**

Name	Default	Description
state		

**Outputs**

Name	Description
s	Specific entropy [J/(kg.K)]

**Modelica.Media.Interfaces.TemplateMedium.specificHeatCapacityCp**

Return specific heat capacity at constant pressure

**Inputs**

Name	Default	Description
state		

**Outputs**

Name	Description
cp	Specific heat capacity at constant pressure [J/(kg.K)]

**Modelica.Media.Interfaces.TemplateMedium.specificHeatCapacityCv**

Return specific heat capacity at constant volume

**Inputs**

Name	Default	Description
state		

**Outputs**

Name	Description
cv	Specific heat capacity at constant volume [J/(kg.K)]

---

**Modelica.Media.Interfaces.TemplateMedium.isentropicExponent**

Return isentropic exponent

**Inputs**

Name	Default	Description
state		

**Outputs**

Name	Description
gamma	Isentropic exponent [1]

---

**Modelica.Media.Interfaces.TemplateMedium.velocityOfSound**

Return velocity of sound

**Inputs**

Name	Default	Description
state		

**Outputs**

Name	Description
a	Velocity of sound [m/s]

---

**Modelica.Media.Interfaces.PartialMedium**

Partial medium properties (base package of all media packages)

**Information**

PartialMedium is a package and contains all **declarations** for a medium. This means that constants, models, and functions are defined that every medium is supposed to support (some of them are optional). A

medium package inherits from **PartialMedium** and provides the equations for the medium. The details of this package are described in [Modelica.Media.UsersGuide](#).

## Package Content

Name	Description
mediumName="unusablePartialMedium"	Name of the medium
substanceNames={mediumName}	Names of the mixture substances. Set substanceNames={mediumName} if only one substance.
extraPropertiesNames=fill("", 0)	Names of the additional (extra) transported properties. Set extraPropertiesNames=fill("",0) if unused
singleState	= true, if u and d are not a function of pressure
reducedX=true	= true if medium contains the equation sum(X) = 1.0; set reducedX=true if only one substance (see docu for details)
fixedX=false	= true if medium contains the equation X = reference_X
reference_p=101325	Reference pressure of Medium: default 1 atmosphere
reference_T=298.15	Reference temperature of Medium: default 25 deg Celsius
reference_X;if nX == 0 then fill(0, nX) else fill(1/nX, nX)	Default mass fractions of medium
p_default=101325	Default value for pressure of medium (for initialization)
T_default=Modelica.SIunits.Conversions.from_degC(20)	Default value for temperature of medium (for initialization)
h_default=specificEnthalpy_pTX(p_default, T_default, X_default)	Default value for specific enthalpy of medium (for initialization)
X_default=reference_X	Default value for mass fractions of medium (for initialization)
nS=size(substanceNames, 1)	Number of substances
nX;if nS == 1 then 0 else nS	Number of mass fractions (= 0, if only one substance)
nXi;if fixedX then 0 else if reducedX then nS - 1 else nX	Number of structurally independent mass fractions (see docu for details)
nC=size(extraPropertiesNames, 1)	Number of extra (outside of standard mass-balance) transported properties
 FluidConstants	critical, triple, molecular and other standard data of fluid
 ThermodynamicState	Minimal variable set that is available as input argument to every medium function
 BasePropertiesRecord	Variables contained in every instance of BaseProperties
 BaseProperties	Base properties (p, d, T, h, u, R, MM and, if applicable, X) of a medium
 setState_pTX	Return thermodynamic state as function of p, T and composition X or Xi
 setState_phX	Return thermodynamic state as function of p, h and composition X or Xi
 setState_psX	Return thermodynamic state as function of p, s and composition X or Xi
 setState_dTX	Return thermodynamic state as function of d, T and composition X or Xi
 dynamicViscosity	Return dynamic viscosity
 thermalConductivity	Return thermal conductivity
 prandtlNumber	Return the Prandtl number

<code>(f) pressure</code>	Return pressure
<code>(f) temperature</code>	Return temperature
<code>(f) density</code>	Return density
<code>(f) specificEnthalpy</code>	Return specific enthalpy
<code>(f) specificInternalEnergy</code>	Return specific internal energy
<code>(f) specificEntropy</code>	Return specific entropy
<code>(f) specificGibbsEnergy</code>	Return specific Gibbs energy
<code>(f) specificHelmholtzEnergy</code>	Return specific Helmholtz energy
<code>(f) specificHeatCapacityCp</code>	Return specific heat capacity at constant pressure
<code>(f) heatCapacity_cp</code>	alias for deprecated name
<code>(f) specificHeatCapacityCv</code>	Return specific heat capacity at constant volume
<code>(f) heatCapacity_cv</code>	alias for deprecated name
<code>(f) isentropicExponent</code>	Return isentropic exponent
<code>(f) isentropicEnthalpy</code>	Return isentropic enthalpy
<code>(f) velocityOfSound</code>	Return velocity of sound
<code>(f) isobaricExpansionCoefficient</code>	Return overall the isobaric expansion coefficient beta
<code>(f) beta</code>	alias for isobaricExpansionCoefficient for user convenience
<code>(f) isothermalCompressibility</code>	Return overall the isothermal compressibility factor
<code>(f) kappa</code>	alias of isothermalCompressibility for user convenience
<code>(f) density_derp_h</code>	Return density derivative wrt pressure at const specific enthalpy
<code>(f) density_derh_p</code>	Return density derivative wrt specific enthalpy at constant pressure
<code>(f) density_derp_T</code>	Return density derivative wrt pressure at const temperature
<code>(f) density_derT_p</code>	Return density derivative wrt temperature at constant pressure
<code>(f) density_derX</code>	Return density derivative wrt mass fraction
<code>(f) molarMass</code>	Return the molar mass of the medium
<code>(f) specificEnthalpy_pTX</code>	Return specific enthalpy from p, T, and X or Xi
<code>(f) density_pTX</code>	Return density from p, T, and X or Xi
<code>(f) temperature_phX</code>	Return temperature from p, h, and X or Xi
<code>(f) density_phX</code>	Return density from p, h, and X or Xi
<code>(f) temperature_psX</code>	Return temperature from p,s, and X or Xi
<code>(f) density_psX</code>	Return density from p, s, and X or Xi
<code>(f) specificEnthalpy_psX</code>	Return specific enthalpy from p, s, and X or Xi
<code>AbsolutePressure</code>	Type for absolute pressure with medium specific attributes
<code>Density</code>	Type for density with medium specific attributes
<code>DynamicViscosity</code>	Type for dynamic viscosity with medium specific attributes
<code>EnthalpyFlowRate</code>	Type for enthalpy flow rate with medium specific attributes

MassFlowRate	Type for mass flow rate with medium specific attributes
MassFraction	Type for mass fraction with medium specific attributes
MoleFraction	Type for mole fraction with medium specific attributes
MolarMass	Type for molar mass with medium specific attributes
MolarVolume	Type for molar volume with medium specific attributes
IsentropicExponent	Type for isentropic exponent with medium specific attributes
SpecificEnergy	Type for specific energy with medium specific attributes
SpecificInternalEnergy	Type for specific internal energy with medium specific attributes
SpecificEnthalpy	Type for specific enthalpy with medium specific attributes
SpecificEntropy	Type for specific entropy with medium specific attributes
SpecificHeatCapacity	Type for specific heat capacity with medium specific attributes
SurfaceTension	Type for surface tension with medium specific attributes
Temperature	Type for temperature with medium specific attributes
ThermalConductivity	Type for thermal conductivity with medium specific attributes
PrandtlNumber	Type for Prandtl number with medium specific attributes
VelocityOfSound	Type for velocity of sound with medium specific attributes
ExtraProperty	Type for unspecified, mass-specific property transported by flow
CumulativeExtraProperty	Type for conserved integral of unspecified, mass specific property
ExtraPropertyFlowRate	Type for flow rate of unspecified, mass-specific property
IsobaricExpansionCoefficient	Type for isobaric expansion coefficient with medium specific attributes
DipoleMoment	Type for dipole moment with medium specific attributes
DerDensityByPressure	Type for partial derivative of density with respect to pressure with medium specific attributes
DerDensityByEnthalpy	Type for partial derivative of density with respect to enthalpy with medium specific attributes
DerEnthalpyByPressure	Type for partial derivative of enthalpy with respect to pressure with medium specific attributes
DerDensityByTemperature	Type for partial derivative of density with respect to temperature with medium specific attributes
 Choices	Types, constants to define menu choices

## Types and constants

```

constant String mediumName = "unusablePartialMedium" "Name of the medium";

constant String substanceNames [:]={mediumName}
"Names of the mixture substances. Set substanceNames={mediumName} if only one
substance./";

constant String extraPropertiesNames [:]=fill("", 0)
"Names of the additional (extra) transported properties. Set
extraPropertiesNames=fill("\\\",0) if unused";

constant Boolean singleState
"= true, if u and d are not a function of pressure";

```

```
constant Boolean reducedX=true
"= true if medium contains the equation sum(X) = 1.0; set reducedX=true if
only one substance (see docu for details)";

constant Boolean fixedX=false
"= true if medium contains the equation X = reference_X";

constant AbsolutePressure reference_p=101325
"Reference pressure of Medium: default 1 atmosphere";

constant Temperature reference_T=298.15
"Reference temperature of Medium: default 25 deg Celsius";

constant MassFraction reference_X[nX]= if nX == 0 then fill(0,nX) else
fill(1/nX, nX)
"Default mass fractions of medium";

constant AbsolutePressure p_default=101325
"Default value for pressure of medium (for initialization)";

constant Temperature T_default = Modelica.SIunits.Conversions.from_degC(20)
"Default value for temperature of medium (for initialization)";

constant SpecificEnthalpy h_default = specificEnthalpy_pTX(p_default, T_default,
X_default)
"Default value for specific enthalpy of medium (for initialization)";

constant MassFraction X_default[nX]=reference_X
"Default value for mass fractions of medium (for initialization)";

final constant Integer nS=size(substanceNames, 1) "Number of substances";

constant Integer nX=if nS == 1 then 0 else nS
"Number of mass fractions (= 0, if only one substance)";

constant Integer nXi=if fixedX then 0 else if reducedX then nS - 1 else nX
"Number of structurally independent mass fractions (see docu for details)";

final constant Integer nC=size(extraPropertiesNames, 1)
"Number of extra (outside of standard mass-balance) transported properties";

type AbsolutePressure = SI.AbsolutePressure (
  min=0,
  max=1.e8,
  nominal=1.e5,
  start=1.e5) "Type for absolute pressure with medium specific attributes";

type Density = SI.Density (
  min=0,
  max=1.e5,
  nominal=1,
  start=1) "Type for density with medium specific attributes";

type DynamicViscosity = SI.DynamicViscosity (
```

```

min=0,
max=1.e8,
nominal=1.e-3,
start=1.e-3) "Type for dynamic viscosity with medium specific attributes";

type EnthalpyFlowRate = SI.EnthalpyFlowRate (
  nominal=1000.0,
  min=-1.0e8,
  max=1.e8) "Type for enthalpy flow rate with medium specific attributes";

type MassFlowRate = SI.MassFlowRate (
  quantity="MassFlowRate." + mediumName,
  min=-1.0e5,
  max=1.e5) "Type for mass flow rate with medium specific attributes";

type MassFraction = Real (
  quantity="MassFraction",
  final unit="kg/kg",
  min=0,
  max=1,
  nominal=0.1) "Type for mass fraction with medium specific attributes";

type MoleFraction = Real (
  quantity="MoleFraction",
  final unit="mol/mol",
  min=0,
  max=1,
  nominal=0.1) "Type for mole fraction with medium specific attributes";

type MolarMass = SI.MolarMass (
  min=0.001,
  max=0.25,
  nominal=0.032) "Type for molar mass with medium specific attributes";

type MolarVolume = SI.MolarVolume (
  min=1e-6,
  max=1.0e6,
  nominal=1.0) "Type for molar volume with medium specific attributes";

type IsentropicExponent = SI.RatioOfSpecificHeatCapacities (
  min=1,
  max=500000,
  nominal=1.2,
  start=1.2) "Type for isentropic exponent with medium specific attributes";

type SpecificEnergy = SI.SpecificEnergy (
  min=-1.0e8,
  max=1.e8,
  nominal=1.e6) "Type for specific energy with medium specific attributes";

type SpecificInternalEnergy = SpecificEnergy
"Type for specific internal energy with medium specific attributes";

type SpecificEnthalpy = SI.SpecificEnthalpy (
  min=-1.0e8,

```

```
max=1.e8,
nominal=1.e6)
"Type for specific enthalpy with medium specific attributes";

type SpecificEntropy = SI.SpecificEntropy (
min=-1.e6,
max=1.e6,
nominal=1.e3) "Type for specific entropy with medium specific attributes";

type SpecificHeatCapacity = SI.SpecificHeatCapacity (
min=0,
max=1.e6,
nominal=1.e3,
start=1.e3)
"Type for specific heat capacity with medium specific attributes";

type SurfaceTension = SI.SurfaceTension
"Type for surface tension with medium specific attributes";

type Temperature = SI.Temperature (
min=1,
max=1.e4,
nominal=300,
start=300) "Type for temperature with medium specific attributes";

type ThermalConductivity = SI.ThermalConductivity (
min=0,
max=500,
nominal=1,
start=1) "Type for thermal conductivity with medium specific attributes";

type PrandtlNumber = SI.PrandtlNumber (
min=1e-3,
max=1e5,
nominal=1.0) "Type for Prandtl number with medium specific attributes";

type VelocityOfSound = SI.Velocity (
min=0,
max=1.e5,
nominal=1000,
start=1000) "Type for velocity of sound with medium specific attributes";

type ExtraProperty = Real (min=0.0, start=1.0)
"Type for unspecified, mass-specific property transported by flow";

type CumulativeExtraProperty = Real (min=0.0, start=1.0)
"Type for conserved integral of unspecified, mass specific property";

type ExtraPropertyFlowRate = Real
"Type for flow rate of unspecified, mass-specific property";

type IsobaricExpansionCoefficient = Real (
min=1e-8,
max=1.0e8,
unit="1/K")
```

```

"Type for isobaric expansion coefficient with medium specific attributes";

type DipoleMoment = Real (
  min=0.0,
  max=2.0,
  unit="debye",
  quantity="ElectricDipoleMoment")
"Type for dipole moment with medium specific attributes";

type DerDensityByPressure = SI.DerDensityByPressure
"Type for partial derivative of density with respect to pressure with medium
specific attributes";

type DerDensityByEnthalpy = SI.DerDensityByEnthalpy
"Type for partial derivative of density with respect to enthalpy with medium
specific attributes";

type DerEnthalpyByPressure = SI.DerEnthalpyByPressure
"Type for partial derivative of enthalpy with respect to pressure with medium
specific attributes";

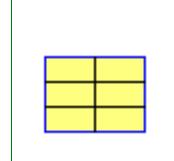
type DerDensityByTemperature = SI.DerDensityByTemperature
"Type for partial derivative of density with respect to temperature with medium
specific attributes";

```

---

## Modelica.Media.Interfaces.PartialMedium.FluidConstants

**critical, triple, molecular and other standard data of fluid**



### Modelica definition

```

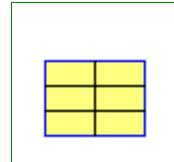
replaceable record FluidConstants
  "critical, triple, molecular and other standard data of fluid"
  extends Modelica.Icons.Record;
  String iupacName "complete IUPAC name (or common name, if non-existent)";
  String casRegistryNumber
    "chemical abstracts sequencing number (if it exists)";
  String chemicalFormula
    "Chemical formula, (brutto, nomenclature according to Hill)";
  String structureFormula "Chemical structure formula";
  MolarMass molarMass "molar mass";
end FluidConstants;

```

---

## Modelica.Media.Interfaces.PartialMedium.ThermodynamicState

**Minimal variable set that is available as input argument to every medium function**



### Modelica definition

```

replaceable record ThermodynamicState
  "Minimal variable set that is available as input argument to every medium
function"

```

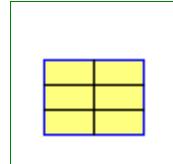
```

  extends Modelica.Icons.Record;
end ThermodynamicState;

```

## Modelica.Media.Interfaces.PartialMedium.BasePropertiesRecord

Variables contained in every instance of BaseProperties



### Modelica definition

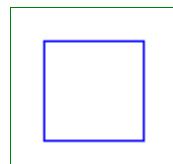
```

replaceable record BasePropertiesRecord
  "Variables contained in every instance of BaseProperties"
  extends Modelica.Icons.Record;
  AbsolutePressure p "Absolute pressure of medium";
  Density d "Density of medium";
  Temperature T "Temperature of medium";
  MassFraction[nX] X(start=reference_X)
    "Mass fractions (= (component mass)/total mass m_i/m)";
  MassFraction[nXi] Xi(start=reference_X[1:nXi])
    "Structurally independent mass fractions";
  SpecificEnthalpy h "Specific enthalpy of medium";
  SpecificInternalEnergy u "Specific internal energy of medium";
  SpecificHeatCapacity R "Gas constant (of mixture if applicable)";
  MolarMass MM "Molar mass (of mixture or single fluid)";
end BasePropertiesRecord;

```

## Modelica.Media.Interfaces.PartialMedium.BaseProperties

Base properties (p, d, T, h, u, R, MM and, if applicable, X) of a medium



### Information

Model **BaseProperties** is a model within package **PartialMedium** and contains the **declarations** of the minimum number of variables that every medium model is supposed to support. A specific medium inherits from model **BaseProperties** and provides the equations for the basic properties. Note, that in package PartialMedium the following constants are defined:

Type	Name	Description
String	mediumName	Unique name of the medium (used to check whether two media in a model are the same)
String	substanceNames	Names of the mixture substances that are treated as independent. If medium consists of a single substance, set substanceNames=fill("",0). If medium consists of n substances, provide either n-1 or n substance names, depending whether mass fractions PartialMedium.BaseProperties.X shall have dimension PartialMedium.nX = n-1 or PartialMedium.nX = n
Boolean	incompressible	= true, if density is constant; otherwise set it to false

In every medium **3+nX equations** have to be defined that provide relations between the following **5+nX variables**, declared in model BaseProperties, where nX is the number of independent mass fractions defined in package PartialMedium:

Variable	Unit	Description
T	K	temperature

p	Pa	absolute pressure
d	kg/m <sup>3</sup>	density
h	J/kg	specific enthalpy
u	J/kg	specific internal energy
X[nX]	kg/kg	independent mass fractions m_i/m

In some components, such as "Ambient", explicit equations for medium variables are provided as "boundary conditions". For example, the "Ambient" component may define a temperature T\_ambient.

## Parameters

Name	Default	Description
standardOrderComponents	true	if true, last element in components is computed from 1-sum(Xi)
<b>Advanced</b>		
preferredMediumStates	false	= true if StateSelect.prefer shall be used for the independent property variables of the medium

## Modelica.Media.Interfaces.PartialMedium.setState\_pTX

Return thermodynamic state as function of p, T and composition X or Xi



### Inputs

Name	Default	Description
p		Pressure [Pa]
T		Temperature [K]
X[:]	reference_X	Mass fractions [kg/kg]

### Outputs

Name	Description
state	

## Modelica.Media.Interfaces.PartialMedium.setState\_phX

Return thermodynamic state as function of p, h and composition X or Xi



### Inputs

Name	Default	Description
p		Pressure [Pa]
h		Specific enthalpy [J/kg]
X[:]	reference_X	Mass fractions [kg/kg]

### Outputs

Name	Description
state	

---

## 686 Modelica.Media.Interfaces.PartialMedium.setState\_psX

---

### Modelica.Media.Interfaces.PartialMedium.setState\_psX

Return thermodynamic state as function of p, s and composition X or Xi



#### Inputs

Name	Default	Description
p		Pressure [Pa]
s		Specific entropy [J/(kg.K)]
X[:]	reference_X	Mass fractions [kg/kg]

#### Outputs

Name	Description
state	

---

### Modelica.Media.Interfaces.PartialMedium.setState\_dTX

Return thermodynamic state as function of d, T and composition X or Xi



#### Inputs

Name	Default	Description
d		density [kg/m3]
T		Temperature [K]
X[:]	reference_X	Mass fractions [kg/kg]

#### Outputs

Name	Description
state	

---

### Modelica.Media.Interfaces.PartialMedium.dynamicViscosity

Return dynamic viscosity



#### Inputs

Name	Default	Description
state		

#### Outputs

Name	Description
eta	Dynamic viscosity [Pa.s]

---

### Modelica.Media.Interfaces.PartialMedium.thermalConductivity

Return thermal conductivity



**Inputs**

Name	Default	Description
state		

**Outputs**

Name	Description
lambda	Thermal conductivity [W/(m.K)]

**Modelica.Media.Interfaces.PartialMedium.prandtlNumber**

Return the Prandtl number

**Inputs**

Name	Default	Description
state		

**Outputs**

Name	Description
Pr	Prandtl number [1]

**Modelica.Media.Interfaces.PartialMedium.pressure**

Return pressure

**Inputs**

Name	Default	Description
state		

**Outputs**

Name	Description
p	Pressure [Pa]

**Modelica.Media.Interfaces.PartialMedium.temperature**

Return temperature

**Inputs**

Name	Default	Description
state		

**Outputs**

Name	Description
T	Temperature [K]

**Modelica.Media.Interfaces.PartialMedium.density**

Return density

**Inputs**

Name	Default	Description
state		

**Outputs**

Name	Description
d	Density [kg/m3]

---

**Modelica.Media.Interfaces.PartialMedium.specificEnthalpy**

Return specific enthalpy

**Inputs**

Name	Default	Description
state		

**Outputs**

Name	Description
h	Specific enthalpy [J/kg]

---

**Modelica.Media.Interfaces.PartialMedium.specificInternalEnergy**

Return specific internal energy

**Inputs**

Name	Default	Description
state		

**Outputs**

Name	Description
u	Specific internal energy [J/kg]

---

**Modelica.Media.Interfaces.PartialMedium.specificEntropy**

Return specific entropy

**Inputs**

Name	Default	Description
state		

**Outputs**

Name	Description
s	Specific entropy [J/(kg.K)]

**Modelica.Media.Interfaces.PartialMedium.specificGibbsEnergy**

Return specific Gibbs energy

**Inputs**

Name	Default	Description
state		

**Outputs**

Name	Description
g	Specific Gibbs energy [J/kg]

**Modelica.Media.Interfaces.PartialMedium.specificHelmholtzEnergy**

Return specific Helmholtz energy

**Inputs**

Name	Default	Description
state		

**Outputs**

Name	Description
f	Specific Helmholtz energy [J/kg]

**Modelica.Media.Interfaces.PartialMedium.specificHeatCapacityCp**

Return specific heat capacity at constant pressure

**Inputs**

Name	Default	Description
state		

**Outputs**

Name	Description
cp	Specific heat capacity at constant pressure [J/(kg.K)]

**Modelica.Media.Interfaces.PartialMedium.heatCapacity\_cp**

alias for deprecated name



---

## 690 Modelica.Media.Interfaces.PartialMedium.heatCapacity\_cp

---

### Inputs

Name	Default	Description
state		

### Outputs

Name	Description
cp	Specific heat capacity at constant pressure [J/(kg.K)]

---



## Modelica.Media.Interfaces.PartialMedium.specificHeatCapacityCv

Return specific heat capacity at constant volume

### Inputs

Name	Default	Description
state		

### Outputs

Name	Description
cv	Specific heat capacity at constant volume [J/(kg.K)]

---



## Modelica.Media.Interfaces.PartialMedium.heatCapacity\_cv

alias for deprecated name

### Inputs

Name	Default	Description
state		

### Outputs

Name	Description
cv	Specific heat capacity at constant volume [J/(kg.K)]

---



## Modelica.Media.Interfaces.PartialMedium.isentropicExponent

Return isentropic exponent

### Inputs

Name	Default	Description
state		

### Outputs

Name	Description
gamma	Isentropic exponent [1]

**Modelica.Media.Interfaces.PartialMedium.isentropicEnthalpy**

Return isentropic enthalpy

**Inputs**

Name	Default	Description
p_downstream		downstream pressure [Pa]
refState		reference state for entropy

**Outputs**

Name	Description
h_is	Isentropic enthalpy [J/kg]

**Modelica.Media.Interfaces.PartialMedium.velocityOfSound**

Return velocity of sound

**Inputs**

Name	Default	Description
state		

**Outputs**

Name	Description
a	Velocity of sound [m/s]

**Modelica.Media.Interfaces.PartialMedium.isobaricExpansionCoefficient**

Return overall the isobaric expansion coefficient beta

**Inputs**

Name	Default	Description
state		

**Outputs**

Name	Description
beta	Isobaric expansion coefficient [1/K]

**Modelica.Media.Interfaces.PartialMedium.beta**

alias for isobaricExpansionCoefficient for user convenience

**Inputs**

Name	Default	Description

## 692 Modelica.Media.Interfaces.PartialMedium.beta

---

state		
-------	--	--

### Outputs

Name	Description
beta	Isobaric expansion coefficient [1/K]

---

## Modelica.Media.Interfaces.PartialMedium.isothermalCompressibility

Return overall the isothermal compressibility factor



### Inputs

Name	Default	Description
state		

### Outputs

Name	Description
kappa	Isothermal compressibility [1/Pa]

---

## Modelica.Media.Interfaces.PartialMedium.kappa

alias of isothermalCompressibility for user convenience



### Inputs

Name	Default	Description
state		

### Outputs

Name	Description
kappa	Isothermal compressibility [1/Pa]

---

## Modelica.Media.Interfaces.PartialMedium.density\_derP\_h

Return density derivative wrt pressure at const specific enthalpy



### Inputs

Name	Default	Description
state		

### Outputs

Name	Description
ddph	Density derivative wrt pressure [s2/m2]

**Modelica.Media.Interfaces.PartialMedium.density\_derh\_p**

Return density derivative wrt specific enthalpy at constant pressure

**Inputs**

Name	Default	Description
state		

**Outputs**

Name	Description
ddhp	Density derivative wrt specific enthalpy [kg.s <sup>2</sup> /m <sup>5</sup> ]

**Modelica.Media.Interfaces.PartialMedium.density\_derT\_p**

Return density derivative wrt pressure at const temperature

**Inputs**

Name	Default	Description
state		

**Outputs**

Name	Description
ddpT	Density derivative wrt pressure [s <sup>2</sup> /m <sup>2</sup> ]

**Modelica.Media.Interfaces.PartialMedium.density\_derT\_p**

Return density derivative wrt temperature at constant pressure

**Inputs**

Name	Default	Description
state		

**Outputs**

Name	Description
ddTp	Density derivative wrt temperature [kg/(m <sup>3</sup> .K)]

**Modelica.Media.Interfaces.PartialMedium.density\_derX**

Return density derivative wrt mass fraction

**Inputs**

Name	Default	Description
state		

---

## 694 Modelica.Media.Interfaces.PartialMedium.density\_derX

---

### Outputs

Name	Description
dddX[nX]	Derivative of density wrt mass fraction [kg/m3]

---



### Modelica.Media.Interfaces.PartialMedium.molarMass

Return the molar mass of the medium

### Inputs

Name	Default	Description
state		

### Outputs

Name	Description
MM	Mixture molar mass [kg/mol]

---



### Modelica.Media.Interfaces.PartialMedium.specificEnthalpy\_pTX

Return specific enthalpy from p, T, and X or Xi

### Inputs

Name	Default	Description
p		Pressure [Pa]
T		Temperature [K]
X[:]	reference_X	Mass fractions [kg/kg]

### Outputs

Name	Description
h	Specific enthalpy [J/kg]

---



### Modelica.Media.Interfaces.PartialMedium.density\_pTX

Return density from p, T, and X or Xi

### Inputs

Name	Default	Description
p		Pressure [Pa]
T		Temperature [K]
X[:]		Mass fractions [kg/kg]

### Outputs

Name	Description
d	Density [kg/m3]

**Modelica.Media.Interfaces.PartialMedium.temperature\_phX**

Return temperature from p, h, and X or Xi

**Inputs**

Name	Default	Description
p		Pressure [Pa]
h		Specific enthalpy [J/kg]
X[:]	reference_X	Mass fractions [kg/kg]

**Outputs**

Name	Description
T	Temperature [K]

**Modelica.Media.Interfaces.PartialMedium.density\_phX**

Return density from p, h, and X or Xi

**Inputs**

Name	Default	Description
p		Pressure [Pa]
h		Specific enthalpy [J/kg]
X[:]	reference_X	Mass fractions [kg/kg]

**Outputs**

Name	Description
d	Density [kg/m3]

**Modelica.Media.Interfaces.PartialMedium.temperature\_psX**

Return temperature from p,s, and X or Xi

**Inputs**

Name	Default	Description
p		Pressure [Pa]
s		Specific entropy [J/(kg.K)]
X[:]	reference_X	Mass fractions [kg/kg]

**Outputs**

Name	Description
T	Temperature [K]

---

## 696 Modelica.Media.Interfaces.PartialMedium.density\_psX

---

### Modelica.Media.Interfaces.PartialMedium.density\_psX



Return density from p, s, and X or Xi

#### Inputs

Name	Default	Description
p		Pressure [Pa]
s		Specific entropy [J/(kg.K)]
X[:]	reference_X	Mass fractions [kg/kg]

#### Outputs

Name	Description
d	Density [kg/m3]

---

### Modelica.Media.Interfaces.PartialMedium.specificEnthalpy\_psX



Return specific enthalpy from p, s, and X or Xi

#### Inputs

Name	Default	Description
p		Pressure [Pa]
s		Specific entropy [J/(kg.K)]
X[:]	reference_X	Mass fractions [kg/kg]

#### Outputs

Name	Description
h	Specific enthalpy [J/kg]

---

### Modelica.Media.Interfaces.PartialMedium.AbsolutePressure

Type for absolute pressure with medium specific attributes

#### Parameters

Name	Default	Description
min	0	
max	1.e8	
start	1.e5	
nominal	1.e5	

---

### Modelica.Media.Interfaces.PartialMedium.Density

Type for density with medium specific attributes

## Parameters

Name	Default	Description
min	0	
max	1.e5	
start	1	
nominal	1	

## Modelica.Media.Interfaces.PartialMedium.DynamicViscosity

Type for dynamic viscosity with medium specific attributes

## Parameters

Name	Default	Description
min	0	
max	1.e8	
start	1.e-3	
nominal	1.e-3	

## Modelica.Media.Interfaces.PartialMedium.EnthalpyFlowRate

Type for enthalpy flow rate with medium specific attributes

## Parameters

Name	Default	Description
min	-1.0e8	
max	1.e8	
nominal	1000.0	

## Modelica.Media.Interfaces.PartialMedium.MassFlowRate

Type for mass flow rate with medium specific attributes

## Parameters

Name	Default	Description
quantity	"MassFlowRate." + mediumName	
min	-1.0e5	
max	1.e5	

## Modelica.Media.Interfaces.PartialMedium.MassFraction

Type for mass fraction with medium specific attributes

### Parameters

Name	Default	Description
quantity	"MassFraction"	
min	0	
max	1	
nominal	0.1	

---

## Modelica.Media.Interfaces.PartialMedium.MoleFraction

Type for mole fraction with medium specific attributes

### Parameters

Name	Default	Description
quantity	"MoleFraction"	
min	0	
max	1	
nominal	0.1	

---

## Modelica.Media.Interfaces.PartialMedium.MolarMass

Type for molar mass with medium specific attributes

### Parameters

Name	Default	Description
min	0.001	
max	0.25	
nominal	0.032	

---

## Modelica.Media.Interfaces.PartialMedium.MolarVolume

Type for molar volume with medium specific attributes

### Parameters

Name	Default	Description
min	1e-6	
max	1.0e6	
nominal	1.0	

---

## Modelica.Media.Interfaces.PartialMedium.ISENTROPICExponent

Type for isentropic exponent with medium specific attributes

**Parameters**

Name	Default	Description
min	1	
max	500000	
start	1.2	
nominal	1.2	

**Modelica.Media.Interfaces.PartialMedium.SpecificEnergy**

Type for specific energy with medium specific attributes

**Parameters**

Name	Default	Description
min	-1.0e8	
max	1.e8	
nominal	1.e6	

**Modelica.Media.Interfaces.PartialMedium.SpecificInternalEnergy**

Type for specific internal energy with medium specific attributes

**Modelica.Media.Interfaces.PartialMedium.SpecificEnthalpy**

Type for specific enthalpy with medium specific attributes

**Parameters**

Name	Default	Description
min	-1.0e8	
max	1.e8	
nominal	1.e6	

**Modelica.Media.Interfaces.PartialMedium.SpecificEntropy**

Type for specific entropy with medium specific attributes

**Parameters**

Name	Default	Description
min	-1.e6	
max	1.e6	
nominal	1.e3	

**Modelica.Media.Interfaces.PartialMedium.SpecificHeatCapacity**

Type for specific heat capacity with medium specific attributes

**Parameters**

Name	Default	Description
min	0	
max	1.e6	
start	1.e3	
nominal	1.e3	

---

**Modelica.Media.Interfaces.PartialMedium.SurfaceTension**

Type for surface tension with medium specific attributes

---

**Modelica.Media.Interfaces.PartialMedium.Temperature**

Type for temperature with medium specific attributes

**Parameters**

Name	Default	Description
min	1	
max	1.e4	
start	300	
nominal	300	

---

**Modelica.Media.Interfaces.PartialMedium.ThermalConductivity**

Type for thermal conductivity with medium specific attributes

**Parameters**

Name	Default	Description
min	0	
max	500	
start	1	
nominal	1	

---

**Modelica.Media.Interfaces.PartialMedium.PrandtlNumber**

Type for Prandtl number with medium specific attributes

**Parameters**

Name	Default	Description
min	1e-3	

---

max	1e5	
nominal	1.0	

**Modelica.Media.Interfaces.PartialMedium.VelocityOfSound**

Type for velocity of sound with medium specific attributes

**Parameters**

Name	Default	Description
min	0	
max	1.e5	
start	1000	
nominal	1000	

**Modelica.Media.Interfaces.PartialMedium.ExtraProperty**

Type for unspecified, mass-specific property transported by flow

**Parameters**

Name	Default	Description
min	0.0	
start	1.0	

**Modelica.Media.Interfaces.PartialMedium.CumulativeExtraProperty**

Type for conserved integral of unspecified, mass specific property

**Parameters**

Name	Default	Description
min	0.0	
start	1.0	

**Modelica.Media.Interfaces.PartialMedium.ExtraPropertyFlowRate**

Type for flow rate of unspecified, mass-specific property

**Modelica.Media.Interfaces.PartialMedium.IsobaricExpansionCoefficient**

Type for isobaric expansion coefficient with medium specific attributes

**Parameters**

Name	Default	Description
unit	"1/K"	

min	1e-8	
max	1.0e8	

---

## Modelica.Media.Interfaces.PartialMedium.DipoleMoment

Type for dipole moment with medium specific attributes

### Parameters

Name	Default	Description
quantity	"ElectricDipoleMoment"	
unit	"debye"	
min	0.0	
max	2.0	

---

## Modelica.Media.Interfaces.PartialMedium.DerDensityByPressure

Type for partial derivative of density with respect to pressure with medium specific attributes

---

## Modelica.Media.Interfaces.PartialMedium.DerDensityByEnthalpy

Type for partial derivative of density with respect to enthalpy with medium specific attributes

---

## Modelica.Media.Interfaces.PartialMedium.DerEnthalpyByPressure

Type for partial derivative of enthalpy with respect to pressure with medium specific attributes

---

## Modelica.Media.Interfaces.PartialMedium.DerDensityByTemperature

Type for partial derivative of density with respect to temperature with medium specific attributes

---

## Modelica.Media.Interfaces.PartialMedium.Choices

Types, constants to define menu choices

### Package Content

Name	Description
<input type="checkbox"/> Init	Type, constants and menu choices to define initialization, as temporary solution until enumerations are available
<input type="checkbox"/> ReferenceEnthalpy	Type, constants and menu choices to define reference enthalpy, as temporary solution until enumerations are available
<input type="checkbox"/> ReferenceEntropy	Type, constants and menu choices to define reference entropy, as temporary solution until enumerations are available
<input type="checkbox"/> pd	Type, constants and menu choices to define whether p or d are known, as temporary solution until enumerations are available

 Th	Type, constants and menu choices to define whether T or h are known, as temporary solution until enumerations are available
 Explicit	Type, constants and menu choices to define the explicitly given state variable inputs

## Modelica.Media.Interfaces.PartialMedium.Choices.Init

Type, constants and menu choices to define initialization, as temporary solution until enumerations are available

### Package Content

Name	Description
NoInit=1	
InitialStates=2	
SteadyState=3	
SteadyMass=4	
Temp	Temporary type with choices for menus (until enumerations are available)

### Types and constants

```
constant Integer NoInit=1;

constant Integer InitialStates=2;

constant Integer SteadyState=3;

constant Integer SteadyMass=4;

type Temp
  "Temporary type with choices for menus (until enumerations are available)"
  extends Integer;
end Temp;
```

## Modelica.Media.Interfaces.PartialMedium.Choices.Init.Temp

Temporary type with choices for menus (until enumerations are available)

### Modelica.Media.Interfaces.PartialMedium.Choices.ReferenceEnthalpy

Type, constants and menu choices to define reference enthalpy, as temporary solution until enumerations are available

### Package Content

Name	Description
ZeroAt0K=1	
ZeroAt25C=2	

---

## 704 Modelica.Media.Interfaces.PartialMedium.Choices.ReferenceEnthalpy

---

UserDefined=3	
Temp	Temporary type with choices for menus (until enumerations are available)

### Types and constants

```
constant Integer ZeroAt0K=1;

constant Integer ZeroAt25C=2;

constant Integer UserDefined=3;

type Temp
"Temporary type with choices for menus (until enumerations are available)"

extends Integer;

end Temp;
```

---

## Modelica.Media.Interfaces.PartialMedium.Choices.ReferenceEnthalpy.Temp

Temporary type with choices for menus (until enumerations are available)

---

## Modelica.Media.Interfaces.PartialMedium.Choices.ReferenceEntropy

Type, constants and menu choices to define reference entropy, as temporary solution until enumerations are available

### Package Content

Name	Description
ZeroAt0K=1	
ZeroAt0C=2	
UserDefined=3	
Temp	Temporary type with choices for menus (until enumerations are available)

### Types and constants

```
constant Integer ZeroAt0K=1;

constant Integer ZeroAt0C=2;

constant Integer UserDefined=3;

type Temp
"Temporary type with choices for menus (until enumerations are available)"

extends Integer;

end Temp;
```

---

**Modelica.Media.Interfaces.PartialMedium.Choices.ReferenceEntropy.Temp****Temporary type with choices for menus (until enumerations are available)****Modelica.Media.Interfaces.PartialMedium.Choices.pd****Type, constants and menu choices to define whether p or d are known, as temporary solution until enumerations are available****Package Content**

Name	Description
default=1	
p_known=2	
d_known=3	
Temp	Temporary type with choices for menus (until enumerations are available)

**Types and constants**

```

constant Integer default=1;

constant Integer p_known=2;

constant Integer d_known=3;

type Temp
  "Temporary type with choices for menus (until enumerations are available)"

  extends Integer;
end Temp;

```

**Modelica.Media.Interfaces.PartialMedium.Choices.pd.Temp****Temporary type with choices for menus (until enumerations are available)****Modelica.Media.Interfaces.PartialMedium.Choices.Th****Type, constants and menu choices to define whether T or h are known, as temporary solution until enumerations are available****Package Content**

Name	Description
default=1	
T_known=2	
h_known=3	
Temp	Temporary type with choices for menus (until enumerations are available)

## Types and constants

```
constant Integer default=1;

constant Integer T_known=2;

constant Integer h_known=3;

type Temp
"Temporary type with choices for menus (until enumerations are available)"

  extends Integer;
end Temp;
```

---

## Modelica.Media.Interfaces.PartialMedium.Choices.Th.Temp

**Temporary type with choices for menus (until enumerations are available)**

---

## Modelica.Media.Interfaces.PartialMedium.Choices.Explicit

**Type, constants and menu choices to define the explicitly given state variable inputs**

---

### Package Content

Name	Description
dT_explicit=0	explicit in density and temperature
ph_explicit=1	explicit in pressure and specific enthalpy
ps_explicit=2	explicit in pressure and specific entropy
pT_explicit=3	explicit in pressure and temperature
Temp	Temporary type with choices for menus (until enumerations are available)

## Types and constants

```
constant Integer dT_explicit=0 "explicit in density and temperature";

constant Integer ph_explicit=1 "explicit in pressure and specific enthalpy";

constant Integer ps_explicit=2 "explicit in pressure and specific entropy";

constant Integer pT_explicit=3 "explicit in pressure and temperature";

type Temp
"Temporary type with choices for menus (until enumerations are available)"

  extends Integer(min=0,max=3);
end Temp;
```

---

## Modelica.Media.Interfaces.PartialMedium.Choices.Explicit.Temp

**Temporary type with choices for menus (until enumerations are available)**

---

## Parameters

Name	Default	Description
min	0	
max	3	

## Modelica.Media.Interfaces.PartialPureSubstance

base class for pure substances of one chemical substance

## Package Content

Name	Description
(f) <code>setState_pT</code>	Return thermodynamic state from p and T
(f) <code>setState_ph</code>	Return thermodynamic state from p and h
(f) <code>setState_ps</code>	Return thermodynamic state from p and s
(f) <code>setState_dT</code>	Return thermodynamic state from d and T
(f) <code>density_ph</code>	Return density from p and h
(f) <code>temperature_ph</code>	Return temperature from p and h
(f) <code>pressure_dT</code>	Return pressure from d and T
(f) <code>specificEnthalpy_dT</code>	Return specific enthalpy from d and T
(f) <code>specificEnthalpy_ps</code>	Return specific enthalpy from p and s
(f) <code>temperature_ps</code>	Return temperature from p and s
(f) <code>density_ps</code>	Return density from p and s
(f) <code>specificEnthalpy_pT</code>	Return specific enthalpy from p and T
(f) <code>density_pT</code>	Return density from p and T
Inherited	
<code>mediumName="unusablePartialMedium"</code>	Name of the medium
<code>substanceNames={mediumName}</code>	Names of the mixture substances. Set <code>substanceNames={mediumName}</code> if only one substance.
<code>extraPropertiesNames=fill("", 0)</code>	Names of the additional (extra) transported properties. Set <code>extraPropertiesNames=fill("",0)</code> if unused
<code>singleState</code>	= true, if u and d are not a function of pressure
<code>reducedX=true</code>	= true if medium contains the equation $\sum(X) = 1.0$ ; set <code>reducedX=true</code> if only one substance (see docu for details)
<code>fixedX=false</code>	= true if medium contains the equation $X = \text{reference\_}X$
<code>reference_p=101325</code>	Reference pressure of Medium: default 1 atmosphere
<code>reference_T=298.15</code>	Reference temperature of Medium: default 25 deg Celsius
<code>reference_X=if nX == 0 then fill(0, nX) else fill(1/nX, nX)</code>	Default mass fractions of medium
<code>p_default=101325</code>	Default value for pressure of medium (for initialization)
<code>T_default=Modelica.SIunits.Conversions.from_degC(20)</code>	Default value for temperature of medium (for initialization)
<code>h_default=specificEnthalpy_pTX(p_default, T_default, X_default)</code>	Default value for specific enthalpy of medium (for initialization)

X_default=reference_X	Default value for mass fractions of medium (for initialization)
nS=size(substanceNames, 1)	Number of substances
nX;if nS == 1 then 0 else nS	Number of mass fractions (= 0, if only one substance)
nXi;if fixedX then 0 else if reducedX then nS - 1 else nX	Number of structurally independent mass fractions (see docu for details)
nC=size(extraPropertiesNames, 1)	Number of extra (outside of standard mass-balance) transported properties
 FluidConstants	critical, triple, molecular and other standard data of fluid
 ThermodynamicState	Minimal variable set that is available as input argument to every medium function
 BasePropertiesRecord	Variables contained in every instance of BaseProperties
 BaseProperties	Base properties (p, d, T, h, u, R, MM and, if applicable, X) of a medium
(f) setState_pTX	Return thermodynamic state as function of p, T and composition X or Xi
(f) setState_phX	Return thermodynamic state as function of p, h and composition X or Xi
(f) setState_psX	Return thermodynamic state as function of p, s and composition X or Xi
(f) setState_dTX	Return thermodynamic state as function of d, T and composition X or Xi
(f) dynamicViscosity	Return dynamic viscosity
(f) thermalConductivity	Return thermal conductivity
(f) prandtlNumber	Return the Prandtl number
(f) pressure	Return pressure
(f) temperature	Return temperature
(f) density	Return density
(f) specificEnthalpy	Return specific enthalpy
(f) specificInternalEnergy	Return specific internal energy
(f) specificEntropy	Return specific entropy
(f) specificGibbsEnergy	Return specific Gibbs energy
(f) specificHelmholtzEnergy	Return specific Helmholtz energy
(f) specificHeatCapacityCp	Return specific heat capacity at constant pressure
(f) heatCapacity_cp	alias for deprecated name
(f) specificHeatCapacityCv	Return specific heat capacity at constant volume
(f) heatCapacity_cv	alias for deprecated name
(f) isentropicExponent	Return isentropic exponent
(f) isentropicEnthalpy	Return isentropic enthalpy
(f) velocityOfSound	Return velocity of sound
(f) isobaricExpansionCoefficient	Return overall the isobaric expansion coefficient beta
(f) beta	alias for isobaricExpansionCoefficient for user convenience
(f) isothermalCompressibility	Return overall the isothermal compressibility factor

 <code>kappa</code>	alias of isothermalCompressibility for user convenience
 <code>density_derP_h</code>	Return density derivative wrt pressure at const specific enthalpy
 <code>density_derH_p</code>	Return density derivative wrt specific enthalpy at constant pressure
 <code>density_derP_T</code>	Return density derivative wrt pressure at const temperature
 <code>density_derT_p</code>	Return density derivative wrt temperature at constant pressure
 <code>density_derX</code>	Return density derivative wrt mass fraction
 <code>molarMass</code>	Return the molar mass of the medium
 <code>specificEnthalpy_pTX</code>	Return specific enthalpy from p, T, and X or Xi
 <code>density_pTX</code>	Return density from p, T, and X or Xi
 <code>temperature_phX</code>	Return temperature from p, h, and X or Xi
 <code>density_phX</code>	Return density from p, h, and X or Xi
 <code>temperature_psX</code>	Return temperature from p,s, and X or Xi
 <code>density_psX</code>	Return density from p, s, and X or Xi
 <code>specificEnthalpy_psX</code>	Return specific enthalpy from p, s, and X or Xi
<code>AbsolutePressure</code>	Type for absolute pressure with medium specific attributes
<code>Density</code>	Type for density with medium specific attributes
<code>DynamicViscosity</code>	Type for dynamic viscosity with medium specific attributes
<code>EnthalpyFlowRate</code>	Type for enthalpy flow rate with medium specific attributes
<code>MassFlowRate</code>	Type for mass flow rate with medium specific attributes
<code>MassFraction</code>	Type for mass fraction with medium specific attributes
<code>MoleFraction</code>	Type for mole fraction with medium specific attributes
<code>MolarMass</code>	Type for molar mass with medium specific attributes
<code>MolarVolume</code>	Type for molar volume with medium specific attributes
<code>IsentropicExponent</code>	Type for isentropic exponent with medium specific attributes
<code>SpecificEnergy</code>	Type for specific energy with medium specific attributes
<code>SpecificInternalEnergy</code>	Type for specific internal energy with medium specific attributes
<code>SpecificEnthalpy</code>	Type for specific enthalpy with medium specific attributes
<code>SpecificEntropy</code>	Type for specific entropy with medium specific attributes
<code>SpecificHeatCapacity</code>	Type for specific heat capacity with medium specific attributes
<code>SurfaceTension</code>	Type for surface tension with medium specific attributes
<code>Temperature</code>	Type for temperature with medium specific attributes
<code>ThermalConductivity</code>	Type for thermal conductivity with medium specific attributes
<code>PrandtlNumber</code>	Type for Prandtl number with medium specific attributes
<code>VelocityOfSound</code>	Type for velocity of sound with medium specific attributes
<code>ExtraProperty</code>	Type for unspecified, mass-specific property transported by flow
<code>CumulativeExtraProperty</code>	Type for conserved integral of unspecified, mass specific property
<code>ExtraPropertyFlowRate</code>	Type for flow rate of unspecified, mass-specific property

## 710 Modelica.Media.Interfaces.PartialPureSubstance

IsobaricExpansionCoefficient	Type for isobaric expansion coefficient with medium specific attributes
DipoleMoment	Type for dipole moment with medium specific attributes
DerDensityByPressure	Type for partial derivative of density with respect to pressure with medium specific attributes
DerDensityByEnthalpy	Type for partial derivative of density with respect to enthalpy with medium specific attributes
DerEnthalpyByPressure	Type for partial derivative of enthalpy with respect to pressure with medium specific attributes
DerDensityByTemperature	Type for partial derivative of density with respect to temperature with medium specific attributes
<input checked="" type="checkbox"/> Choices	Types, constants to define menu choices

### Modelica.Media.Interfaces.PartialPureSubstance.setState\_pT

Return thermodynamic state from p and T



#### Inputs

Name	Default	Description
p		Pressure [Pa]
T		Temperature [K]

#### Outputs

Name	Description
state	

### Modelica.Media.Interfaces.PartialPureSubstance.setState\_ph

Return thermodynamic state from p and h



#### Inputs

Name	Default	Description
p		Pressure [Pa]
h		Specific enthalpy [J/kg]

#### Outputs

Name	Description
state	

### Modelica.Media.Interfaces.PartialPureSubstance.setState\_ps

Return thermodynamic state from p and s



#### Inputs

Name	Default	Description

---

p	Pressure [Pa]
s	Specific entropy [J/(kg.K)]

**Outputs**

Name	Description
state	

**Modelica.Media.Interfaces.PartialPureSubstance.setState\_dT**

Return thermodynamic state from d and T

**Inputs**

Name	Default	Description
d		density [kg/m3]
T		Temperature [K]

**Outputs**

Name	Description
state	

**Modelica.Media.Interfaces.PartialPureSubstance.density\_ph**

Return density from p and h

**Inputs**

Name	Default	Description
p		Pressure [Pa]
h		Specific enthalpy [J/kg]

**Outputs**

Name	Description
d	Density [kg/m3]

**Modelica.Media.Interfaces.PartialPureSubstance.temperature\_ph**

Return temperature from p and h

**Inputs**

Name	Default	Description
p		Pressure [Pa]
h		Specific enthalpy [J/kg]

**Outputs**

Name	Description

## 712 Modelica.Media.Interfaces.PartialPureSubstance.temperature\_ph

---

T	Temperature [K]
---	-----------------

---

### Modelica.Media.Interfaces.PartialPureSubstance.pressure\_dT

Return pressure from d and T



#### Inputs

Name	Default	Description
d		Density [kg/m3]
T		Temperature [K]

#### Outputs

Name	Description
p	Pressure [Pa]

---

### Modelica.Media.Interfaces.PartialPureSubstance.specificEnthalpy\_dT

Return specific enthalpy from d and T



#### Inputs

Name	Default	Description
d		Density [kg/m3]
T		Temperature [K]

#### Outputs

Name	Description
h	specific enthalpy [J/kg]

---

### Modelica.Media.Interfaces.PartialPureSubstance.specificEnthalpy\_ps

Return specific enthalpy from p and s



#### Inputs

Name	Default	Description
p		Pressure [Pa]
s		Specific entropy [J/(kg.K)]

#### Outputs

Name	Description
h	specific enthalpy [J/kg]

---

### Modelica.Media.Interfaces.PartialPureSubstance.temperature\_ps

Return temperature from p and s



## Inputs

Name	Default	Description
p		Pressure [Pa]
s		Specific entropy [J/(kg.K)]

## Outputs

Name	Description
T	Temperature [K]

## Modelica.Media.Interfaces.PartialPureSubstance.density\_ps



Return density from p and s

## Inputs

Name	Default	Description
p		Pressure [Pa]
s		Specific entropy [J/(kg.K)]

## Outputs

Name	Description
d	Density [kg/m3]

## Modelica.Media.Interfaces.PartialPureSubstance.specificEnthalpy\_pT



Return specific enthalpy from p and T

## Inputs

Name	Default	Description
p		Pressure [Pa]
T		Temperature [K]

## Outputs

Name	Description
h	specific enthalpy [J/kg]

## Modelica.Media.Interfaces.PartialPureSubstance.density\_pT



Return density from p and T

## Inputs

Name	Default	Description
p		Pressure [Pa]
T		Temperature [K]

## Outputs

Name	Description
d	Density [kg/m3]

---

## Modelica.Media.Interfaces.PartialLinearFluid

Generic pure liquid model with constant cp, compressibility and thermal expansion coefficients

## Information

### Linear Compressibility Fluid Model

This linear compressibility fluid model is based on the assumptions that:

- The specific heat capacity at constant pressure (cp) is constant
- The isobaric expansion coefficient (beta) is constant
- The isothermal compressibility (kappa) is constant
- Pressure and temperature are used as states

That means that the density is a linear function in temperature and in pressure. In order to define the complete model, a number of constant reference values are needed which are computed at the reference values of the states pressure p and temperature T. The model can be interpreted as a linearization of a full non-linear fluid model (but it is not linear in all thermodynamic coordinates). Reference values are needed for

1. the density (reference\_d),
2. the specific enthalpy (reference\_h),
3. the specific entropy (reference\_s).

Apart from that, a user needs to define the molar mass, MM\_const. Note that it is possible to define a fluid by computing the reference values from a full non-linear fluid model by computing the package constants using the standard functions defined in a fluid package (see example in liquids package).

## Efficiency considerations

One of the main reasons to use a simple, linear fluid model is to achieve high performance in simulations. There are a number of possible compromises and possibilities to improve performance. Some of them can be influenced by a flag. The following rules where used in this model:

- All forward evaluations (using the ThermodynamicState record as input) are exactly following the assumptions above.
- If the flag **constantJacobian** is set to true in the package, all functions that typically appear in thermodynamic jacobians (specificHeatCapacityCv, density\_derh\_p, density\_derh\_T, density\_derT\_p) are evaluated at reference conditions (that means using the reference density) instead of the density of the current pressure and temperature. This makes it possible to evaluate the thermodynamic jacobian at compile time.
- For inverse functions using other inputs than the states (e.g pressure p and specific enthalpy h), the inversion is using the reference state whenever that is necessary to achieve a symbolic inversion.
- If **constantJacobian** is set to false, the above list of functions is computed exactly according to the above list of assumptions

## Authors:

Francesco Casella  
Dipartimento di Elettronica e Informazione  
Politecnico di Milano  
Via Ponzio 34/5  
I-20133 Milano, Italy  
email: [casella@elet.polimi.it](mailto:casella@elet.polimi.it)

and  
 Hubertus Tummescheit  
 Modelon AB  
 Ideon Science Park  
 SE-22730 Lund, Sweden  
 email: [Hubertus.Tummescheit@Modelon.se](mailto:Hubertus.Tummescheit@Modelon.se)

## Package Content

Name	Description
cp_const	Specific heat capacity at constant pressure
beta_const	Thermal expansion coefficient at constant pressure
kappa_const	Isothermal compressibility
MM_const	Molar mass
reference_d	Density in reference conditions
reference_h	Specific enthalpy in reference conditions
reference_s	Specific enthalpy in reference conditions
constantJacobian	if true, entries in thermodynamic Jacobian are constant, taken at reference conditions
 ThermodynamicState	a selection of variables that uniquely defines the thermodynamic state
 BaseProperties	Base properties of medium
(f) setState_pTX	set the thermodynamic state record from p and T (X not needed)
(f) setState_phX	set the thermodynamic state record from p and h (X not needed)
(f) setState_dTX	set the thermodynamic state record from d and T (X not needed)
(f) setState_psX	set the thermodynamic state record from p and s (X not needed)
(f) pressure	Return the pressure from the thermodynamic state
(f) temperature	Return the temperature from the thermodynamic state
(f) density	Return the density from the thermodynamic state
(f) specificEnthalpy	Return the specific enthalpy from the thermodynamic state
(f) specificEntropy	Return the specific entropy from the thermodynamic state
(f) specificInternalEnergy	Return the specific internal energy from the thermodynamic state
(f) specificGibbsEnergy	Return specific Gibbs energy from the thermodynamic state
(f) specificHelmholtzEnergy	Return specific Helmholtz energy from the thermodynamic state
(f) velocityOfSound	Return velocity of sound from the thermodynamic state
(f) isentropicExponent	Return isentropic exponent from the thermodynamic state
(f) isentropicEnthalpy	Return isentropic enthalpy
(f) specificHeatCapacityCp	Return specific heat capacity at constant volume
(f) specificHeatCapacityCv	Return specific heat capacity at constant volume from the thermodynamic state

<code>(f) isothermalCompressibility</code>	Return the iso-thermal compressibility kappa
<code>(f) isobaricExpansionCoefficient</code>	Return the iso-baric expansion coefficient
<code>(f) density_derP_h</code>	Return density derivative wrt pressure at const specific enthalpy
<code>(f) density_derH_p</code>	Return density derivative wrt specific enthalpy at constant pressure
<code>(f) density_derP_T</code>	Return density derivative wrt pressure at const temperature
<code>(f) density_derT_p</code>	Return density derivative wrt temperature at constant pressure
<code>(f) molarMass</code>	Return molar mass
<code>(f) T_ph</code>	Return temperature from pressure and specific enthalpy
<code>(f) T_ps</code>	Return temperature from pressure and specific entropy
<b>Inherited</b>	
<code>(f) setState_pT</code>	Return thermodynamic state from p and T
<code>(f) setState_ph</code>	Return thermodynamic state from p and h
<code>(f) setState_ps</code>	Return thermodynamic state from p and s
<code>(f) setState_dT</code>	Return thermodynamic state from d and T
<code>(f) density_ph</code>	Return density from p and h
<code>(f) temperature_ph</code>	Return temperature from p and h
<code>(f) pressure_dT</code>	Return pressure from d and T
<code>(f) specificEnthalpy_dT</code>	Return specific enthalpy from d and T
<code>(f) specificEnthalpy_ps</code>	Return specific enthalpy from p and s
<code>(f) temperature_ps</code>	Return temperature from p and s
<code>(f) density_ps</code>	Return density from p and s
<code>(f) specificEnthalpy_pT</code>	Return specific enthalpy from p and T
<code>(f) density_pT</code>	Return density from p and T
mediumName="unusablePartialMedium"	Name of the medium
substanceNames={mediumName}	Names of the mixture substances. Set substanceNames={mediumName} if only one substance.
extraPropertiesNames=fill("", 0)	Names of the additional (extra) transported properties. Set extraPropertiesNames=fill("",0) if unused
singleState	= true, if u and d are not a function of pressure
reducedX=true	= true if medium contains the equation sum(X) = 1.0; set reducedX=true if only one substance (see docu for details)
fixedX=false	= true if medium contains the equation X = reference_X
reference_p=101325	Reference pressure of Medium: default 1 atmosphere
reference_T=298.15	Reference temperature of Medium: default 25 deg Celsius
reference_X=if nX == 0 then fill(0, nX) else fill(1/nX, nX)	Default mass fractions of medium
p_default=101325	Default value for pressure of medium (for initialization)
T_default=Modelica.SIunits.Conversions.from_degC(20)	Default value for temperature of medium (for initialization)
h_default=specificEnthalpy_pTX(p_default,	Default value for specific enthalpy of medium (for

T_default, X_default)	initialization)
X_default=reference_X	Default value for mass fractions of medium (for initialization)
nS=size(substanceNames, 1)	Number of substances
nX;if nS == 1 then 0 else nS	Number of mass fractions (= 0, if only one substance)
nXi;if fixedX then 0 else if reducedX then nS - 1 else nX	Number of structurally independent mass fractions (see docu for details)
nC=size(extraPropertiesNames, 1)	Number of extra (outside of standard mass-balance) transported properties
 FluidConstants	critical, triple, molecular and other standard data of fluid
 BasePropertiesRecord	Variables contained in every instance of BaseProperties
(f) dynamicViscosity	Return dynamic viscosity
(f) thermalConductivity	Return thermal conductivity
(f) prandtlNumber	Return the Prandtl number
(f) heatCapacity_cp	alias for deprecated name
(f) heatCapacity_cv	alias for deprecated name
(f) beta	alias for isobaricExpansionCoefficient for user convenience
(f) kappa	alias of isothermalCompressibility for user convenience
(f) density_derX	Return density derivative wrt mass fraction
(f) specificEnthalpy_pTX	Return specific enthalpy from p, T, and X or Xi
(f) density_pTX	Return density from p, T, and X or Xi
(f) temperature_phX	Return temperature from p, h, and X or Xi
(f) density_phX	Return density from p, h, and X or Xi
(f) temperature_psX	Return temperature from p,s, and X or Xi
(f) density_psX	Return density from p, s, and X or Xi
(f) specificEnthalpy_psX	Return specific enthalpy from p, s, and X or Xi
AbsolutePressure	Type for absolute pressure with medium specific attributes
Density	Type for density with medium specific attributes
DynamicViscosity	Type for dynamic viscosity with medium specific attributes
EnthalpyFlowRate	Type for enthalpy flow rate with medium specific attributes
MassFlowRate	Type for mass flow rate with medium specific attributes
MassFraction	Type for mass fraction with medium specific attributes
MoleFraction	Type for mole fraction with medium specific attributes
MolarMass	Type for molar mass with medium specific attributes
MolarVolume	Type for molar volume with medium specific attributes
IsentropicExponent	Type for isentropic exponent with medium specific attributes
SpecificEnergy	Type for specific energy with medium specific attributes
SpecificInternalEnergy	Type for specific internal energy with medium specific attributes
SpecificEnthalpy	Type for specific enthalpy with medium specific attributes
SpecificEntropy	Type for specific entropy with medium specific attributes
SpecificHeatCapacity	Type for specific heat capacity with medium specific attributes

SurfaceTension	Type for surface tension with medium specific attributes
Temperature	Type for temperature with medium specific attributes
ThermalConductivity	Type for thermal conductivity with medium specific attributes
PrandtlNumber	Type for Prandtl number with medium specific attributes
VelocityOfSound	Type for velocity of sound with medium specific attributes
ExtraProperty	Type for unspecified, mass-specific property transported by flow
CumulativeExtraProperty	Type for conserved integral of unspecified, mass specific property
ExtraPropertyFlowRate	Type for flow rate of unspecified, mass-specific property
IsobaricExpansionCoefficient	Type for isobaric expansion coefficient with medium specific attributes
DipoleMoment	Type for dipole moment with medium specific attributes
DerDensityByPressure	Type for partial derivative of density with respect to pressure with medium specific attributes
DerDensityByEnthalpy	Type for partial derivative of density with respect to enthalpy with medium specific attributes
DerEnthalpyByPressure	Type for partial derivative of enthalpy with respect to pressure with medium specific attributes
DerDensityByTemperature	Type for partial derivative of density with respect to temperature with medium specific attributes
 Choices	Types, constants to define menu choices

## Types and constants

```

constant SpecificHeatCapacity cp_const
"Specific heat capacity at constant pressure";

constant IsobaricExpansionCoefficient beta_const
"Thermal expansion coefficient at constant pressure";

constant SI.IsothermalCompressibility kappa_const
"Isothermal compressibility";

constant MolarMass MM_const "Molar mass";

constant Density reference_d "Density in reference conditions";

constant SpecificEnthalpy reference_h
"Specific enthalpy in reference conditions";

constant SpecificEntropy reference_s
"Specific entropy in reference conditions";

constant Boolean constantJacobian
"if true, entries in thermodynamic Jacobian are constant, taken at reference
conditions";

```

**Modelica.Media.Interfaces.PartialLinearFluid.ThermodynamicState**

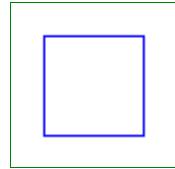
a selection of variables that uniquely defines the thermodynamic state

**Modelica definition**

```
redeclare replaceable record ThermodynamicState
  "a selection of variables that uniquely defines the thermodynamic state"
  AbsolutePressure p "Absolute pressure of medium";
  Temperature T "Temperature of medium";
end ThermodynamicState;
```

**Modelica.Media.Interfaces.PartialLinearFluid.BaseProperties**

Base properties of medium

**Parameters**

Name	Default	Description
standardOrderComponents	true	if true, last element in components is computed from 1-sum(Xi)
<b>Advanced</b>		
preferredMediumStates	false	= true if StateSelect.prefer shall be used for the independent property variables of the medium

**Modelica.Media.Interfaces.PartialLinearFluid.setState\_pTX**

set the thermodynamic state record from p and T (X not needed)

**Inputs**

Name	Default	Description
p		Pressure [Pa]
T		Temperature [K]
X[:]	reference_X	Mass fractions [kg/kg]

**Outputs**

Name	Description
state	

**Modelica.Media.Interfaces.PartialLinearFluid.setState\_phX**

set the thermodynamic state record from p and h (X not needed)

**Inputs**

Name	Default	Description
p		Pressure [Pa]
h		Specific enthalpy [J/kg]

---

## 720 Modelica.Media.Interfaces.PartialLinearFluid.setState\_phX

---

X[:]	reference_X	Mass fractions [kg/kg]
------	-------------	------------------------

### Outputs

Name	Description
state	

---

## Modelica.Media.Interfaces.PartialLinearFluid.setState\_dTX

set the thermodynamic state record from d and T (X not needed)



### Inputs

Name	Default	Description
d		density [kg/m <sup>3</sup> ]
T		Temperature [K]
X[:]	reference_X	Mass fractions [kg/kg]

### Outputs

Name	Description
state	

---

## Modelica.Media.Interfaces.PartialLinearFluid.setState\_psX

set the thermodynamic state record from p and s (X not needed)



### Inputs

Name	Default	Description
p		Pressure [Pa]
s		Specific entropy [J/(kg.K)]
X[:]	reference_X	Mass fractions [kg/kg]

### Outputs

Name	Description
state	

---

## Modelica.Media.Interfaces.PartialLinearFluid.pressure

Return the pressure from the thermodynamic state



### Inputs

Name	Default	Description
state		

### Outputs

Name	Description

---

p	Pressure [Pa]
---	---------------

**Modelica.Media.Interfaces.PartialLinearFluid.temperature**

Return the temperature from the thermodynamic state

**Inputs**

Name	Default	Description
state		

**Outputs**

Name	Description
T	Temperature [K]

**Modelica.Media.Interfaces.PartialLinearFluid.density**

Return the density from the thermodynamic state

**Inputs**

Name	Default	Description
state		

**Outputs**

Name	Description
d	Density [kg/m3]

**Modelica.Media.Interfaces.PartialLinearFluid.specificEnthalpy**

Return the specific enthalpy from the thermodynamic state

**Inputs**

Name	Default	Description
state		

**Outputs**

Name	Description
h	Specific enthalpy [J/kg]

**Modelica.Media.Interfaces.PartialLinearFluid.specificEntropy**

Return the specific entropy from the thermodynamic state

## 722 Modelica.Media.Interfaces.PartialLinearFluid.specificEntropy

---

### Inputs

Name	Default	Description
state		

### Outputs

Name	Description
s	Specific entropy [J/(kg.K)]



## Modelica.Media.Interfaces.PartialLinearFluid.specificInternalEnergy

Return the specific internal energy from the thermodynamic state

### Inputs

Name	Default	Description
state		

### Outputs

Name	Description
u	Specific internal energy [J/kg]



## Modelica.Media.Interfaces.PartialLinearFluid.specificGibbsEnergy

Return specific Gibbs energy from the thermodynamic state

### Inputs

Name	Default	Description
state		

### Outputs

Name	Description
g	Specific Gibbs energy [J/kg]



## Modelica.Media.Interfaces.PartialLinearFluid.specificHelmholtzEnergy

Return specific Helmholtz energy from the thermodynamic state

### Inputs

Name	Default	Description
state		

### Outputs

Name	Description
f	Specific Helmholtz energy [J/kg]

**Modelica.Media.Interfaces.PartialLinearFluid.velocityOfSound**

Return velocity of sound from the thermodynamic state

**Inputs**

Name	Default	Description
state		

**Outputs**

Name	Description
a	Velocity of sound [m/s]

**Modelica.Media.Interfaces.PartialLinearFluid.isentropicExponent**

Return isentropic exponent from the thermodynamic state

**Inputs**

Name	Default	Description
state		

**Outputs**

Name	Description
gamma	Isentropic exponent [1]

**Modelica.Media.Interfaces.PartialLinearFluid.isentropicEnthalpy**

Return isentropic enthalpy

**Information**

A minor approximation is used: the reference density is used instead of the real one, which would require a numeric solution.

**Inputs**

Name	Default	Description
p_downstream		downstream pressure [Pa]
refState		reference state for entropy

**Outputs**

Name	Description
h_is	Isentropic enthalpy [J/kg]

**Modelica.Media.Interfaces.PartialLinearFluid.specificHeatCapacityCp**

Return specific heat capacity at constant volume

**Inputs**

Name	Default	Description
state		

**Outputs**

Name	Description
cp	Specific heat capacity at constant pressure [J/(kg.K)]

---

**Modelica.Media.Interfaces.PartialLinearFluid.specificHeatCapacityCv**

Return specific heat capacity at constant volume from the thermodynamic state

**Inputs**

Name	Default	Description
state		

**Outputs**

Name	Description
cv	Specific heat capacity at constant volume [J/(kg.K)]

---

**Modelica.Media.Interfaces.PartialLinearFluid.isoThermalCompressibility**

Return the iso-thermal compressibility kappa

**Inputs**

Name	Default	Description
state		

**Outputs**

Name	Description
kappa	Isothermal compressibility [1/Pa]

---

**Modelica.Media.Interfaces.PartialLinearFluid.isobaricExpansionCoefficient**

Return the iso-baric expansion coefficient

**Inputs**

Name	Default	Description
state		

**Outputs**

Name	Description
beta	Isobaric expansion coefficient [1/K]

**Modelica.Media.Interfaces.PartialLinearFluid.density\_derP\_h**

Return density derivative wrt pressure at const specific enthalpy

**Inputs**

Name	Default	Description
state		

**Outputs**

Name	Description
ddph	Density derivative wrt pressure [s2/m2]

**Modelica.Media.Interfaces.PartialLinearFluid.density\_derH\_p**

Return density derivative wrt specific enthalpy at constant pressure

**Inputs**

Name	Default	Description
state		

**Outputs**

Name	Description
ddhp	Density derivative wrt specific enthalpy [kg.s2/m5]

**Modelica.Media.Interfaces.PartialLinearFluid.density\_derP\_T**

Return density derivative wrt pressure at const temperature

**Inputs**

Name	Default	Description
state		

**Outputs**

Name	Description
ddpT	Density derivative wrt pressure [s2/m2]

**Modelica.Media.Interfaces.PartialLinearFluid.density\_derT\_p**

Return density derivative wrt temperature at constant pressure



---

## 726 Modelica.Media.Interfaces.PartialLinearFluid.density\_derT\_p

---

### Inputs

Name	Default	Description
state		

### Outputs

Name	Description
ddTp	Density derivative wrt temperature [kg/(m <sup>3</sup> .K)]

---

## Modelica.Media.Interfaces.PartialLinearFluid.molarMass



Return molar mass

### Inputs

Name	Default	Description
state		

### Outputs

Name	Description
MM	Mixture molar mass [kg/mol]

---

## Modelica.Media.Interfaces.PartialLinearFluid.T\_ph

Return temperature from pressure and specific enthalpy

### Inputs

Name	Default	Description
h		Specific enthalpy [J/kg]
p		pressure [Pa]

### Outputs

Name	Description
T	Temperature [K]

---

## Modelica.Media.Interfaces.PartialLinearFluid.T\_ps

Return temperature from pressure and specific entropy

### Inputs

Name	Default	Description
p		Pressure [Pa]
s		Specific entropy [J/(kg.K)]

## Outputs

Name	Description
T	Temperature [K]

## Modelica.Media.Interfaces.PartialMixtureMedium

base class for pure substances of several chemical substances

### Package Content

Name	Description
 ThermodynamicState	thermodynamic state variables
 FluidConstants	extended fluid constants
fluidConstants	constant data for the fluid
 gasConstant	Return the gas constant of the mixture (also for liquids)
 moleToMassFractions	Return mass fractions X from mole fractions
 massToMoleFractions	Return mole fractions from mass fractions X
Inherited	
mediumName="unusablePartialMedium"	Name of the medium
substanceNames={mediumName}	Names of the mixture substances. Set substanceNames={mediumName} if only one substance.
extraPropertiesNames=fill("", 0)	Names of the additional (extra) transported properties. Set extraPropertiesNames=fill("",0) if unused
singleState	= true, if u and d are not a function of pressure
reducedX=true	= true if medium contains the equation sum(X) = 1.0; set reducedX=true if only one substance (see docu for details)
fixedX=false	= true if medium contains the equation X = reference_X
reference_p=101325	Reference pressure of Medium: default 1 atmosphere
reference_T=298.15	Reference temperature of Medium: default 25 deg Celsius
reference_X=if nX == 0 then fill(0, nX) else fill(1/nX, nX)	Default mass fractions of medium
p_default=101325	Default value for pressure of medium (for initialization)
T_default=Modelica.Sunits.Conversions.from_degC(20)	Default value for temperature of medium (for initialization)
h_default=specificEnthalpy_pTX(p_default, T_default, X_default)	Default value for specific enthalpy of medium (for initialization)
X_default=reference_X	Default value for mass fractions of medium (for initialization)
nS=size(substanceNames, 1)	Number of substances
nX=if nS == 1 then 0 else nS	Number of mass fractions (= 0, if only one substance)
nXi=if fixedX then 0 else if reducedX then nS - 1 else nX	Number of structurally independent mass fractions (see docu for details)
nC=size(extraPropertiesNames, 1)	Number of extra (outside of standard mass-balance) transported properties
 BasePropertiesRecord	Variables contained in every instance of BaseProperties
 BaseProperties	Base properties (p, d, T, h, u, R, MM and, if applicable, X) of a medium

(f) <code>setState_pTX</code>	Return thermodynamic state as function of p, T and composition X or Xi
(f) <code>setState_phX</code>	Return thermodynamic state as function of p, h and composition X or Xi
(f) <code>setState_psX</code>	Return thermodynamic state as function of p, s and composition X or Xi
(f) <code>setState_dTX</code>	Return thermodynamic state as function of d, T and composition X or Xi
(f) <code>dynamicViscosity</code>	Return dynamic viscosity
(f) <code>thermalConductivity</code>	Return thermal conductivity
(f) <code>prandtlNumber</code>	Return the Prandtl number
(f) <code>pressure</code>	Return pressure
(f) <code>temperature</code>	Return temperature
(f) <code>density</code>	Return density
(f) <code>specificEnthalpy</code>	Return specific enthalpy
(f) <code>specificInternalEnergy</code>	Return specific internal energy
(f) <code>specificEntropy</code>	Return specific entropy
(f) <code>specificGibbsEnergy</code>	Return specific Gibbs energy
(f) <code>specificHelmholtzEnergy</code>	Return specific Helmholtz energy
(f) <code>specificHeatCapacityCp</code>	Return specific heat capacity at constant pressure
(f) <code>heatCapacity_cp</code>	alias for deprecated name
(f) <code>specificHeatCapacityCv</code>	Return specific heat capacity at constant volume
(f) <code>heatCapacity_cv</code>	alias for deprecated name
(f) <code>isentropicExponent</code>	Return isentropic exponent
(f) <code>isentropicEnthalpy</code>	Return isentropic enthalpy
(f) <code>velocityOfSound</code>	Return velocity of sound
(f) <code>isobaricExpansionCoefficient</code>	Return overall the isobaric expansion coefficient beta
(f) <code>beta</code>	alias for isobaricExpansionCoefficient for user convenience
(f) <code>isothermalCompressibility</code>	Return overall the isothermal compressibility factor
(f) <code>kappa</code>	alias of isothermalCompressibility for user convenience
(f) <code>density_derP_h</code>	Return density derivative wrt pressure at const specific enthalpy
(f) <code>density_derH_p</code>	Return density derivative wrt specific enthalpy at constant pressure
(f) <code>density_derP_T</code>	Return density derivative wrt pressure at const temperature
(f) <code>density_derT_p</code>	Return density derivative wrt temperature at constant pressure
(f) <code>density_derX</code>	Return density derivative wrt mass fraction
(f) <code>molarMass</code>	Return the molar mass of the medium
(f) <code>specificEnthalpy_pTX</code>	Return specific enthalpy from p, T, and X or Xi
(f) <code>density_pTX</code>	Return density from p, T, and X or Xi

 <a href="#">temperature_phX</a>	Return temperature from p, h, and X or Xi
 <a href="#">density_phX</a>	Return density from p, h, and X or Xi
 <a href="#">temperature_psX</a>	Return temperature from p,s, and X or Xi
 <a href="#">density_psX</a>	Return density from p, s, and X or Xi
 <a href="#">specificEnthalpy_psX</a>	Return specific enthalpy from p, s, and X or Xi
<a href="#">AbsolutePressure</a>	Type for absolute pressure with medium specific attributes
<a href="#">Density</a>	Type for density with medium specific attributes
<a href="#">DynamicViscosity</a>	Type for dynamic viscosity with medium specific attributes
<a href="#">EnthalpyFlowRate</a>	Type for enthalpy flow rate with medium specific attributes
<a href="#">MassFlowRate</a>	Type for mass flow rate with medium specific attributes
<a href="#">MassFraction</a>	Type for mass fraction with medium specific attributes
<a href="#">MoleFraction</a>	Type for mole fraction with medium specific attributes
<a href="#">MolarMass</a>	Type for molar mass with medium specific attributes
<a href="#">MolarVolume</a>	Type for molar volume with medium specific attributes
<a href="#">IsentropicExponent</a>	Type for isentropic exponent with medium specific attributes
<a href="#">SpecificEnergy</a>	Type for specific energy with medium specific attributes
<a href="#">SpecificInternalEnergy</a>	Type for specific internal energy with medium specific attributes
<a href="#">SpecificEnthalpy</a>	Type for specific enthalpy with medium specific attributes
<a href="#">SpecificEntropy</a>	Type for specific entropy with medium specific attributes
<a href="#">SpecificHeatCapacity</a>	Type for specific heat capacity with medium specific attributes
<a href="#">SurfaceTension</a>	Type for surface tension with medium specific attributes
<a href="#">Temperature</a>	Type for temperature with medium specific attributes
<a href="#">ThermalConductivity</a>	Type for thermal conductivity with medium specific attributes
<a href="#">PrandtlNumber</a>	Type for Prandtl number with medium specific attributes
<a href="#">VelocityOfSound</a>	Type for velocity of sound with medium specific attributes
<a href="#">ExtraProperty</a>	Type for unspecified, mass-specific property transported by flow
<a href="#">CumulativeExtraProperty</a>	Type for conserved integral of unspecified, mass specific property
<a href="#">ExtraPropertyFlowRate</a>	Type for flow rate of unspecified, mass-specific property
<a href="#">IsobaricExpansionCoefficient</a>	Type for isobaric expansion coefficient with medium specific attributes
<a href="#">DipoleMoment</a>	Type for dipole moment with medium specific attributes
<a href="#">DerDensityByPressure</a>	Type for partial derivative of density with respect to pressure with medium specific attributes
<a href="#">DerDensityByEnthalpy</a>	Type for partial derivative of density with respect to enthalpy with medium specific attributes
<a href="#">DerEnthalpyByPressure</a>	Type for partial derivative of enthalpy with respect to pressure with medium specific attributes
<a href="#">DerDensityByTemperature</a>	Type for partial derivative of density with respect to temperature with medium specific attributes
 <a href="#">Choices</a>	Types, constants to define menu choices

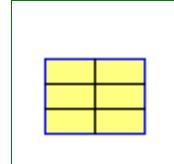
## Types and constants

```
constant FluidConstants[nS] fluidConstants "constant data for the fluid";
```

---

## Modelica.Media.Interfaces.PartialMixtureMedium.ThermodynamicState

### thermodynamic state variables



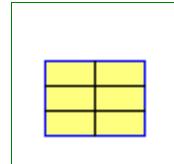
#### Modelica definition

```
redeclare replaceable record extends ThermodynamicState
  "thermodynamic state variables"
  AbsolutePressure p "Absolute pressure of medium";
  Temperature T "Temperature of medium";
  MassFraction X[nX] "Mass fractions (= (component mass)/total mass m_i/m)";
end ThermodynamicState;
```

---

## Modelica.Media.Interfaces.PartialMixtureMedium.FluidConstants

### extended fluid constants



#### Modelica definition

```
redeclare replaceable record extends FluidConstants
  "extended fluid constants"
  Temperature criticalTemperature "critical temperature";
  AbsolutePressure criticalPressure "critical pressure";
  MolarVolume criticalMolarVolume "critical molar Volume";
  Real acentricFactor "Pitzer acentric factor";
  Temperature triplePointTemperature "triple point temperature";
  AbsolutePressure triplePointPressure "triple point pressure";
  Temperature meltingPoint "melting point at 101325 Pa";
  Temperature normalBoilingPoint "normal boiling point (at 101325 Pa)";
  DipoleMoment dipoleMoment
    "dipole moment of molecule in Debye (1 debye = 3.33564e10-30 C.m)";
  Boolean hasIdealGasHeatCapacity=false
    "true if ideal gas heat capacity is available";
  Boolean hasCriticalData=false "true if critical data are known";
  Boolean hasDipoleMoment=false "true if a dipole moment known";
  Boolean hasFundamentalEquation=false "true if a fundamental equation";
  Boolean hasLiquidHeatCapacity=false
    "true if liquid heat capacity is available";
  Boolean hasSolidHeatCapacity=false "true if solid heat capacity is available";
  Boolean hasAccurateViscosityData=false
    "true if accurate data for a viscosity function is available";
  Boolean hasAccurateConductivityData=false
    "true if accurate data for thermal conductivity is available";
  Boolean hasVapourPressureCurve=false
    "true if vapour pressure data, e.g. Antoine coefficients are known";
  Boolean hasAcentricFactor=false "true if Pitzer acentric factor is known";
  SpecificEnthalpy HCRIT0=0.0
    "Critical specific enthalpy of the fundamental equation";
  SpecificEntropy SCRIT0=0.0
```

---

---

```

    "Critical specific entropy of the fundamental equation";
SpecificEnthalpy deltah=0.0
    "Difference between specific enthalpy model (h_m) and f.eq. (h_f) (h_m -
h_f)";
SpecificEntropy deltas=0.0
    "Difference between specific enthalpy model (s_m) and f.eq. (s_f) (s_m -
s_f)";
end FluidConstants;

```

---

**Modelica.Media.Interfaces.PartialMixtureMedium.gasConstant**

Return the gas constant of the mixture (also for liquids)

**Inputs**

Name	Default	Description
state		thermodynamic state

**Outputs**

Name	Description
R	mixture gas constant [J/(kg.K)]

**Modelica.Media.Interfaces.PartialMixtureMedium.moleToMassFractions**

Return mass fractions X from mole fractions

**Inputs**

Name	Default	Description
moleFractions[:]		Mole fractions of mixture [1]
MMX[:]		molar masses of components [kg/mol]

**Outputs**

Name	Description
X[size(moleFractions, 1)]	Mass fractions of gas mixture [1]

**Modelica.Media.Interfaces.PartialMixtureMedium.massToMoleFractions**

Return mole fractions from mass fractions X

**Inputs**

Name	Default	Description
X[:]		Mass fractions of mixture [1]
MMX[:]		molar masses of components [kg/mol]

## Outputs

Name	Description
moleFractions[size(X, 1)]	Mole fractions of gas mixture [1]

## Modelica.Media.Interfaces.PartialCondensingGases

Base class for mixtures of condensing and non-condensing gases

### Package Content

Name	Description
(f) saturationPressure	Return saturation pressure of condensing fluid
(f) enthalpyOfVaporization	Return vaporization enthalpy of condensing fluid
(f) enthalpyOfLiquid	Return liquid enthalpy of condensing fluid
(f) enthalpyOfGas	Return enthalpy of non-condensing gas mixture
(f) enthalpyOfCondensingGas	Return enthalpy of condensing gas (most often steam)
Inherited	
ThermodynamicState	thermodynamic state variables
FluidConstants	extended fluid constants
fluidConstants	constant data for the fluid
(f) gasConstant	Return the gas constant of the mixture (also for liquids)
(f) moleToMassFractions	Return mass fractions X from mole fractions
(f) massToMoleFractions	Return mole fractions from mass fractions X
mediumName="unusablePartialMedium"	Name of the medium
substanceNames={mediumName}	Names of the mixture substances. Set substanceNames={mediumName} if only one substance.
extraPropertiesNames=fill("", 0)	Names of the additional (extra) transported properties. Set extraPropertiesNames=fill("",0) if unused
singleState	= true, if u and d are not a function of pressure
reducedX=true	= true if medium contains the equation sum(X) = 1.0; set reducedX=true if only one substance (see docu for details)
fixedX=false	= true if medium contains the equation X = reference_X
reference_p=101325	Reference pressure of Medium: default 1 atmosphere
reference_T=298.15	Reference temperature of Medium: default 25 deg Celsius
reference_X=if nX == 0 then fill(0, nX) else fill(1/nX, nX)	Default mass fractions of medium
p_default=101325	Default value for pressure of medium (for initialization)
T_default=Modelica.Slunits.Conversions.from_degC(20)	Default value for temperature of medium (for initialization)
h_default=specificEnthalpy_pTX(p_default, T_default, X_default)	Default value for specific enthalpy of medium (for initialization)
X_default=reference_X	Default value for mass fractions of medium (for initialization)
nS=size(substanceNames, 1)	Number of substances
nX=if nS == 1 then 0 else nS	Number of mass fractions (= 0, if only one substance)

$nXi = \text{if fixedX then } 0 \text{ else if reducedX then } nS - 1 \text{ else } nX$	Number of structurally independent mass fractions (see docu for details)
$nC = \text{size}(\text{extraPropertiesNames}, 1)$	Number of extra (outside of standard mass-balance) transported properties
 BasePropertiesRecord	Variables contained in every instance of BaseProperties
 BaseProperties	Base properties (p, d, T, h, u, R, MM and, if applicable, X) of a medium
(f) setState_pTX	Return thermodynamic state as function of p, T and composition X or Xi
(f) setState_phX	Return thermodynamic state as function of p, h and composition X or Xi
(f) setState_psX	Return thermodynamic state as function of p, s and composition X or Xi
(f) setState_dTX	Return thermodynamic state as function of d, T and composition X or Xi
(f) dynamicViscosity	Return dynamic viscosity
(f) thermalConductivity	Return thermal conductivity
(f) prandtlNumber	Return the Prandtl number
(f) pressure	Return pressure
(f) temperature	Return temperature
(f) density	Return density
(f) specificEnthalpy	Return specific enthalpy
(f) specificInternalEnergy	Return specific internal energy
(f) specificEntropy	Return specific entropy
(f) specificGibbsEnergy	Return specific Gibbs energy
(f) specificHelmholtzEnergy	Return specific Helmholtz energy
(f) specificHeatCapacityCp	Return specific heat capacity at constant pressure
(f) heatCapacity_cp	alias for deprecated name
(f) specificHeatCapacityCv	Return specific heat capacity at constant volume
(f) heatCapacity_cv	alias for deprecated name
(f) isentropicExponent	Return isentropic exponent
(f) isentropicEnthalpy	Return isentropic enthalpy
(f) velocityOfSound	Return velocity of sound
(f) isobaricExpansionCoefficient	Return overall the isobaric expansion coefficient beta
(f) beta	alias for isobaricExpansionCoefficient for user convenience
(f) isothermalCompressibility	Return overall the isothermal compressibility factor
(f) kappa	alias of isothermalCompressibility for user convenience
(f) density_derP_h	Return density derivative wrt pressure at const specific enthalpy
(f) density_derH_p	Return density derivative wrt specific enthalpy at constant pressure
(f) density_derP_T	Return density derivative wrt pressure at const temperature

(f) <code>density_derT_p</code>	Return density derivative wrt temperature at constant pressure
(f) <code>density_derX</code>	Return density derivative wrt mass fraction
(f) <code>molarMass</code>	Return the molar mass of the medium
(f) <code>specificEnthalpy_pTX</code>	Return specific enthalpy from p, T, and X or Xi
(f) <code>density_pTX</code>	Return density from p, T, and X or Xi
(f) <code>temperature_phX</code>	Return temperature from p, h, and X or Xi
(f) <code>density_phX</code>	Return density from p, h, and X or Xi
(f) <code>temperature_psX</code>	Return temperature from p,s, and X or Xi
(f) <code>density_psX</code>	Return density from p, s, and X or Xi
(f) <code>specificEnthalpy_psX</code>	Return specific enthalpy from p, s, and X or Xi
<code>AbsolutePressure</code>	Type for absolute pressure with medium specific attributes
<code>Density</code>	Type for density with medium specific attributes
<code>DynamicViscosity</code>	Type for dynamic viscosity with medium specific attributes
<code>EnthalpyFlowRate</code>	Type for enthalpy flow rate with medium specific attributes
<code>MassFlowRate</code>	Type for mass flow rate with medium specific attributes
<code>MassFraction</code>	Type for mass fraction with medium specific attributes
<code>MoleFraction</code>	Type for mole fraction with medium specific attributes
<code>MolarMass</code>	Type for molar mass with medium specific attributes
<code>MolarVolume</code>	Type for molar volume with medium specific attributes
<code>IsentropicExponent</code>	Type for isentropic exponent with medium specific attributes
<code>SpecificEnergy</code>	Type for specific energy with medium specific attributes
<code>SpecificInternalEnergy</code>	Type for specific internal energy with medium specific attributes
<code>SpecificEnthalpy</code>	Type for specific enthalpy with medium specific attributes
<code>SpecificEntropy</code>	Type for specific entropy with medium specific attributes
<code>SpecificHeatCapacity</code>	Type for specific heat capacity with medium specific attributes
<code>SurfaceTension</code>	Type for surface tension with medium specific attributes
<code>Temperature</code>	Type for temperature with medium specific attributes
<code>ThermalConductivity</code>	Type for thermal conductivity with medium specific attributes
<code>PrandtlNumber</code>	Type for Prandtl number with medium specific attributes
<code>VelocityOfSound</code>	Type for velocity of sound with medium specific attributes
<code>ExtraProperty</code>	Type for unspecified, mass-specific property transported by flow
<code>CumulativeExtraProperty</code>	Type for conserved integral of unspecified, mass specific property
<code>ExtraPropertyFlowRate</code>	Type for flow rate of unspecified, mass-specific property
<code>IsobaricExpansionCoefficient</code>	Type for isobaric expansion coefficient with medium specific attributes
<code>DipoleMoment</code>	Type for dipole moment with medium specific attributes
<code>DerDensityByPressure</code>	Type for partial derivative of density with respect to pressure with medium specific attributes
<code>DerDensityByEnthalpy</code>	Type for partial derivative of density with respect to enthalpy with medium specific attributes

DerEnthalpyByPressure	Type for partial derivative of enthalpy with respect to pressure with medium specific attributes
DerDensityByTemperature	Type for partial derivative of density with respect to temperature with medium specific attributes
Choices	Types, constants to define menu choices

**Modelica.Media.Interfaces.PartialCondensingGases.saturationPressure**

Return saturation pressure of condensing fluid

**Inputs**

Name	Default	Description
Tsat		saturation temperature [K]

**Outputs**

Name	Description
psat	saturation pressure [Pa]

**Modelica.Media.Interfaces.PartialCondensingGases.enthalpyOfVaporization**

Return vaporization enthalpy of condensing fluid

**Inputs**

Name	Default	Description
T		temperature [K]

**Outputs**

Name	Description
r0	vaporization enthalpy [J/kg]

**Modelica.Media.Interfaces.PartialCondensingGases.enthalpyOfLiquid**

Return liquid enthalpy of condensing fluid

**Inputs**

Name	Default	Description
T		temperature [K]

**Outputs**

Name	Description
h	liquid enthalpy [J/kg]

**Modelica.Media.Interfaces.PartialCondensingGases.enthalpyOfGas**

Return enthalpy of non-condensing gas mixture

**Inputs**

Name	Default	Description
T		temperature [K]
X[:]		vector of mass fractions [kg/kg]

**Outputs**

Name	Description
h	liquid enthalpy [J/kg]

**Modelica.Media.Interfaces.PartialCondensingGases.enthalpyOfCondensingGas**

Return enthalpy of condensing gas (most often steam)

**Inputs**

Name	Default	Description
T		temperature [K]

**Outputs**

Name	Description
h	liquid enthalpy [J/kg]

**Modelica.Media.Interfaces.PartialTwoPhaseMedium****Package Content**

Name	Description
smoothModel	true if the (derived) model should not generate state events
onePhase	true if the (derived) model should never be called with two-phase inputs
FluidLimits	validity limits for fluid model
FluidConstants	extended fluid constants
fluidConstants	constant data for the fluid
ThermodynamicState	Thermodynamic state of two phase medium
SaturationProperties	Saturation properties of two phase medium
FixedPhase	phase of the fluid: 1 for 1-phase, 2 for two-phase, 0 for not known, e.g. interactive use
BaseProperties	Base properties (p, d, T, h, u, R, MM, sat) of two phase medium
setDewState	Return the thermodynamic state on the dew line
setBubbleState	Return the thermodynamic state on the bubble line

(f) <code>setState_dTX</code>	Return thermodynamic state as function of d, T and composition X or Xi
(f) <code>setState_phX</code>	Return thermodynamic state as function of p, h and composition X or Xi
(f) <code>setState_psX</code>	Return thermodynamic state as function of p, s and composition X or Xi
(f) <code>setState_pTX</code>	Return thermodynamic state as function of p, T and composition X or Xi
(f) <code>setSat_T</code>	Return saturation property record from temperature
(f) <code>setSat_p</code>	Return saturation property record from pressure
(f) <code>bubbleEnthalpy</code>	Return bubble point specific enthalpy
(f) <code>dewEnthalpy</code>	Return dew point specific enthalpy
(f) <code>bubbleEntropy</code>	Return bubble point specific entropy
(f) <code>dewEntropy</code>	Return dew point specific entropy
(f) <code>bubbleDensity</code>	Return bubble point density
(f) <code>dewDensity</code>	Return dew point density
(f) <code>saturationPressure</code>	Return saturation pressure
(f) <code>saturationTemperature</code>	Return saturation temperature
(f) <code>saturationPressure_sat</code>	Return saturation temperature
(f) <code>saturationTemperature_sat</code>	Return saturation temperature
(f) <code>saturationTemperature_derP</code>	Return derivative of saturation temperature w.r.t. pressure
(f) <code>saturationTemperature_derP_sat</code>	Return derivative of saturation temperature w.r.t. pressure
(f) <code>surfaceTension</code>	Return surface tension sigma in the two phase region
(f) <code>molarMass</code>	Return the molar mass of the medium
(f) <code>dBubbleDensity_dPressure</code>	Return bubble point density derivative
(f) <code>dDewDensity_dPressure</code>	Return dew point density derivative
(f) <code>dBubbleEnthalpy_dPressure</code>	Return bubble point specific enthalpy derivative
(f) <code>dDewEnthalpy_dPressure</code>	Return dew point specific enthalpy derivative
(f) <code>specificEnthalpy_pTX</code>	Return specific enthalpy from pressure, temperature and mass fraction
(f) <code>temperature_phX</code>	Return temperature from p, h, and X or Xi
(f) <code>density_phX</code>	Return density from p, h, and X or Xi
(f) <code>temperature_psX</code>	Return temperature from p, s, and X or Xi
(f) <code>density_psX</code>	Return density from p, s, and X or Xi
(f) <code>specificEnthalpy_psX</code>	Return specific enthalpy from p, s, and X or Xi
(f) <code>setState_pT</code>	Return thermodynamic state from p and T
(f) <code>setState_ph</code>	Return thermodynamic state from p and h
(f) <code>setState_ps</code>	Return thermodynamic state from p and s
(f) <code>setState_dT</code>	Return thermodynamic state from d and T
(f) <code>setState_px</code>	Return thermodynamic state from pressure and vapour

	quality
(f) <code>setState_Tx</code>	Return thermodynamic state from temperature and vapour quality
(f) <code>vapourQuality</code>	Return vapour quality
(f) <code>density_ph</code>	Return density from p and h
(f) <code>temperature_ph</code>	Return temperature from p and h
(f) <code>pressure_dT</code>	Return pressure from d and T
(f) <code>specificEnthalpy_dT</code>	Return specific enthalpy from d and T
(f) <code>specificEnthalpy_ps</code>	Return specific enthalpy from p and s
(f) <code>temperature_ps</code>	Return temperature from p and s
(f) <code>density_ps</code>	Return density from p and s
(f) <code>specificEnthalpy_pT</code>	Return specific enthalpy from p and T
(f) <code>density_pT</code>	Return density from p and T
<b>Inherited</b>	
<code>mediumName="unusablePartialMedium"</code>	Name of the medium
<code>substanceNames={mediumName}</code>	Names of the mixture substances. Set <code>substanceNames={mediumName}</code> if only one substance.
<code>extraPropertiesNames=fill("", 0)</code>	Names of the additional (extra) transported properties. Set <code>extraPropertiesNames=fill("",0)</code> if unused
<code>singleState</code>	= true, if u and d are not a function of pressure
<code>reducedX=true</code>	= true if medium contains the equation $\sum(X) = 1.0$ ; set <code>reducedX=true</code> if only one substance (see docu for details)
<code>fixedX=false</code>	= true if medium contains the equation $X = \text{reference\_}X$
<code>reference_p=101325</code>	Reference pressure of Medium: default 1 atmosphere
<code>reference_T=298.15</code>	Reference temperature of Medium: default 25 deg Celsius
<code>reference_X=if nX == 0 then fill(0, nX) else fill(1/nX, nX)</code>	Default mass fractions of medium
<code>p_default=101325</code>	Default value for pressure of medium (for initialization)
<code>T_default=Modelica.SIunits.Conversions.from_degC(20)</code>	Default value for temperature of medium (for initialization)
<code>h_default=specificEnthalpy_pTX(p_default, T_default, X_default)</code>	Default value for specific enthalpy of medium (for initialization)
<code>X_default=reference_X</code>	Default value for mass fractions of medium (for initialization)
<code>nS=size(substanceNames, 1)</code>	Number of substances
<code>nX=if nS == 1 then 0 else nS</code>	Number of mass fractions (= 0, if only one substance)
<code>nXi=if fixedX then 0 else if reducedX then nS - 1 else nX</code>	Number of structurally independent mass fractions (see docu for details)
<code>nC=size(extraPropertiesNames, 1)</code>	Number of extra (outside of standard mass-balance) transported properties
 <code>BasePropertiesRecord</code>	Variables contained in every instance of BaseProperties
(f) <code>dynamicViscosity</code>	Return dynamic viscosity
(f) <code>thermalConductivity</code>	Return thermal conductivity
(f) <code>prandtlNumber</code>	Return the Prandtl number
(f) <code>pressure</code>	Return pressure

<code>(f) temperature</code>	Return temperature
<code>(f) density</code>	Return density
<code>(f) specificEnthalpy</code>	Return specific enthalpy
<code>(f) specificInternalEnergy</code>	Return specific internal energy
<code>(f) specificEntropy</code>	Return specific entropy
<code>(f) specificGibbsEnergy</code>	Return specific Gibbs energy
<code>(f) specificHelmholtzEnergy</code>	Return specific Helmholtz energy
<code>(f) specificHeatCapacityCp</code>	Return specific heat capacity at constant pressure
<code>(f) heatCapacity_cp</code>	alias for deprecated name
<code>(f) specificHeatCapacityCv</code>	Return specific heat capacity at constant volume
<code>(f) heatCapacity_cv</code>	alias for deprecated name
<code>(f) isentropicExponent</code>	Return isentropic exponent
<code>(f) isentropicEnthalpy</code>	Return isentropic enthalpy
<code>(f) velocityOfSound</code>	Return velocity of sound
<code>(f) isobaricExpansionCoefficient</code>	Return overall the isobaric expansion coefficient beta
<code>(f) beta</code>	alias for isobaricExpansionCoefficient for user convenience
<code>(f) isothermalCompressibility</code>	Return overall the isothermal compressibility factor
<code>(f) kappa</code>	alias of isothermalCompressibility for user convenience
<code>(f) density_derP_h</code>	Return density derivative wrt pressure at const specific enthalpy
<code>(f) density_derH_p</code>	Return density derivative wrt specific enthalpy at constant pressure
<code>(f) density_derP_T</code>	Return density derivative wrt pressure at const temperature
<code>(f) density_derT_p</code>	Return density derivative wrt temperature at constant pressure
<code>(f) density_derX</code>	Return density derivative wrt mass fraction
<code>(f) density_pTX</code>	Return density from p, T, and X or Xi
<code>AbsolutePressure</code>	Type for absolute pressure with medium specific attributes
<code>Density</code>	Type for density with medium specific attributes
<code>DynamicViscosity</code>	Type for dynamic viscosity with medium specific attributes
<code>EnthalpyFlowRate</code>	Type for enthalpy flow rate with medium specific attributes
<code>MassFlowRate</code>	Type for mass flow rate with medium specific attributes
<code>MassFraction</code>	Type for mass fraction with medium specific attributes
<code>MoleFraction</code>	Type for mole fraction with medium specific attributes
<code>MolarMass</code>	Type for molar mass with medium specific attributes
<code>MolarVolume</code>	Type for molar volume with medium specific attributes
<code>IsentropicExponent</code>	Type for isentropic exponent with medium specific attributes
<code>SpecificEnergy</code>	Type for specific energy with medium specific attributes
<code>SpecificInternalEnergy</code>	Type for specific internal energy with medium specific attributes
<code>SpecificEnthalpy</code>	Type for specific enthalpy with medium specific attributes

SpecificEntropy	Type for specific entropy with medium specific attributes
SpecificHeatCapacity	Type for specific heat capacity with medium specific attributes
SurfaceTension	Type for surface tension with medium specific attributes
Temperature	Type for temperature with medium specific attributes
ThermalConductivity	Type for thermal conductivity with medium specific attributes
PrandtlNumber	Type for Prandtl number with medium specific attributes
VelocityOfSound	Type for velocity of sound with medium specific attributes
ExtraProperty	Type for unspecified, mass-specific property transported by flow
CumulativeExtraProperty	Type for conserved integral of unspecified, mass specific property
ExtraPropertyFlowRate	Type for flow rate of unspecified, mass-specific property
IsobaricExpansionCoefficient	Type for isobaric expansion coefficient with medium specific attributes
DipoleMoment	Type for dipole moment with medium specific attributes
DerDensityByPressure	Type for partial derivative of density with respect to pressure with medium specific attributes
DerDensityByEnthalpy	Type for partial derivative of density with respect to enthalpy with medium specific attributes
DerEnthalpyByPressure	Type for partial derivative of enthalpy with respect to pressure with medium specific attributes
DerDensityByTemperature	Type for partial derivative of density with respect to temperature with medium specific attributes
Choices	Types, constants to define menu choices

## Types and constants

```

constant Boolean smoothModel
"true if the (derived) model should not generate state events";

constant Boolean onePhase
"true if the (derived) model should never be called with two-phase inputs";

constant FluidConstants[nS] fluidConstants "constant data for the fluid";

type FixedPhase = Integer(min=0,max=2)
"phase of the fluid: 1 for 1-phase, 2 for two-phase, 0 for not known, e.g.
interactive use";

```

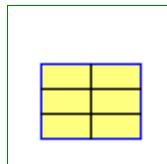
---

## Modelica.Media.Interfaces.PartialTwoPhaseMedium.FluidLimits

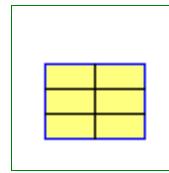
### validity limits for fluid model

### Information

The minimum pressure mostly applies to the liquid state only. The minimum density is also arbitrary, but is reasonable for technical applications to limit iterations in non-linear systems. The limits in enthalpy and entropy are used as safeguards in inverse iterations.



---

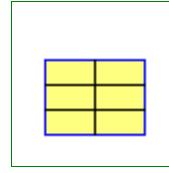
**Modelica.Media.Interfaces.PartialTwoPhaseMedium.FluidConstants****extended fluid constants****Modelica definition**

```

redeclare replaceable record extends FluidConstants
  "extended fluid constants"
  Temperature criticalTemperature "critical temperature";
  AbsolutePressure criticalPressure "critical pressure";
  MolarVolume criticalMolarVolume "critical molar Volume";
  Real acentricFactor "Pitzer acentric factor";
  Temperature triplePointTemperature "triple point temperature";
  AbsolutePressure triplePointPressure "triple point pressure";
  Temperature meltingPoint "melting point at 101325 Pa";
  Temperature normalBoilingPoint "normal boiling point (at 101325 Pa)";
  DipoleMoment dipoleMoment
    "dipole moment of molecule in Debye (1 debye = 3.33564e10-30 C.m)";
  Boolean hasIdealGasHeatCapacity=false
    "true if ideal gas heat capacity is available";
  Boolean hasCriticalData=false "true if critical data are known";
  Boolean hasDipoleMoment=false "true if a dipole moment known";
  Boolean hasFundamentalEquation=false "true if a fundamental equation";
  Boolean hasLiquidHeatCapacity=false
    "true if liquid heat capacity is available";
  Boolean hasSolidHeatCapacity=false "true if solid heat capacity is available";
  Boolean hasAccurateViscosityData=false
    "true if accurate data for a viscosity function is available";
  Boolean hasAccurateConductivityData=false
    "true if accurate data for thermal conductivity is available";
  Boolean hasVapourPressureCurve=false
    "true if vapour pressure data, e.g. Antoine coefficents are known";
  Boolean hasAcentricFactor=false "true if Pitzer acentric factor is known";
  SpecificEnthalpy HCRIT0=0.0
    "Critical specific enthalpy of the fundamental equation";
  SpecificEntropy SCRIT0=0.0
    "Critical specific entropy of the fundamental equation";
  SpecificEnthalpy deltah=0.0
    "Difference between specific enthalpy model (h_m) and f.eq. (h_f) (h_m - h_f)";
  SpecificEntropy deltas=0.0
    "Difference between specific enthalpy model (s_m) and f.eq. (s_f) (s_m - s_f)";
end FluidConstants;

```

---

**Modelica.Media.Interfaces.PartialTwoPhaseMedium.ThermodynamicState****Thermodynamic state of two phase medium****Modelica definition**

```

redeclare replaceable record extends ThermodynamicState
  "Thermodynamic state of two phase medium"

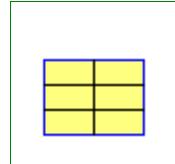
```

```
FixedPhase phase(min=0, max=2)
  "phase of the fluid: 1 for 1-phase, 2 for two-phase, 0 for not known, e.g.
interactive use";
end ThermodynamicState;
```

---

## Modelica.Media.Interfaces.PartialTwoPhaseMedium.SaturationProperties

Saturation properties of two phase medium



### Modelica definition

```
replaceable record SaturationProperties
  "Saturation properties of two phase medium"
  extends Modelica.Icons.Record;
  AbsolutePressure psat "saturation pressure";
  Temperature Tsat "saturation temperature";
end SaturationProperties;
```

---

## Modelica.Media.Interfaces.PartialTwoPhaseMedium.FixedPhase

phase of the fluid: 1 for 1-phase, 2 for two-phase, 0 for not known, e.g. interactive use

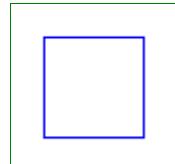
### Parameters

Name	Default	Description
min	0	
max	2	

---

## Modelica.Media.Interfaces.PartialTwoPhaseMedium.BaseProperties

Base properties (p, d, T, h, u, R, MM, sat) of two phase medium



### Parameters

Name	Default	Description
standardOrderComponents	true	if true, last element in components is computed from 1-sum(Xi)
<b>Advanced</b>		
preferredMediumStates	false	= true if StateSelect.prefer shall be used for the independent property variables of the medium

---

## Modelica.Media.Interfaces.PartialTwoPhaseMedium.setDewState

Return the thermodynamic state on the dew line



**Inputs**

Name	Default	Description
sat		saturation point
phase	1	phase: default is one phase

**Outputs**

Name	Description
state	complete thermodynamic state info

**Modelica.Media.Interfaces.PartialTwoPhaseMedium.setBubbleState**

Return the thermodynamic state on the bubble line

**Inputs**

Name	Default	Description
sat		saturation point
phase	1	phase: default is one phase

**Outputs**

Name	Description
state	complete thermodynamic state info

**Modelica.Media.Interfaces.PartialTwoPhaseMedium.setState\_dTX**

Return thermodynamic state as function of d, T and composition X or Xi

**Inputs**

Name	Default	Description
phase	0	2 for two-phase, 1 for one-phase, 0 if not known
d		density [kg/m3]
T		Temperature [K]
X[:]	reference_X	Mass fractions [kg/kg]

**Outputs**

Name	Description
state	

**Modelica.Media.Interfaces.PartialTwoPhaseMedium.setState\_phX**

Return thermodynamic state as function of p, h and composition X or Xi

**Inputs**

Name	Default	Description

---

## 744 Modelica.Media.Interfaces.PartialTwoPhaseMedium.setState\_phX

---

phase	0	2 for two-phase, 1 for one-phase, 0 if not known
p		Pressure [Pa]
h		Specific enthalpy [J/kg]
X[:]	reference_X	Mass fractions [kg/kg]

### Outputs

Name	Description
state	

---

## Modelica.Media.Interfaces.PartialTwoPhaseMedium.setState\_psX

Return thermodynamic state as function of p, s and composition X or Xi



### Inputs

Name	Default	Description
phase	0	2 for two-phase, 1 for one-phase, 0 if not known
p		Pressure [Pa]
s		Specific entropy [J/(kg.K)]
X[:]	reference_X	Mass fractions [kg/kg]

### Outputs

Name	Description
state	

---

## Modelica.Media.Interfaces.PartialTwoPhaseMedium.setState\_pTX

Return thermodynamic state as function of p, T and composition X or Xi



### Inputs

Name	Default	Description
phase	0	2 for two-phase, 1 for one-phase, 0 if not known
p		Pressure [Pa]
T		Temperature [K]
X[:]	reference_X	Mass fractions [kg/kg]

### Outputs

Name	Description
state	

---

## Modelica.Media.Interfaces.PartialTwoPhaseMedium.setSat\_T

Return saturation property record from temperature



**Inputs**

Name	Default	Description
T		temperature [K]

**Outputs**

Name	Description
sat	saturation property record

**Modelica.Media.Interfaces.PartialTwoPhaseMedium.setSat\_p**

Return saturation property record from pressure

**Inputs**

Name	Default	Description
p		pressure [Pa]

**Outputs**

Name	Description
sat	saturation property record

**Modelica.Media.Interfaces.PartialTwoPhaseMedium.bubbleEnthalpy**

Return bubble point specific enthalpy

**Inputs**

Name	Default	Description
sat		saturation property record

**Outputs**

Name	Description
hl	boiling curve specific enthalpy [J/kg]

**Modelica.Media.Interfaces.PartialTwoPhaseMedium.dewEnthalpy**

Return dew point specific enthalpy

**Inputs**

Name	Default	Description
sat		saturation property record

**Outputs**

Name	Description
hv	dew curve specific enthalpy [J/kg]

**Modelica.Media.Interfaces.PartialTwoPhaseMedium.bubbleEntropy**

Return bubble point specific entropy

**Inputs**

Name	Default	Description
sat		saturation property record

**Outputs**

Name	Description
sl	boiling curve specific entropy [J/(kg.K)]

---

**Modelica.Media.Interfaces.PartialTwoPhaseMedium.dewEntropy**

Return dew point specific entropy

**Inputs**

Name	Default	Description
sat		saturation property record

**Outputs**

Name	Description
sv	dew curve specific entropy [J/(kg.K)]

---

**Modelica.Media.Interfaces.PartialTwoPhaseMedium.bubbleDensity**

Return bubble point density

**Inputs**

Name	Default	Description
sat		saturation property record

**Outputs**

Name	Description
dl	boiling curve density [kg/m3]

---

**Modelica.Media.Interfaces.PartialTwoPhaseMedium.dewDensity**

Return dew point density

**Inputs**

Name	Default	Description
sat		saturation property record

**Outputs**

Name	Description
dv	dew curve density [kg/m3]

**Modelica.Media.Interfaces.PartialTwoPhaseMedium.saturationPressure**

Return saturation pressure

**Inputs**

Name	Default	Description
T		temperature [K]

**Outputs**

Name	Description
p	saturation pressure [Pa]

**Modelica.Media.Interfaces.PartialTwoPhaseMedium.saturationTemperature**

Return saturation temperature

**Inputs**

Name	Default	Description
p		pressure [Pa]

**Outputs**

Name	Description
T	saturation temperature [K]

**Modelica.Media.Interfaces.PartialTwoPhaseMedium.saturationPressure\_sat**

Return saturation temperature

**Inputs**

Name	Default	Description
sat		saturation property record

**Outputs**

Name	Description
p	saturation pressure [Pa]

**Modelica.Media.Interfaces.PartialTwoPhaseMedium.saturationTemperature\_sat**

Return saturation temperature



**Inputs**

Name	Default	Description
sat		saturation property record

**Outputs**

Name	Description
T	saturation temperature [K]

**Modelica.Media.Interfaces.PartialTwoPhaseMedium.saturationTemperature\_derp**

Return derivative of saturation temperature w.r.t. pressure

**Inputs**

Name	Default	Description
p		pressure [Pa]

**Outputs**

Name	Description
dTp	derivative of saturation temperature w.r.t. pressure

**Modelica.Media.Interfaces.PartialTwoPhaseMedium.saturationTemperature\_derp\_sat**

Return derivative of saturation temperature w.r.t. pressure

**Inputs**

Name	Default	Description
sat		saturation property record

**Outputs**

Name	Description
dTp	derivative of saturation temperature w.r.t. pressure

**Modelica.Media.Interfaces.PartialTwoPhaseMedium.surfaceTension**

Return surface tension sigma in the two phase region

**Inputs**

Name	Default	Description
sat		saturation property record

**Outputs**

Name	Description

---

sigma	Surface tension sigma in the two phase region [N/m]
-------	---

**Modelica.Media.Interfaces.PartialTwoPhaseMedium.molarMass**

Return the molar mass of the medium

**Inputs**

Name	Default	Description
state		

**Outputs**

Name	Description
MM	Mixture molar mass [kg/mol]

**Modelica.Media.Interfaces.PartialTwoPhaseMedium.dBubbleDensity\_dPressure**

Return bubble point density derivative

**Inputs**

Name	Default	Description
sat		saturation property record

**Outputs**

Name	Description
ddldp	boiling curve density derivative [s <sup>2</sup> /m <sup>2</sup> ]

**Modelica.Media.Interfaces.PartialTwoPhaseMedium.dDewDensity\_dPressure**

Return dew point density derivative

**Inputs**

Name	Default	Description
sat		saturation property record

**Outputs**

Name	Description
ddvdp	saturated steam density derivative [s <sup>2</sup> /m <sup>2</sup> ]

**Modelica.Media.Interfaces.PartialTwoPhaseMedium.dBubbleEnthalpy\_dPressure**

Return bubble point specific enthalpy derivative



---

## 750 Modelica.Media.Interfaces.PartialTwoPhaseMedium.dBubbleEnthalpy\_dPressure

---

### Inputs

Name	Default	Description
sat		saturation property record

### Outputs

Name	Description
dhldp	boiling curve specific enthalpy derivative [J.m.s2/kg2]



## Modelica.Media.Interfaces.PartialTwoPhaseMedium.dDewEnthalpy\_dPressure

Return dew point specific enthalpy derivative

### Inputs

Name	Default	Description
sat		saturation property record

### Outputs

Name	Description
dhvdp	saturated steam specific enthalpy derivative [J.m.s2/kg2]



## Modelica.Media.Interfaces.PartialTwoPhaseMedium.specificEnthalpy\_pTX

Return specific enthalpy from pressure, temperature and mass fraction

### Inputs

Name	Default	Description
p		Pressure [Pa]
T		Temperature [K]
X[nX]		Mass fractions [kg/kg]
phase	0	2 for two-phase, 1 for one-phase, 0 if not known

### Outputs

Name	Description
h	Specific enthalpy at p, T, X [J/kg]



## Modelica.Media.Interfaces.PartialTwoPhaseMedium.temperature\_phX

Return temperature from p, h, and X or Xi

### Inputs

Name	Default	Description
p		Pressure [Pa]
h		Specific enthalpy [J/kg]

X[nX]		Mass fractions [kg/kg]
phase	0	2 for two-phase, 1 for one-phase, 0 if not known

## Outputs

Name	Description
T	Temperature [K]

## Modelica.Media.Interfaces.PartialTwoPhaseMedium.density\_phX

Return density from p, h, and X or Xi



## Inputs

Name	Default	Description
p		Pressure [Pa]
h		Specific enthalpy [J/kg]
X[nX]		Mass fractions [kg/kg]
phase	0	2 for two-phase, 1 for one-phase, 0 if not known

## Outputs

Name	Description
d	density [kg/m3]

## Modelica.Media.Interfaces.PartialTwoPhaseMedium.temperature\_psX

Return temperature from p, s, and X or Xi



## Inputs

Name	Default	Description
p		Pressure [Pa]
s		Specific entropy [J/(kg.K)]
X[nX]		Mass fractions [kg/kg]
phase	0	2 for two-phase, 1 for one-phase, 0 if not known

## Outputs

Name	Description
T	Temperature [K]

## Modelica.Media.Interfaces.PartialTwoPhaseMedium.density\_psX

Return density from p, s, and X or Xi



## Inputs

Name	Default	Description
p		Pressure [Pa]

---

## 752 Modelica.Media.Interfaces.PartialTwoPhaseMedium.density\_psX

---

s	Specific entropy [J/(kg.K)]
X[nX]	Mass fractions [kg/kg]
phase	0 for two-phase, 1 for one-phase, 0 if not known

### Outputs

Name	Description
d	Density [kg/m3]

---

## Modelica.Media.Interfaces.PartialTwoPhaseMedium.specificEnthalpy\_psX



Return specific enthalpy from p, s, and X or Xi

### Inputs

Name	Default	Description
p		Pressure [Pa]
s		Specific entropy [J/(kg.K)]
X[nX]		Mass fractions [kg/kg]
phase	0	2 for two-phase, 1 for one-phase, 0 if not known

### Outputs

Name	Description
h	specific enthalpy [J/kg]

---

## Modelica.Media.Interfaces.PartialTwoPhaseMedium.setState\_pT



Return thermodynamic state from p and T

### Inputs

Name	Default	Description
p		Pressure [Pa]
T		Temperature [K]
phase	0	2 for two-phase, 1 for one-phase, 0 if not known

### Outputs

Name	Description
state	

---

## Modelica.Media.Interfaces.PartialTwoPhaseMedium.setState\_ph



Return thermodynamic state from p and h

### Inputs

Name	Default	Description
p		Pressure [Pa]

h	Specific enthalpy [J/kg]
phase	0 2 for two-phase, 1 for one-phase, 0 if not known

## Outputs

Name	Description
state	

### Modelica.Media.Interfaces.PartialTwoPhaseMedium.setState\_ps

Return thermodynamic state from p and s



## Inputs

Name	Default	Description
p		Pressure [Pa]
s		Specific entropy [J/(kg.K)]
phase	0	2 for two-phase, 1 for one-phase, 0 if not known

## Outputs

Name	Description
state	

### Modelica.Media.Interfaces.PartialTwoPhaseMedium.setState\_dT

Return thermodynamic state from d and T



## Inputs

Name	Default	Description
d		density [kg/m3]
T		Temperature [K]
phase	0	2 for two-phase, 1 for one-phase, 0 if not known

## Outputs

Name	Description
state	

### Modelica.Media.Interfaces.PartialTwoPhaseMedium.setState\_px

Return thermodynamic state from pressure and vapour quality

## Inputs

Name	Default	Description
p		Pressure [Pa]
x		Vapour quality [kg/kg]

---

## 754 Modelica.Media.Interfaces.PartialTwoPhaseMedium.setState\_px

---

### Outputs

Name	Description
state	Thermodynamic state record

---

## Modelica.Media.Interfaces.PartialTwoPhaseMedium.setState\_Tx

Return thermodynamic state from temperature and vapour quality

### Inputs

Name	Default	Description
T		Temperature [K]
x		Vapour quality [kg/kg]

### Outputs

Name	Description
state	

---

## Modelica.Media.Interfaces.PartialTwoPhaseMedium.vapourQuality

Return vapour quality

### Inputs

Name	Default	Description
state		Thermodynamic state record

### Outputs

Name	Description
x	Vapour quality [kg/kg]

---

## Modelica.Media.Interfaces.PartialTwoPhaseMedium.density\_ph

Return density from p and h



### Inputs

Name	Default	Description
p		Pressure [Pa]
h		Specific enthalpy [J/kg]
phase	0	2 for two-phase, 1 for one-phase, 0 if not known

### Outputs

Name	Description
d	Density [kg/m3]

---

**Modelica.Media.Interfaces.PartialTwoPhaseMedium.temperature\_ph**

Return temperature from p and h

**Inputs**

Name	Default	Description
p		Pressure [Pa]
h		Specific enthalpy [J/kg]
phase	0	2 for two-phase, 1 for one-phase, 0 if not known

**Outputs**

Name	Description
T	Temperature [K]

**Modelica.Media.Interfaces.PartialTwoPhaseMedium.pressure\_dT**

Return pressure from d and T

**Inputs**

Name	Default	Description
d		Density [kg/m3]
T		Temperature [K]
phase	0	2 for two-phase, 1 for one-phase, 0 if not known

**Outputs**

Name	Description
p	Pressure [Pa]

**Modelica.Media.Interfaces.PartialTwoPhaseMedium.specificEnthalpy\_dT**

Return specific enthalpy from d and T

**Inputs**

Name	Default	Description
d		Density [kg/m3]
T		Temperature [K]
phase	0	2 for two-phase, 1 for one-phase, 0 if not known

**Outputs**

Name	Description
h	specific enthalpy [J/kg]

**Modelica.Media.Interfaces.PartialTwoPhaseMedium.specificEnthalpy\_ps**

Return specific enthalpy from p and s

---

## 756 Modelica.Media.Interfaces.PartialTwoPhaseMedium.specificEnthalpy\_ps

---

### Inputs

Name	Default	Description
p		Pressure [Pa]
s		Specific entropy [J/(kg.K)]
phase	0	2 for two-phase, 1 for one-phase, 0 if not known

### Outputs

Name	Description
h	specific enthalpy [J/kg]

---

## Modelica.Media.Interfaces.PartialTwoPhaseMedium.temperature\_ps

Return temperature from p and s



### Inputs

Name	Default	Description
p		Pressure [Pa]
s		Specific entropy [J/(kg.K)]
phase	0	2 for two-phase, 1 for one-phase, 0 if not known

### Outputs

Name	Description
T	Temperature [K]

---

## Modelica.Media.Interfaces.PartialTwoPhaseMedium.density\_ps

Return density from p and s



### Inputs

Name	Default	Description
p		Pressure [Pa]
s		Specific entropy [J/(kg.K)]
phase	0	2 for two-phase, 1 for one-phase, 0 if not known

### Outputs

Name	Description
d	Density [kg/m3]

---

## Modelica.Media.Interfaces.PartialTwoPhaseMedium.specificEnthalpy\_pT

Return specific enthalpy from p and T



## Inputs

Name	Default	Description
p		Pressure [Pa]
T		Temperature [K]
phase	0	2 for two-phase, 1 for one-phase, 0 if not known

## Outputs

Name	Description
h	specific enthalpy [J/kg]

---

## Modelica.Media.Interfaces.PartialTwoPhaseMedium.density\_pT

Return density from p and T



## Inputs

Name	Default	Description
p		Pressure [Pa]
T		Temperature [K]
phase	0	2 for two-phase, 1 for one-phase, 0 if not known

## Outputs

Name	Description
d	Density [kg/m3]

---

## Modelica.Media.Interfaces.PartialSimpleMedium

Medium model with linear dependency of u, h from temperature. All other quantities, especially density, are constant.

## Package Content

Name	Description
cp_const	Constant specific heat capacity at constant pressure
cv_const	Constant specific heat capacity at constant volume
d_const	Constant density
eta_const	Constant dynamic viscosity
lambda_const	Constant thermal conductivity
a_const	Constant velocity of sound
T_min	Minimum temperature valid for medium model
T_max	Maximum temperature valid for medium model
T0=reference_T	Zero enthalpy temperature
MM_const	Molar mass
fluidConstants	fluid constants
ThermodynamicState	Thermodynamic state
BaseProperties	Base properties

---

**758 Modelica.Media.Interfaces.PartialSimpleMedium**


---

(f) <code>setState_pTX</code>	Return thermodynamic state from p, T, and X or Xi
(f) <code>setState_phX</code>	Return thermodynamic state from p, h, and X or Xi
(f) <code>setState_psX</code>	Return thermodynamic state from p, s, and X or Xi
(f) <code>setState_dTX</code>	Return thermodynamic state from d, T, and X or Xi
(f) <code>dynamicViscosity</code>	Return dynamic viscosity
(f) <code>thermalConductivity</code>	Return thermal conductivity
(f) <code>specificHeatCapacityCp</code>	Return specific heat capacity at constant pressure
(f) <code>specificHeatCapacityCv</code>	Return specific heat capacity at constant volume
(f) <code>isentropicExponent</code>	Return isentropic exponent
(f) <code>velocityOfSound</code>	Return velocity of sound
(f) <code>specificEnthalpy_pTX</code>	Return specific enthalpy from p, T, and X or Xi
(f) <code>temperature_phX</code>	Return temperature from p, h, and X or Xi
(f) <code>density_phX</code>	Return density from p, h, and X or Xi
<b>Inherited</b>	
(f) <code>setState_pT</code>	Return thermodynamic state from p and T
(f) <code>setState_ph</code>	Return thermodynamic state from p and h
(f) <code>setState_ps</code>	Return thermodynamic state from p and s
(f) <code>setState_dT</code>	Return thermodynamic state from d and T
(f) <code>density_ph</code>	Return density from p and h
(f) <code>temperature_ph</code>	Return temperature from p and h
(f) <code>pressure_dT</code>	Return pressure from d and T
(f) <code>specificEnthalpy_dT</code>	Return specific enthalpy from d and T
(f) <code>specificEnthalpy_ps</code>	Return specific enthalpy from p and s
(f) <code>temperature_ps</code>	Return temperature from p and s
(f) <code>density_ps</code>	Return density from p and s
(f) <code>specificEnthalpy_pT</code>	Return specific enthalpy from p and T
(f) <code>density_pT</code>	Return density from p and T
<code>mediumName="unusablePartialMedium"</code>	Name of the medium
<code>substanceNames={mediumName}</code>	Names of the mixture substances. Set <code>substanceNames={mediumName}</code> if only one substance.
<code>extraPropertiesNames=fill("", 0)</code>	Names of the additional (extra) transported properties. Set <code>extraPropertiesNames=fill("",0)</code> if unused
<code>singleState</code>	= true, if u and d are not a function of pressure
<code>reducedX=true</code>	= true if medium contains the equation $\sum(X) = 1.0$ ; set <code>reducedX=true</code> if only one substance (see docu for details)
<code>fixedX=false</code>	= true if medium contains the equation $X = \text{reference\_}X$
<code>reference_p=101325</code>	Reference pressure of Medium: default 1 atmosphere
<code>reference_T=298.15</code>	Reference temperature of Medium: default 25 deg Celsius
<code>reference_X=if nX == 0 then fill(0, nX) else fill(1/nX, nX)</code>	Default mass fractions of medium

p_default=101325	Default value for pressure of medium (for initialization)
T_default=Modelica.Slunits.Conversions.from_degC(20)	Default value for temperature of medium (for initialization)
h_default=specificEnthalpy_pTX(p_default, T_default, X_default)	Default value for specific enthalpy of medium (for initialization)
X_default=reference_X	Default value for mass fractions of medium (for initialization)
nS=size(substanceNames, 1)	Number of substances
nX;if nS == 1 then 0 else nS	Number of mass fractions (= 0, if only one substance)
nXi;if fixedX then 0 else if reducedX then nS - 1 else nX	Number of structurally independent mass fractions (see docu for details)
nC=size(extraPropertiesNames, 1)	Number of extra (outside of standard mass-balance) transported properties
 FluidConstants	critical, triple, molecular and other standard data of fluid
 BasePropertiesRecord	Variables contained in every instance of BaseProperties
(f) prandtlNumber	Return the Prandtl number
(f) pressure	Return pressure
(f) temperature	Return temperature
(f) density	Return density
(f) specificEnthalpy	Return specific enthalpy
(f) specificInternalEnergy	Return specific internal energy
(f) specificEntropy	Return specific entropy
(f) specificGibbsEnergy	Return specific Gibbs energy
(f) specificHelmholtzEnergy	Return specific Helmholtz energy
(f) heatCapacity_cp	alias for deprecated name
(f) heatCapacity_cv	alias for deprecated name
(f) isentropicEnthalpy	Return isentropic enthalpy
(f) isobaricExpansionCoefficient	Return overall the isobaric expansion coefficient beta
(f) beta	alias for isobaricExpansionCoefficient for user convenience
(f) isothermalCompressibility	Return overall the isothermal compressibility factor
(f) kappa	alias of isothermalCompressibility for user convenience
(f) density_derh_h	Return density derivative wrt pressure at const specific enthalpy
(f) density_derh_p	Return density derivative wrt specific enthalpy at constant pressure
(f) density_derh_T	Return density derivative wrt pressure at const temperature
(f) density_derT_p	Return density derivative wrt temperature at constant pressure
(f) density_derX	Return density derivative wrt mass fraction
(f) molarMass	Return the molar mass of the medium
(f) density_pTX	Return density from p, T, and X or Xi
(f) temperature_psX	Return temperature from p,s, and X or Xi

 <a href="#">density_psX</a>	Return density from p, s, and X or Xi
 <a href="#">specificEnthalpy_psX</a>	Return specific enthalpy from p, s, and X or Xi
<a href="#">AbsolutePressure</a>	Type for absolute pressure with medium specific attributes
<a href="#">Density</a>	Type for density with medium specific attributes
<a href="#">DynamicViscosity</a>	Type for dynamic viscosity with medium specific attributes
<a href="#">EnthalpyFlowRate</a>	Type for enthalpy flow rate with medium specific attributes
<a href="#">MassFlowRate</a>	Type for mass flow rate with medium specific attributes
<a href="#">MassFraction</a>	Type for mass fraction with medium specific attributes
<a href="#">MoleFraction</a>	Type for mole fraction with medium specific attributes
<a href="#">MolarMass</a>	Type for molar mass with medium specific attributes
<a href="#">MolarVolume</a>	Type for molar volume with medium specific attributes
<a href="#">IsentropicExponent</a>	Type for isentropic exponent with medium specific attributes
<a href="#">SpecificEnergy</a>	Type for specific energy with medium specific attributes
<a href="#">SpecificInternalEnergy</a>	Type for specific internal energy with medium specific attributes
<a href="#">SpecificEnthalpy</a>	Type for specific enthalpy with medium specific attributes
<a href="#">SpecificEntropy</a>	Type for specific entropy with medium specific attributes
<a href="#">SpecificHeatCapacity</a>	Type for specific heat capacity with medium specific attributes
<a href="#">SurfaceTension</a>	Type for surface tension with medium specific attributes
<a href="#">Temperature</a>	Type for temperature with medium specific attributes
<a href="#">ThermalConductivity</a>	Type for thermal conductivity with medium specific attributes
<a href="#">PrandtlNumber</a>	Type for Prandtl number with medium specific attributes
<a href="#">VelocityOfSound</a>	Type for velocity of sound with medium specific attributes
<a href="#">ExtraProperty</a>	Type for unspecified, mass-specific property transported by flow
<a href="#">CumulativeExtraProperty</a>	Type for conserved integral of unspecified, mass specific property
<a href="#">ExtraPropertyFlowRate</a>	Type for flow rate of unspecified, mass-specific property
<a href="#">IsobaricExpansionCoefficient</a>	Type for isobaric expansion coefficient with medium specific attributes
<a href="#">DipoleMoment</a>	Type for dipole moment with medium specific attributes
<a href="#">DerDensityByPressure</a>	Type for partial derivative of density with respect to pressure with medium specific attributes
<a href="#">DerDensityByEnthalpy</a>	Type for partial derivative of density with respect to enthalpy with medium specific attributes
<a href="#">DerEnthalpyByPressure</a>	Type for partial derivative of enthalpy with respect to pressure with medium specific attributes
<a href="#">DerDensityByTemperature</a>	Type for partial derivative of density with respect to temperature with medium specific attributes
 <a href="#">Choices</a>	Types, constants to define menu choices

## Types and constants

```
constant SpecificHeatCapacity cp_const
"Constant specific heat capacity at constant pressure";
```

```
constant SpecificHeatCapacity cv_const
```

```

"Constant specific heat capacity at constant volume";

constant Density d_const "Constant density";

constant DynamicViscosity eta_const "Constant dynamic viscosity";

constant ThermalConductivity lambda_const "Constant thermal conductivity";

constant VelocityOfSound a_const "Constant velocity of sound";

constant Temperature T_min "Minimum temperature valid for medium model";

constant Temperature T_max "Maximum temperature valid for medium model";

constant Temperature T0=reference_T "Zero enthalpy temperature";

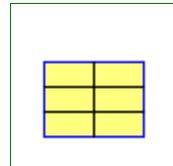
constant MolarMass MM_const "Molar mass";

constant FluidConstants[nS] fluidConstants "fluid constants";

```

## Modelica.Media.Interfaces.PartialSimpleMedium.ThermodynamicState

### Thermodynamic state



### Modelica definition

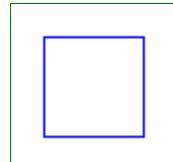
```

redeclare replaceable record extends ThermodynamicState
  "Thermodynamic state"
  AbsolutePressure p "Absolute pressure of medium";
  Temperature T "Temperature of medium";
end ThermodynamicState;

```

## Modelica.Media.Interfaces.PartialSimpleMedium.BaseProperties

### Base properties



### Information

This is the most simple incompressible medium model, where specific enthalpy  $h$  and specific internal energy  $u$  are only a function of temperature  $T$  and all other provided medium quantities are assumed to be constant.

### Parameters

Name	Default	Description
standardOrderComponents	true	if true, last element in components is computed from $1 - \sum(X_i)$
<b>Advanced</b>		
preferredMediumStates	false	= true if StateSelect.prefer shall be used for the independent property variables of the medium

**Modelica.Media.Interfaces.PartialSimpleMedium.setState\_pTX**

Return thermodynamic state from p, T, and X or Xi

**Inputs**

Name	Default	Description
p		Pressure [Pa]
T		Temperature [K]
X[:]	reference_X	Mass fractions [kg/kg]

**Outputs**

Name	Description
state	

**Modelica.Media.Interfaces.PartialSimpleMedium.setState\_phX**

Return thermodynamic state from p, h, and X or Xi

**Inputs**

Name	Default	Description
p		Pressure [Pa]
h		Specific enthalpy [J/kg]
X[:]	reference_X	Mass fractions [kg/kg]

**Outputs**

Name	Description
state	

**Modelica.Media.Interfaces.PartialSimpleMedium.setState\_psX**

Return thermodynamic state from p, s, and X or Xi

**Inputs**

Name	Default	Description
p		Pressure [Pa]
s		Specific entropy [J/(kg.K)]
X[:]	reference_X	Mass fractions [kg/kg]

**Outputs**

Name	Description
state	

**Modelica.Media.Interfaces.PartialSimpleMedium.setState\_dTX**

Return thermodynamic state from d, T, and X or Xi

**Inputs**

Name	Default	Description
d		density [kg/m3]
T		Temperature [K]
X[:]	reference_X	Mass fractions [kg/kg]

**Outputs**

Name	Description
state	

**Modelica.Media.Interfaces.PartialSimpleMedium.dynamicViscosity**

Return dynamic viscosity

**Inputs**

Name	Default	Description
state		

**Outputs**

Name	Description
eta	Dynamic viscosity [Pa.s]

**Modelica.Media.Interfaces.PartialSimpleMedium.thermalConductivity**

Return thermal conductivity

**Inputs**

Name	Default	Description
state		

**Outputs**

Name	Description
lambda	Thermal conductivity [W/(m.K)]

**Modelica.Media.Interfaces.PartialSimpleMedium.specificHeatCapacityCp**

Return specific heat capacity at constant pressure

**Inputs**

Name	Default	Description

---

**764 Modelica.Media.Interfaces.PartialSimpleMedium.specificHeatCapacityCp**

---

state		
-------	--	--

**Outputs**

Name	Description
cp	Specific heat capacity at constant pressure [J/(kg.K)]

---

**Modelica.Media.Interfaces.PartialSimpleMedium.specificHeatCapacityCv**

Return specific heat capacity at constant volume

**Inputs**

Name	Default	Description
state		

**Outputs**

Name	Description
cv	Specific heat capacity at constant volume [J/(kg.K)]

---

**Modelica.Media.Interfaces.PartialSimpleMedium.isentropicExponent**

Return isentropic exponent

**Inputs**

Name	Default	Description
state		

**Outputs**

Name	Description
gamma	Isentropic exponent [1]

---

**Modelica.Media.Interfaces.PartialSimpleMedium.velocityOfSound**

Return velocity of sound

**Inputs**

Name	Default	Description
state		

**Outputs**

Name	Description
a	Velocity of sound [m/s]

**Modelica.Media.Interfaces.PartialSimpleMedium.specificEnthalpy\_pTX**

Return specific enthalpy from p, T, and X or Xi

**Inputs**

Name	Default	Description
p		Pressure [Pa]
T		Temperature [K]
X[nX]		Mass fractions [kg/kg]

**Outputs**

Name	Description
h	Specific enthalpy [J/kg]

**Modelica.Media.Interfaces.PartialSimpleMedium.temperature\_phX**

Return temperature from p, h, and X or Xi

**Inputs**

Name	Default	Description
p		Pressure [Pa]
h		Specific enthalpy [J/kg]
X[nX]		Mass fractions [kg/kg]

**Outputs**

Name	Description
T	Temperature [K]

**Modelica.Media.Interfaces.PartialSimpleMedium.density\_phX**

Return density from p, h, and X or Xi

**Inputs**

Name	Default	Description
p		Pressure [Pa]
h		Specific enthalpy [J/kg]
X[nX]		Mass fractions [kg/kg]

**Outputs**

Name	Description
d	density [kg/m3]

**Modelica.Media.Interfaces.PartialSimpleIdealGasMedium**

Medium model of Ideal gas with constant cp and cv. All other quantities, e.g. transport properties, are

constant.

## Package Content

Name	Description
cp_const	Constant specific heat capacity at constant pressure
cv_const=cp_const - R_gas	Constant specific heat capacity at constant volume
R_gas	medium specific gas constant
MM_const	Molar mass
eta_const	Constant dynamic viscosity
lambda_const	Constant thermal conductivity
T_min	Minimum temperature valid for medium model
T_max	Maximum temperature valid for medium model
T0=reference_T	Zero enthalpy temperature
 ThermodynamicState	Thermodynamic state of ideal gas
 BaseProperties	Base properties of ideal gas
(f) setState_pTX	Return thermodynamic state from p, T, and X or Xi
(f) setState_phX	Return thermodynamic state from p, h, and X or Xi
(f) setState_psX	Return thermodynamic state from p, s, and X or Xi
(f) setState_dTX	Return thermodynamic state from d, T, and X or Xi
(f) pressure	Return pressure of ideal gas
(f) temperature	Return temperature of ideal gas
(f) density	Return density of ideal gas
(f) specificEnthalpy	Return specific enthalpy
(f) specificInternalEnergy	Return specific internal energy
(f) specificEntropy	Return specific entropy
(f) specificGibbsEnergy	Return specific Gibbs energy
(f) specificHelmholtzEnergy	Return specific Helmholtz energy
(f) dynamicViscosity	Return dynamic viscosity
(f) thermalConductivity	Return thermal conductivity
(f) specificHeatCapacityCp	Return specific heat capacity at constant pressure
(f) specificHeatCapacityCv	Return specific heat capacity at constant volume
(f) isentropicExponent	Return isentropic exponent
(f) velocityOfSound	Return velocity of sound
(f) specificEnthalpy_pTX	Return specific enthalpy from p, T, and X or Xi
(f) temperature_phX	Return temperature from p, h, and X or Xi
(f) density_phX	Return density from p, h, and X or Xi
Inherited	
(f) setState_pT	Return thermodynamic state from p and T
(f) setState_ph	Return thermodynamic state from p and h

<code>setState_ps</code>	Return thermodynamic state from p and s
<code>setState_dT</code>	Return thermodynamic state from d and T
<code>density_ph</code>	Return density from p and h
<code>temperature_ph</code>	Return temperature from p and h
<code>pressure_dT</code>	Return pressure from d and T
<code>specificEnthalpy_dT</code>	Return specific enthalpy from d and T
<code>specificEnthalpy_ps</code>	Return specific enthalpy from p and s
<code>temperature_ps</code>	Return temperature from p and s
<code>density_ps</code>	Return density from p and s
<code>specificEnthalpy_pT</code>	Return specific enthalpy from p and T
<code>density_pT</code>	Return density from p and T
<code>mediumName="unusablePartialMedium"</code>	Name of the medium
<code>substanceNames={mediumName}</code>	Names of the mixture substances. Set substanceNames={mediumName} if only one substance.
<code>extraPropertiesNames=fill("", 0)</code>	Names of the additional (extra) transported properties. Set extraPropertiesNames=fill("",0) if unused
<code>singleState</code>	= true, if u and d are not a function of pressure
<code>reducedX=true</code>	= true if medium contains the equation sum(X) = 1.0; set reducedX=true if only one substance (see docu for details)
<code>fixedX=false</code>	= true if medium contains the equation X = reference_X
<code>reference_p=101325</code>	Reference pressure of Medium: default 1 atmosphere
<code>reference_T=298.15</code>	Reference temperature of Medium: default 25 deg Celsius
<code>reference_X=if nX == 0 then fill(0, nX) else fill(1/nX, nX)</code>	Default mass fractions of medium
<code>p_default=101325</code>	Default value for pressure of medium (for initialization)
<code>T_default=Modelica.Slunits.Conversions.from_degC(20)</code>	Default value for temperature of medium (for initialization)
<code>h_default=specificEnthalpy_pTX(p_default, T_default, X_default)</code>	Default value for specific enthalpy of medium (for initialization)
<code>X_default=reference_X</code>	Default value for mass fractions of medium (for initialization)
<code>nS=size(substanceNames, 1)</code>	Number of substances
<code>nX=if nS == 1 then 0 else nS</code>	Number of mass fractions (= 0, if only one substance)
<code>nXi=if fixedX then 0 else if reducedX then nS - 1 else nX</code>	Number of structurally independent mass fractions (see docu for details)
<code>nC=size(extraPropertiesNames, 1)</code>	Number of extra (outside of standard mass-balance) transported properties
	critical, triple, molecular and other standard data of fluid
	Variables contained in every instance of BaseProperties
<code>prandtlNumber</code>	Return the Prandtl number
<code>heatCapacity_cp</code>	alias for deprecated name
<code>heatCapacity_cv</code>	alias for deprecated name
<code>isentropicEnthalpy</code>	Return isentropic enthalpy
<code>isobaricExpansionCoefficient</code>	Return overall the isobaric expansion coefficient beta

(f) <a href="#">beta</a>	alias for isobaricExpansionCoefficient for user convenience
(f) <a href="#">isothermalCompressibility</a>	Return overall the isothermal compressibility factor
(f) <a href="#">kappa</a>	alias of isothermalCompressibility for user convenience
(f) <a href="#">density_derP_h</a>	Return density derivative wrt pressure at const specific enthalpy
(f) <a href="#">density_derH_p</a>	Return density derivative wrt specific enthalpy at constant pressure
(f) <a href="#">density_derP_T</a>	Return density derivative wrt pressure at const temperature
(f) <a href="#">density_derT_p</a>	Return density derivative wrt temperature at constant pressure
(f) <a href="#">density_derX</a>	Return density derivative wrt mass fraction
(f) <a href="#">molarMass</a>	Return the molar mass of the medium
(f) <a href="#">density_pTX</a>	Return density from p, T, and X or Xi
(f) <a href="#">temperature_psX</a>	Return temperature from p,s, and X or Xi
(f) <a href="#">density_psX</a>	Return density from p, s, and X or Xi
(f) <a href="#">specificEnthalpy_psX</a>	Return specific enthalpy from p, s, and X or Xi
AbsolutePressure	Type for absolute pressure with medium specific attributes
Density	Type for density with medium specific attributes
DynamicViscosity	Type for dynamic viscosity with medium specific attributes
EnthalpyFlowRate	Type for enthalpy flow rate with medium specific attributes
MassFlowRate	Type for mass flow rate with medium specific attributes
MassFraction	Type for mass fraction with medium specific attributes
MoleFraction	Type for mole fraction with medium specific attributes
MolarMass	Type for molar mass with medium specific attributes
MolarVolume	Type for molar volume with medium specific attributes
IsentropicExponent	Type for isentropic exponent with medium specific attributes
SpecificEnergy	Type for specific energy with medium specific attributes
SpecificInternalEnergy	Type for specific internal energy with medium specific attributes
SpecificEnthalpy	Type for specific enthalpy with medium specific attributes
SpecificEntropy	Type for specific entropy with medium specific attributes
SpecificHeatCapacity	Type for specific heat capacity with medium specific attributes
SurfaceTension	Type for surface tension with medium specific attributes
Temperature	Type for temperature with medium specific attributes
ThermalConductivity	Type for thermal conductivity with medium specific attributes
PrandtlNumber	Type for Prandtl number with medium specific attributes
VelocityOfSound	Type for velocity of sound with medium specific attributes
ExtraProperty	Type for unspecified, mass-specific property transported by flow
CumulativeExtraProperty	Type for conserved integral of unspecified, mass specific property
ExtraPropertyFlowRate	Type for flow rate of unspecified, mass-specific property
IsobaricExpansionCoefficient	Type for isobaric expansion coefficient with medium specific attributes

DipoleMoment	Type for dipole moment with medium specific attributes
DerDensityByPressure	Type for partial derivative of density with respect to pressure with medium specific attributes
DerDensityByEnthalpy	Type for partial derivative of density with respect to enthalpy with medium specific attributes
DerEnthalpyByPressure	Type for partial derivative of enthalpy with respect to pressure with medium specific attributes
DerDensityByTemperature	Type for partial derivative of density with respect to temperature with medium specific attributes
 Choices	Types, constants to define menu choices

## Types and constants

```

constant SpecificHeatCapacity cp_const
"Constant specific heat capacity at constant pressure";

constant SpecificHeatCapacity cv_const= cp_const - R_gas
"Constant specific heat capacity at constant volume";

constant SpecificHeatCapacity R_gas "medium specific gas constant";

constant MolarMass MM_const "Molar mass";

constant DynamicViscosity eta_const "Constant dynamic viscosity";

constant ThermalConductivity lambda_const "Constant thermal conductivity";

constant Temperature T_min "Minimum temperature valid for medium model";

constant Temperature T_max "Maximum temperature valid for medium model";

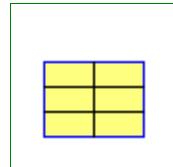
constant Temperature T0= reference_T "Zero enthalpy temperature";

```

---

## Modelica.Media.Interfaces.PartialSimpleIdealGasMedium.ThermodynamicState

Thermodynamic state of ideal gas



### Modelica definition

```

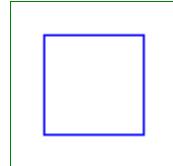
redeclare replaceable record extends ThermodynamicState
  "Thermodynamic state of ideal gas"
  AbsolutePressure p "Absolute pressure of medium";
  Temperature T "Temperature of medium";
end ThermodynamicState;

```

---

## Modelica.Media.Interfaces.PartialSimpleIdealGasMedium.BaseProperties

Base properties of ideal gas



## Information

This is the most simple incompressible medium model, where specific enthalpy  $h$  and specific internal energy  $u$  are only a function of temperature  $T$  and all other provided medium quantities are assumed to be constant.

## Parameters

Name	Default	Description
standardOrderComponents	true	if true, last element in components is computed from 1-sum( $\chi_i$ )
<b>Advanced</b>		
preferredMediumStates	false	= true if StateSelect.prefer shall be used for the independent property variables of the medium

---

## Modelica.Media.Interfaces.PartialSimpleIdealGasMedium.setState\_pTX

Return thermodynamic state from p, T, and X or  $\chi$



### Inputs

Name	Default	Description
p		Pressure [Pa]
T		Temperature [K]
X[:]	reference_X	Mass fractions [kg/kg]

### Outputs

Name	Description
state	

---

## Modelica.Media.Interfaces.PartialSimpleIdealGasMedium.setState\_phX

Return thermodynamic state from p, h, and X or  $\chi$



### Inputs

Name	Default	Description
p		Pressure [Pa]
h		Specific enthalpy [J/kg]
X[:]	reference_X	Mass fractions [kg/kg]

### Outputs

Name	Description
state	

---

## Modelica.Media.Interfaces.PartialSimpleIdealGasMedium.setState\_psX

Return thermodynamic state from p, s, and X or  $\chi$



## Inputs

Name	Default	Description
p		Pressure [Pa]
s		Specific entropy [J/(kg.K)]
X[:]	reference_X	Mass fractions [kg/kg]

## Outputs

Name	Description
state	

## Modelica.Media.Interfaces.PartialSimpleIdealGasMedium.setState\_dTX

Return thermodynamic state from d, T, and X or Xi



## Inputs

Name	Default	Description
d		density [kg/m <sup>3</sup> ]
T		Temperature [K]
X[:]	reference_X	Mass fractions [kg/kg]

## Outputs

Name	Description
state	

## Modelica.Media.Interfaces.PartialSimpleIdealGasMedium.pressure

Return pressure of ideal gas



## Inputs

Name	Default	Description
state		

## Outputs

Name	Description
p	Pressure [Pa]

## Modelica.Media.Interfaces.PartialSimpleIdealGasMedium.temperature

Return temperature of ideal gas



## Inputs

Name	Default	Description
state		

---

## 772 Modelica.Media.Interfaces.PartialSimpleIdealGasMedium.temperature

---

### Outputs

Name	Description
T	Temperature [K]

---



### Modelica.Media.Interfaces.PartialSimpleIdealGasMedium.density

Return density of ideal gas

### Inputs

Name	Default	Description
state		

### Outputs

Name	Description
d	Density [kg/m3]

---



### Modelica.Media.Interfaces.PartialSimpleIdealGasMedium.specificEnthalpy

Return specific enthalpy

### Inputs

Name	Default	Description
state		

### Outputs

Name	Description
h	Specific enthalpy [J/kg]

---



### Modelica.Media.Interfaces.PartialSimpleIdealGasMedium.specificInternalEnergy

Return specific internal energy

### Inputs

Name	Default	Description
state		

### Outputs

Name	Description
u	Specific internal energy [J/kg]

---



### Modelica.Media.Interfaces.PartialSimpleIdealGasMedium.specificEntropy

Return specific entropy

**Inputs**

Name	Default	Description
state		

**Outputs**

Name	Description
s	Specific entropy [J/(kg.K)]

**Modelica.Media.Interfaces.PartialSimpleIdealGasMedium.specificGibbsEnergy**

Return specific Gibbs energy

**Inputs**

Name	Default	Description
state		

**Outputs**

Name	Description
g	Specific Gibbs energy [J/kg]

**Modelica.Media.Interfaces.PartialSimpleIdealGasMedium.specificHelmholtzEnergy**

Return specific Helmholtz energy

**Inputs**

Name	Default	Description
state		

**Outputs**

Name	Description
f	Specific Helmholtz energy [J/kg]

**Modelica.Media.Interfaces.PartialSimpleIdealGasMedium.dynamicViscosity**

Return dynamic viscosity

**Inputs**

Name	Default	Description
state		

**Outputs**

Name	Description
eta	Dynamic viscosity [Pa.s]

**Modelica.Media.Interfaces.PartialSimpleIdealGasMedium.thermalConductivity**

Return thermal conductivity

**Inputs**

Name	Default	Description
state		

**Outputs**

Name	Description
lambda	Thermal conductivity [W/(m.K)]

**Modelica.Media.Interfaces.PartialSimpleIdealGasMedium.specificHeatCapacityCp**

Return specific heat capacity at constant pressure

**Inputs**

Name	Default	Description
state		

**Outputs**

Name	Description
cp	Specific heat capacity at constant pressure [J/(kg.K)]

**Modelica.Media.Interfaces.PartialSimpleIdealGasMedium.specificHeatCapacityCv**

Return specific heat capacity at constant volume

**Inputs**

Name	Default	Description
state		

**Outputs**

Name	Description
cv	Specific heat capacity at constant volume [J/(kg.K)]

**Modelica.Media.Interfaces.PartialSimpleIdealGasMedium.isentropicExponent**

Return isentropic exponent

**Inputs**

Name	Default	Description
state		

## Outputs

Name	Description
gamma	Isentropic exponent [1]

---

## Modelica.Media.Interfaces.PartialSimpleIdealGasMedium.velocityOfSound

Return velocity of sound



## Inputs

Name	Default	Description
state		

## Outputs

Name	Description
a	Velocity of sound [m/s]

---

## Modelica.Media.Interfaces.PartialSimpleIdealGasMedium.specificEnthalpy\_pTX

Return specific enthalpy from p, T, and X or Xi



## Inputs

Name	Default	Description
p		Pressure [Pa]
T		Temperature [K]
X[nX]		Mass fractions [kg/kg]

## Outputs

Name	Description
h	Specific enthalpy at p, T, X [J/kg]

---

## Modelica.Media.Interfaces.PartialSimpleIdealGasMedium.temperature\_phX

Return temperature from p, h, and X or Xi



## Inputs

Name	Default	Description
p		Pressure [Pa]
h		Specific enthalpy [J/kg]
X[nX]		Mass fractions [kg/kg]

## Outputs

Name	Description
T	Temperature [K]

**Modelica.Media.Interfaces.PartialSimpleIdealGasMedium.density\_phX**

Return density from p, h, and X or Xi

**Inputs**

Name	Default	Description
p		Pressure [Pa]
h		Specific enthalpy [J/kg]
X[nX]		Mass fractions [kg/kg]

**Outputs**

Name	Description
d	density [kg/m <sup>3</sup> ]

**Modelica.Media.Common**

data structures and fundamental functions for fluid properties

**Information****Package description**

Package Modelica.Media.Common provides records and functions shared by many of the property sub-packages. High accuracy fluid property models share a lot of common structure, even if the actual models are different. Common data structures and computations shared by these property models are collected in this library.

**Package Content**

Name	Description
SaturationProperties	properties in the two phase region
SaturationBoundaryProperties	properties on both phase boundaries, including some derivatives
IF97BaseTwoPhase	Intermediate property data record for IF 97
IF97PhaseBoundaryProperties	thermodynamic base properties on the phase boundary for IF97 steam tables
GibbsDerivs	derivatives of dimensionless Gibbs-function w.r.t dimensionless pressure and temperature
HelmholtzDerivs	derivatives of dimensionless Helmholtz-function w.r.t dimensionless pressure density and temperature
TwoPhaseTransportProps	defines properties on both phase boundaries, needed in the two phase region
PhaseBoundaryProperties	thermodynamic base properties on the phase boundary
NewtonDerivatives_ph	derivatives for fast inverse calculations of Helmholtz functions: p & h
NewtonDerivatives_ps	derivatives for fast inverse calculation of Helmholtz functions: p & s
NewtonDerivatives_pT	derivatives for fast inverse calculations of Helmholtz functions:p & T

 ExtraDerivatives	additional thermodynamic derivatives
 BridgmansTables	Calculates all entries in Bridgmans tables if first seven variables given
 gibbsToBridgmansTables	calculates base coefficients for bridgemans tables from gibbs enthalpy
 helmholtzToBridgmansTables	calculates base coefficients for Bridgmans tables from helmholtz energy
 gibbsToBoundaryProps	calulate phase boundary property record from dimensionless Gibbs function
 helmholtzToBoundaryProps	calulate phase boundary property record from dimensionless Helmholtz function
 cv2Phase	compute isochoric specific heat capacity inside the two-phase region
 cvdpT2Phase	compute isochoric specific heat capacity inside the two-phase region and derivative of pressure w.r.t. temperature
 gibbsToExtraDerivs	compute additional thermodynamic derivatives from dimensionless Gibbs function
 helmholtzToExtraDerivs	compute additional thermodynamic derivatives from dimensionless Helmholtz function
 Helmholtz_ph	function to calculate analytic derivatives for computing d and t given p and h
 Helmholtz_pT	function to calculate analytic derivatives for computing d and t given p and t
 Helmholtz_ps	function to calculate analytic derivatives for computing d and t given p and s
 OneNonLinearEquation	Determine solution of a non-linear algebraic equation in one unknown without derivatives in a reliable and efficient way

## Types and constants

```

type Rate = Real (final quantity="Rate", final unit="s-1");

type MolarFlowRate = Real (final quantity="MolarFlowRate", final
unit="mol/s");

type MolarReactionRate = Real (final quantity="MolarReactionRate", final unit
= "mol/(m3.s)");

type MolarEnthalpy = Real (final quantity="MolarEnthalpy", final
unit="J/mol");

type DerDensityByEntropy = Real (final quantity="DerDensityByEntropy", final
unit
= "kg2.K/(m3.J)");

type DerEnergyByPressure = Real (final quantity="DerEnergyByPressure", final
unit
= "J/Pa");

type DerEnergyByMoles = Real (final quantity="DerEnergyByMoles", final unit=
"J/mol");

type DerEntropyByTemperature = Real (final quantity="DerEntropyByTemperature",
final unit="J/K2");

```

```
type DerEntropyByPressure = Real (final quantity="DerEntropyByPressure",
                                    final unit="J/(K.Pa)");
type DerEntropyByMoles = Real (final quantity="DerEntropyByMoles", final unit
= "J/(mol.K)");
type DerPressureByDensity = Real (final quantity="DerPressureByDensity",
                                    final unit="Pa.m3/kg");
type DerPressureBySpecificVolume = Real (final quantity=
    "DerPressureBySpecificVolume", final unit="Pa.kg/m3");
type DerPressureByTemperature = Real (final quantity=
    "DerPressureByTemperature", final unit="Pa/K");
type DerVolumeByTemperature = Real (final quantity="DerVolumeByTemperature",
                                    final unit="m3/K");
type DerVolumeByPressure = Real (final quantity="DerVolumeByPressure", final
unit
= "m3/Pa");
type DerVolumeByMoles = Real (final quantity="DerVolumeByMoles", final unit=
    "m3/mol");
type IsenthalpicExponent = Real (final quantity="IsenthalpicExponent", unit=
    "1");
type IsentropicExponent = Real (final quantity="IsentropicExponent", unit="1");
type IsobaricVolumeExpansionCoefficient = Real (final quantity=
    "IsobaricVolumeExpansionCoefficient", unit="1/K");
type IsochoricPressureCoefficient = Real (final quantity=
    "IsochoricPressureCoefficient", unit="1/K");
type IsothermalCompressibility = Real (final quantity=
    "IsothermalCompressibility", unit="1/Pa");
type JouleThomsonCoefficient = Real (final quantity="JouleThomsonCoefficient",
                                    unit="K/Pa");
constant Real MINPOS=1.0e-9
"minimal value for physical variables which are always > 0.0";
constant SI.Area AMIN=MINPOS "minimal init area";
constant SI.Area AMAX=1.0e5 "maximal init area";
constant SI.Area ANOM=1.0 "nominal init area";
constant SI.AmountOfSubstance MOLMIN=-1.0*MINPOS "minimal Mole Number";
```

```
constant SI.AmountOfSubstance MOLMAX=1.0e8 "maximal Mole Number";  
  
constant SI.AmountOfSubstance MOLNOM=1.0 "nominal Mole Number";  
  
constant SI.Density DMIN=MINPOS "minimal init density";  
  
constant SI.Density DMAX=1.0e5 "maximal init density";  
  
constant SI.Density DNOM=1.0 "nominal init density";  
  
constant SI.ThermalConductivity LAMMIN=MINPOS "minimal thermal conductivity";  
  
constant SI.ThermalConductivity LAMNOM=1.0 "nominal thermal conductivity";  
  
constant SI.ThermalConductivity LAMMAX=1000.0 "maximal thermal conductivity";  
  
constant SI.DynamicViscosity ETAMIN=MINPOS "minimal init dynamic viscosity";  
  
constant SI.DynamicViscosity ETAMAX=1.0e8 "maximal init dynamic viscosity";  
  
constant SI.DynamicViscosity ETANOM=100.0 "nominal init dynamic viscosity";  
  
constant SI.Energy EMIN=-1.0e10 "minimal init energy";  
  
constant SI.Energy EMAX=1.0e10 "maximal init energy";  
  
constant SI.Energy ENOM=1.0e3 "nominal init energy";  
  
constant SI.Entropy SMIN=-1.0e6 "minimal init entropy";  
  
constant SI.Entropy SMAX=1.0e6 "maximal init entropy";  
  
constant SI.Entropy SNOM=1.0e3 "nominal init entropy";  
  
constant SI.MassFlowRate MDOTMIN=-1.0e5 "minimal init mass flow rate";  
  
constant SI.MassFlowRate MDOTMAX=1.0e5 "maximal init mass flow rate";  
  
constant SI.MassFlowRate MDOTNOM=1.0 "nominal init mass flow rate";  
  
constant SI.MassFraction MASSXMIN=-1.0*MINPOS "minimal init mass fraction";  
  
constant SI.MassFraction MASSXMAX=1.0 "maximal init mass fraction";  
  
constant SI.MassFraction MASSXNOM=0.1 "nominal init mass fraction";  
  
constant SI.Mass MMIN=-1.0*MINPOS "minimal init mass";  
  
constant SI.Mass MMAX=1.0e8 "maximal init mass";
```

```
constant SI.Mass MNOM=1.0 "nominal init mass";

constant SI.MolarMass MMMIN=0.001 "minimal initial molar mass";

constant SI.MolarMass MMMAX=250.0 "maximal initial molar mass";

constant SI.MolarMass MNMOM=0.2 "nominal initial molar mass";

constant SI.MoleFraction MOLEYMIN=-1.0*MINPOS "minimal init mole fraction";

constant SI.MoleFraction MOLEYMAX=1.0 "maximal init mole fraction";

constant SI.MoleFraction MOLEYNOM=0.1 "nominal init mole fraction";

constant SI.MomentumFlux GMIN=-1.0e8 "minimal init momentum flux";

constant SI.MomentumFlux GMAX=1.0e8 "maximal init momentum flux";

constant SI.MomentumFlux GNOM=1.0 "nominal init momentum flux";

constant SI.Power POWMIN=-1.0e8 "minimal init power or heat";

constant SI.Power POWMAX=1.0e8 "maximal init power or heat";

constant SI.Power POWNOM=1.0e3 "nominal init power or heat";

constant SI.Pressure PMIN=1.0e4 "minimal init pressure";

constant SI.Pressure PMAX=1.0e8 "maximal init pressure";

constant SI.Pressure PNOM=1.0e5 "nominal init pressure";

constant SI.Pressure COMPPMIN=-1.0*MINPOS "minimal init pressure";

constant SI.Pressure COMPPMAX=1.0e8 "maximal init pressure";

constant SI.Pressure COMPPNOM=1.0e5 "nominal init pressure";

constant SI.RatioOfSpecificHeatCapacities KAPPAMIN=1.0
"minimal init isentropic exponent";

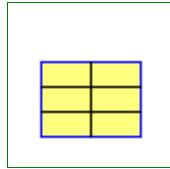
constant SI.RatioOfSpecificHeatCapacities KAPPAMAX=1.7
"maximal init isentropic exponent";

constant SI.RatioOfSpecificHeatCapacities KAPPANOM=1.2
"nominal init isentropic exponent";

constant SI.SpecificEnergy SEMIN=-1.0e8 "minimal init specific energy";

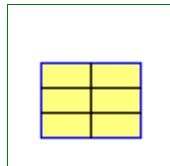
constant SI.SpecificEnergy SEMAX=1.0e8 "maximal init specific energy";
```

```
constant SI.SpecificEnergy SENOM=1.0e6 "nominal init specific energy";  
  
constant SI.SpecificEnthalpy SHMIN=-1.0e8 "minimal init specific enthalpy";  
  
constant SI.SpecificEnthalpy SHMAX=1.0e8 "maximal init specific enthalpy";  
  
constant SI.SpecificEnthalpy SHNOM=1.0e6 "nominal init specific enthalpy";  
  
constant SI.SpecificEntropy SSMIN=-1.0e6 "minimal init specific entropy";  
  
constant SI.SpecificEntropy SSMAX=1.0e6 "maximal init specific entropy";  
  
constant SI.SpecificEntropy SSNOM=1.0e3 "nominal init specific entropy";  
  
constant SI.SpecificHeatCapacity CPMIN=MINPOS  
"minimal init specific heat capacity";  
  
constant SI.SpecificHeatCapacity CPMAX=1.0e6  
"maximal init specific heat capacity";  
  
constant SI.SpecificHeatCapacity CPNOM=1.0e3  
"nominal init specific heat capacity";  
  
constant SI.Temperature TMIN=MINPOS "minimal init temperature";  
  
constant SI.Temperature TMAX=1.0e5 "maximal init temperature";  
  
constant SI.Temperature TNOM=320.0 "nominal init temperature";  
  
constant SI.ThermalConductivity LMIN=MINPOS  
"minimal init thermal conductivity";  
  
constant SI.ThermalConductivity LMAX=500.0  
"maximal init thermal conductivity";  
  
constant SI.ThermalConductivity LNOM=1.0 "nominal init thermal conductivity";  
  
constant SI.Velocity VELMIN=-1.0e5 "minimal init speed";  
  
constant SI.Velocity VELMAX=1.0e5 "maximal init speed";  
  
constant SI.Velocity VELNOM=1.0 "nominal init speed";  
  
constant SI.Volume VMIN=0.0 "minimal init volume";  
  
constant SI.Volume VMAX=1.0e5 "maximal init volume";  
  
constant SI.Volume VNOM=1.0e-3 "nominal init volume";
```

**Modelica.Media.Common.SaturationProperties****properties in the two phase region****Modelica definition**

```
record SaturationProperties "properties in the two phase region"
  extends Modelica.Icons.Record;
  SI.Temp_K T "temperature";
  SI.Density d "density";
  SI.Pressure p "pressure";
  SI.SpecificEnergy u "specific inner energy";
  SI.SpecificEnthalpy h "specific enthalpy";
  SI.SpecificEntropy s "specific entropy";
  SI.SpecificHeatCapacity cp "heat capacity at constant pressure";
  SI.SpecificHeatCapacity cv "heat capacity at constant volume";
  SI.SpecificHeatCapacity R "gas constant";
  SI.RatioOfSpecificHeatCapacities kappa "isentropic expansion coefficient";
  PhaseBoundaryProperties liq
    "thermodynamic base properties on the boiling curve";
  PhaseBoundaryProperties vap "thermodynamic base properties on the dew curve";
  Real dpT(unit="Pa/K") "derivative of saturation pressure wrt temperature";
  SI.MassFraction x "vapour mass fraction";
end SaturationProperties;
```

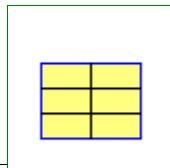
---

**Modelica.Media.Common.SaturationBoundaryProperties****properties on both phase boundaries, including some derivatives****Modelica definition**

```
record SaturationBoundaryProperties
  "properties on both phase boundaries, including some derivatives"

  extends Modelica.Icons.Record;
  SI.Temp_K T "Saturation temperature";
  SI.Density dl "Liquid density";
  SI.Density dv "Vapour density";
  SI.SpecificEnthalpy hl "Liquid specific enthalpy";
  SI.SpecificEnthalpy hv "Vapour specific enthalpy";
  Real dTp "derivative of temperature wrt saturation pressure";
  Real ddldp "derivative of density along boiling curve";
  Real ddvdp "derivative of density along dew curve";
  Real dhldp "derivative of specific enthalpy along boiling curve";
  Real dhvdp "derivative of specific enthalpy along dew curve";
  SI.MassFraction x "vapour mass fraction";
end SaturationBoundaryProperties;
```

---

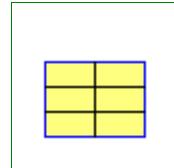
**Modelica.Media.Common.IF97BaseTwoPhase****Intermediate property data record for IF 97**

## Modelica definition

```
record IF97BaseTwoPhase "Intermediate property data record for IF 97"
  extends Modelica.Icons.Record;
  Integer phase= 0 "phase: 2 for two-phase, 1 for one phase, 0 if unknown";
  Integer region(min=1, max=5) "IF 97 region";
  SI.Pressure p "pressure";
  SI.Temperature T "temperature";
  SI.SpecificEnthalpy h "specific enthalpy";
  SI.SpecificHeatCapacity R "gas constant";
  SI.SpecificHeatCapacity cp "specific heat capacity";
  SI.SpecificHeatCapacity cv "specific heat capacity";
  SI.Density rho "density";
  SI.SpecificEntropy s "specific entropy";
  DerPressureByTemperature pt "derivative of pressure wrt temperature";
  DerPressureByDensity pd "derivative of pressure wrt density";
  Real vt "derivative of specific volume w.r.t. temperature";
  Real vp "derivative of specific volume w.r.t. pressure";
  Real x "dryness fraction";
  Real dpT "dp/dT derivative of saturation curve";
end IF97BaseTwoPhase;
```

## Modelica.Media.Common.IF97PhaseBoundaryProperties

thermodynamic base properties on the phase boundary for IF97 steam tables



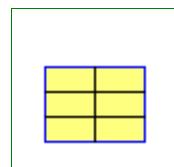
## Modelica definition

```
record IF97PhaseBoundaryProperties
  "thermodynamic base properties on the phase boundary for IF97 steam tables"

  extends Modelica.Icons.Record;
  Boolean region3boundary "true if boundary between 2-phase and region 3";
  SI.SpecificHeatCapacity R "specific heat capacity";
  SI.Temperature T "temperature";
  SI.Density d "density";
  SI.SpecificEnthalpy h "specific enthalpy";
  SI.SpecificEntropy s "specific entropy";
  SI.SpecificHeatCapacity cp "heat capacity at constant pressure";
  SI.SpecificHeatCapacity cv "heat capacity at constant volume";
  DerPressureByTemperature dpT "dp/dT derivative of saturation curve";
  DerPressureByTemperature pt "derivative of pressure wrt temperature";
  DerPressureByDensity pd "derivative of pressure wrt density";
  Real vt(unit="m3/(kg.K)") "derivative of specific volume w.r.t. temperature";
  Real vp(unit="m3/(kg.Pa)") "derivative of specific volume w.r.t. pressure";
end IF97PhaseBoundaryProperties;
```

## Modelica.Media.Common.GibbsDerivs

derivatives of dimensionless Gibbs-function w.r.t dimensionless pressure and temperature



**Modelica definition**

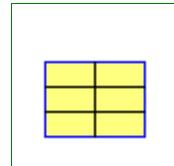
```
record GibbsDerivs
  "derivatives of dimensionless Gibbs-function w.r.t dimensionless pressure and
  temperature"

  extends Modelica.Icons.Record;
  SI.Pressure p "pressure";
  SI.Temperature T "temperature";
  SI.SpecificHeatCapacity R "specific heat capacity";
  Real pi(unit="1") "dimensionless pressure";
  Real tau(unit="1") "dimensionless temperature";
  Real g(unit="1") "dimensionless Gibbs-function";
  Real gpi(unit="1") "derivative of g w.r.t. pi";
  Real gpipi(unit="1") "2nd derivative of g w.r.t. pi";
  Real gtau(unit="1") "derivative of g w.r.t. tau";
  Real gtautau(unit="1") "2nd derivative of g w.r.t tau";
  Real gtaupi(unit="1") "mixed derivative of g w.r.t. pi and tau";
end GibbsDerivs;
```

---

**Modelica.Media.Common.HelmholtzDerivs**

derivatives of dimensionless Helmholtz-function w.r.t dimensionless pressuredensity and temperature

**Modelica definition**

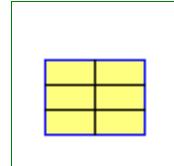
```
record HelmholtzDerivs
  "derivatives of dimensionless Helmholtz-function w.r.t dimensionless
  pressuredensity and temperature"

  extends Modelica.Icons.Record;
  SI.Density d "density";
  SI.Temperature T "temperature";
  SI.SpecificHeatCapacity R "specific heat capacity";
  Real delta(unit="1") "dimensionless density";
  Real tau(unit="1") "dimensionless temperature";
  Real f(unit="1") "dimensionless Helmholtz-function";
  Real fdelta(unit="1") "derivative of f w.r.t. delta";
  Real fdeltadelta(unit="1") "2nd derivative of f w.r.t. delta";
  Real ftau(unit="1") "derivative of f w.r.t. tau";
  Real ftautau(unit="1") "2nd derivative of f w.r.t. tau";
  Real fdeltatau(unit="1") "mixed derivative of f w.r.t. delta and tau";
end HelmholtzDerivs;
```

---

**Modelica.Media.Common.TwoPhaseTransportProps**

defines properties on both phase boundaries, needed in the two phase region

**Modelica definition**

```
record TwoPhaseTransportProps
  "defines properties on both phase boundaries, needed in the two phase region"
  extends Modelica.Icons.Record;
```

---

```

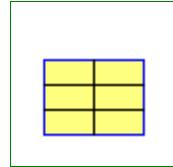
SI.Density d_vap "density on the dew line";
SI.Density d_liq "density on the bubble line";
SI.DynamicViscosity eta_vap "dynamic viscosity on the dew line";
SI.DynamicViscosity eta_liq "dynamic viscosity on the bubble line";
SI.ThermalConductivity lam_vap "thermal conductivity on the dew line";
SI.ThermalConductivity lam_liq "thermal conductivity on the bubble line";
SI.SpecificHeatCapacity cp_vap "cp on the dew line";
SI.SpecificHeatCapacity cp_liq "cp on the bubble line";
SI.MassFraction x "steam quality";
end TwoPhaseTransportProps;

```

---

## Modelica.Media.Common.PhaseBoundaryProperties

**thermodynamic base properties on the phase boundary**



### Modelica definition

```

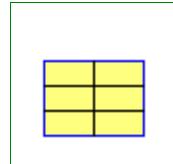
record PhaseBoundaryProperties
  "thermodynamic base properties on the phase boundary"
  extends Modelica.Icons.Record;
  SI.Density d "density";
  SI.SpecificEnthalpy h "specific enthalpy";
  SI.SpecificEnergy u "inner energy";
  SI.SpecificEntropy s "specific entropy";
  SI.SpecificHeatCapacity cp "heat capacity at constant pressure";
  SI.SpecificHeatCapacity cv "heat capacity at constant volume";
  DerPressureByTemperature pt "derivative of pressure wrt temperature";
  DerPressureByDensity pd "derivative of pressure wrt density";
end PhaseBoundaryProperties;

```

---

## Modelica.Media.Common.NewtonDerivatives\_ph

**derivatives for fast inverse calculations of Helmholtz functions: p & h**



### Modelica definition

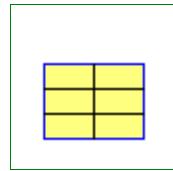
```

record NewtonDerivatives_ph
  "derivatives for fast inverse calculations of Helmholtz functions: p & h"

  extends Modelica.Icons.Record;
  SI.Pressure p "pressure";
  SI.SpecificEnthalpy h "specific enthalpy";
  DerPressureByDensity pd "derivative of pressure w.r.t. density";
  DerPressureByTemperature pt "derivative of pressure w.r.t. temperature";
  Real hd "derivative of specific enthalpy w.r.t. density";
  Real ht "derivative of specific enthalpy w.r.t. temperature";
end NewtonDerivatives_ph;

```

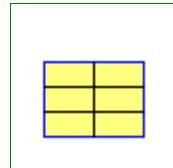
---

**Modelica.Media.Common.NewtonDerivatives\_ps****derivatives for fast inverse calculation of Helmholtz functions: p & s****Modelica definition**

```
record NewtonDerivatives_ps
  "derivatives for fast inverse calculation of Helmholtz functions: p & s"

  extends Modelica.Icons.Record;
  SI.Pressure p "pressure";
  SI.SpecificEntropy s "specific entropy";
  DerPressureByDensity pd "derivative of pressure w.r.t. density";
  DerPressureByTemperature pt "derivative of pressure w.r.t. temperature";
  Real sd "derivative of specific entropy w.r.t. density";
  Real st "derivative of specific entropy w.r.t. temperature";
end NewtonDerivatives_ps;
```

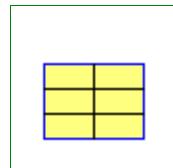
---

**Modelica.Media.Common.NewtonDerivatives\_pT****derivatives for fast inverse calculations of Helmholtz functions:p & T****Modelica definition**

```
record NewtonDerivatives_pT
  "derivatives for fast inverse calculations of Helmholtz functions:p & T"

  extends Modelica.Icons.Record;
  SI.Pressure p "pressure";
  DerPressureByDensity pd "derivative of pressure w.r.t. density";
end NewtonDerivatives_pT;
```

---

**Modelica.Media.Common.ExtraDerivatives****additional thermodynamic derivatives****Modelica definition**

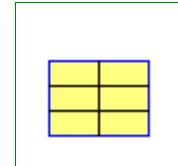
```
record ExtraDerivatives "additional thermodynamic derivatives"
  extends Modelica.Icons.Record;
  IsentropicExponent kappa "isentropic expansion coefficient";
  // k in Bejan
  IsenthalpicExponent theta "isenthalpic exponent";
  // same as kappa, except derivative at const h
  IsobaricVolumeExpansionCoefficient alpha
    "isobaric volume expansion coefficient";
  // beta in Bejan
  IsochoricPressureCoefficient beta "isochoric pressure coefficient";
  // kT in Bejan
  IsothermalCompressibility gamma "isothermal compressibility";
  // kappa in Bejan
  JouleThomsonCoefficient mu "Joule-Thomson coefficient";
```

---

```
// mu_J in Bejan
end ExtraDerivatives;
```

## Modelica.Media.Common.BridgmansTables

**Calculates all entries in Bridgmans tables if first seven variables given**



### Information

Important: the phase equilibrium conditions are not yet considered. this means that bridgmans tables do not yet work in the two phase region. Some derivatives are 0 or infinity anyways. Idea: don't use the values in Bridgmans table directly, all derivatives are calculated as the quotient of two entries in the table. The last letter indicates which variable is held constant in taking the derivative. The second letters are the two variables involved in the derivative and the first letter is always a d to remind of differentiation.

Example 1: Get the derivative of specific entropy s wrt Temperature at constant specific volume (btw identical to constant density)

constant volume --> last letter v  
 Temperature --> second letter T  
 Specific entropy --> second letter s  
 --> the needed value is dsv/dTv

Known variables:

Temperature T  
 pressure p  
 specific volume v  
 specific inner energy u  
 specific enthalpy h  
 specific entropy s  
 specific helmholtz energy f  
 specific gibbs enthalpy g

Not included but useful:

density d

In order to convert derivatives involving density use the following rules:

at constant density == at constant specific volume  
 $ddx/dyx = -d^2 * dvx/dyx$  with y,x any of T,p,u,h,s,f,g  
 $dyx/ddx = -1/(d^2 * dx) dyx/dvx$  with y,x any of T,p,u,h,s,f,g

Usage example assuming water as the medium:

```
model BridgmansTablesForWater
  extends ThermoFluid.BaseClasses.MediumModels.Water.WaterSteamMedium_ph;
  Real derOfsByTAtConstantv "derivative of sp. entropy by temperature at constant
  sp. volume"
  ThermoFluid.BaseClasses.MediumModels.Common.ExtraDerivatives dpro;
  ThermoFluid.BaseClasses.MediumModels.Common.BridgmansTables bt;
  equation
    dpro = ThermoFluid.BaseClasses.MediumModels.SteamIF97.extraDerivs_pT(p[1],T[1]);
    bt.p = p[1];
    bt.T = T[1];
    bt.v = 1/pro[1].d;
    bt.s = pro[1].s;
    bt.cp = pro[1].cp;
    bt.alpha = dpro.alpha;
    bt.gamma = dpro.gamma;
    derOfsByTAtConstantv = bt.dsv/bt.dTv;
    ...
  end BridgmansTablesForWater;
```

**Modelica.Media.Common.gibbsToBridgmansTables**

calculates base coefficients for bridgmans tables from gibbs enthalpy

**Inputs**

Name	Default	Description
g		dimensionless derivatives of Gibbs function

**Outputs**

Name	Description
v	specific volume [m <sup>3</sup> /kg]
p	pressure [Pa]
T	temperature [K]
s	specific entropy [J/(kg.K)]
cp	heat capacity at constant pressure [J/(kg.K)]
alpha	isobaric volume expansion coefficient [1/K]
gamma	isothermal compressibility [1/Pa]

**Modelica.Media.Common.helmholtzToBridgmansTables**

calculates base coefficients for Bridgmans tables from helmholtz energy

**Inputs**

Name	Default	Description
f		dimensionless derivatives of Helmholtz function

**Outputs**

Name	Description
v	specific volume [m <sup>3</sup> /kg]
p	pressure [Pa]
T	temperature [K]
s	specific entropy [J/(kg.K)]
cp	heat capacity at constant pressure [J/(kg.K)]
alpha	isobaric volume expansion coefficient [1/K]
gamma	isothermal compressibility [1/Pa]

**Modelica.Media.Common.gibbsToBoundaryProps**

calulate phase boundary property record from dimensionless Gibbs function



**Inputs**

Name	Default	Description
g		dimensionless derivatives of Gibbs function

**Outputs**

Name	Description
sat	phase boundary properties

**Modelica.Media.Common.helmholtzToBoundaryProps**

calculate phase boundary property record from dimensionless Helmholtz function

**Inputs**

Name	Default	Description
f		dimensionless derivatives of Helmholtz function

**Outputs**

Name	Description
sat	phase boundary property record

**Modelica.Media.Common.cv2Phase**

compute isochoric specific heat capacity inside the two-phase region

**Inputs**

Name	Default	Description
liq		properties on the boiling curve
vap		properties on the condensation curve
x		vapour mass fraction [1]
T		temperature [K]
p		preoperties [Pa]

**Outputs**

Name	Description
cv	isochoric specific heat capacity [J/(kg.K)]

**Modelica.Media.Common.cvdpT2Phase**

compute isochoric specific heat capacity inside the two-phase region and derivative of pressure w.r.t. temperature

**Inputs**

Name	Default	Description
liq		properties on the boiling curve

## 790 Modelica.Media.Common.cvdpT2Phase

---

vap	properties on the condensation curve
x	vapour mass fraction [1]
T	temperature [K]
p	preoperties [Pa]

### Outputs

Name	Description
cv	isochoric specific heat capacity [J/(kg.K)]
dpT	derivative of pressure w.r.t. temperature

---

## Modelica.Media.Common.gibbsToExtraDerivs

compute additional thermodynamic derivatives from dimensionless Gibbs function



### Inputs

Name	Default	Description
g		dimensionless derivatives of Gibbs function

### Outputs

Name	Description
dpro	additional property derivatives

---

## Modelica.Media.Common.helmholtzToExtraDerivs

compute additional thermodynamic derivatives from dimensionless Helmholtz function



### Inputs

Name	Default	Description
f		dimensionless derivatives of Helmholtz function

### Outputs

Name	Description
dpro	additional property derivatives

---

## Modelica.Media.Common.Helmholtz\_ph

function to calculate analytic derivatives for computing d and t given p and h



### Inputs

Name	Default	Description
f		dimensionless derivatives of Helmholtz function

### Outputs

Name	Description

nderivs	derivatives for Newton iteration to calculate d and t from p and h
---------	--

---

### Modelica.Media.Common.Helmholtz\_pT

function to calculate analytic derivatives for computing d and t given p and t



#### Inputs

Name	Default	Description
f		dimensionless derivatives of Helmholtz function

#### Outputs

Name	Description
nderivs	derivatives for Newton iteration to compute d and t from p and t

---

### Modelica.Media.Common.Helmholtz\_ps

function to calculate analytic derivatives for computing d and t given p and s



#### Inputs

Name	Default	Description
f		dimensionless derivatives of Helmholtz function

#### Outputs

Name	Description
nderivs	derivatives for Newton iteration to compute d and t from p and s

---

### Modelica.Media.Common.OneNonLinearEquation

Determine solution of a non-linear algebraic equation in one unknown without derivatives in a reliable and efficient way

#### Information

This function should currently only be used in Modelica.Media, since it might be replaced in the future by another strategy, where the tool is responsible for the solution of the non-linear equation.

This library determines the solution of one non-linear algebraic equation " $y=f(x)$ " in one unknown "x" in a reliable way. As input, the desired value  $y$  of the non-linear function has to be given, as well as an interval  $x_{\min}, x_{\max}$  that contains the solution, i.e., " $f(x_{\min}) - y$ " and " $f(x_{\max}) - y$ " must have a different sign. If possible, a smaller interval is computed by inverse quadratic interpolation (interpolating with a quadratic polynomial through the last 3 points and computing the zero). If this fails, bisection is used, which always reduces the interval by a factor of 2. The inverse quadratic interpolation method has superlinear convergence. This is roughly the same convergence rate as a globally convergent Newton method, but without the need to compute derivatives of the non-linear function. The solver function is a direct mapping of the Algol 60 procedure "zero" to Modelica, from:

Brent R.P.:

**Algorithms for Minimization without derivatives.** Prentice Hall, 1973, pp. 58-59.

## 792 Modelica.Media.Common.OneNonLinearEquation

---

Due to current limitations of the Modelica language (not possible to pass a function reference to a function), the construction to use this solver on a user-defined function is a bit complicated (this method is from Hans Olsson, Dynasim AB). A user has to provide a package in the following way:

```
package MyNonLinearSolver
  extends OneNonLinearEquation;

  redeclare record extends Data
    // Define data to be passed to user function
    ...
  end Data;

  redeclare function extends f_nonlinear
  algorithm
    // Compute the non-linear equation: y = f(x, Data)
  end f_nonlinear;

  // Dummy definition that has to be present for current Dymola
  redeclare function extends solve
  end solve;
end MyNonLinearSolver;

x_zero = MyNonLinearSolver.solve(y_zero, x_min, x_max, data=data);
```

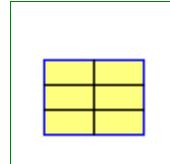
### Package Content

Name	Description
 f_nonlinear_Data	Data specific for function f_nonlinear
 f_nonlinear	Nonlinear algebraic equation in one unknown: $y = f_{\text{nonlinear}}(x, p, X)$
 solve	Solve $f_{\text{nonlinear}}(x_{\text{zero}}) = y_{\text{zero}}$ ; $f_{\text{nonlinear}}(x_{\text{min}}) - y_{\text{zero}}$ and $f_{\text{nonlinear}}(x_{\text{max}}) - y_{\text{zero}}$ must have different sign

---

### Modelica.Media.Common.OneNonLinearEquation.f\_nonlinear\_Data

Data specific for function f\_nonlinear



#### Modelica definition

```
replaceable record f_nonlinear_Data
  "Data specific for function f_nonlinear"
  extends Modelica.Icons.Record;
end f_nonlinear_Data;
```

---

### Modelica.Media.Common.OneNonLinearEquation.f\_nonlinear

Nonlinear algebraic equation in one unknown:  $y = f_{\text{nonlinear}}(x, p, X)$



#### Inputs

Name	Default	Description
x		Independent variable of function

p	0.0	disregarded variables (here always used for pressure)
X[:]	fill(0, 0)	disregarded variables (here always used for composition)
f_nonlinear_data		Additional data for the function

## Outputs

Name	Description
y	= f_nonlinear(x)

---

## Modelica.Media.Common.OneNonLinearEquation.solve

Solve  $f_{\text{nonlinear}}(x_{\text{zero}}) = y_{\text{zero}}$ ;  $f_{\text{nonlinear}}(x_{\text{min}}) - y_{\text{zero}}$  and  $f_{\text{nonlinear}}(x_{\text{max}}) - y_{\text{zero}}$  must have different sign



## Inputs

Name	Default	Description
y_zero		Determine $x_{\text{zero}}$ , such that $f_{\text{nonlinear}}(x_{\text{zero}}) = y_{\text{zero}}$
x_min		Minimum value of x
x_max		Maximum value of x
pressure	0.0	disregarded variables (here always used for pressure)
X[:]	fill(0, 0)	disregarded variables (here always used for composition)
f_nonlinear_data		Additional data for function f_nonlinear
x_tol	100*Modelica.Constants.eps	Relative tolerance of the result

## Outputs

Name	Description
x_zero	$f_{\text{nonlinear}}(x_{\text{zero}}) = y_{\text{zero}}$

---

## Modelica.Media.Air

Medium models for air

## Information

This package contains different medium models for air:

- **SimpleAir**  
Simple dry air medium in a limited temperature range.
- **DryAirNasa**  
Dry air as an ideal gas from Media.IdealGases.MixtureGases.Air.
- **MoistAir**  
Moist air as an ideal gas mixture of steam and dry air with fog below and above the triple point temperature.

## Package Content

Name	Description
	SimpleAir Air: Simple dry air model (0..100 degC)
	DryAirNasa Air: Detailed dry air model as ideal gas (200..6000 K)

 MoistAir	Air: Moist air model (240 ... 400 K)
--	--------------------------------------

## Modelica.Media.Air.SimpleAir

Air: Simple dry air model (0..100 degC)

### Information

#### Simple Ideal gas air model for low temperatures

This model demonstrates how to use the PartialSimpleIdealGas base class to build a simple ideal gas model with a limited temperature validity range.

### Package Content

Name	Description
fluidConstants=FluidConstants(iupacName={"simple air"}, casRegistryNumber={"not a real substance"}, chemicalFormula={"N2, O2"}, structureFormula={"N2, O2"}, molarMass=Modelica.Media.IdealGases.Common.SingleGasesData.N2.MM)	constant data for the fluid
<b>Inherited</b>	
cp_const	Constant specific heat capacity at constant pressure
cv_const=cp_const - R_gas	Constant specific heat capacity at constant volume
R_gas	medium specific gas constant
MM_const	Molar mass
eta_const	Constant dynamic viscosity
lambda_const	Constant thermal conductivity
T_min	Minimum temperature valid for medium model
T_max	Maximum temperature valid for medium model
T0=reference_T	Zero enthalpy temperature
 ThermodynamicState	Thermodynamic state of ideal gas
 BaseProperties	Base properties of ideal gas
(f) setState_pTX	Return thermodynamic state from p, T, and X or Xi
(f) setState_phX	Return thermodynamic state from p, h, and X or Xi
(f) setState_psX	Return thermodynamic state from p, s, and X or Xi
(f) setState_dTX	Return thermodynamic state from d, T, and X or Xi
(f) pressure	Return pressure of ideal gas
(f) temperature	Return temperature of ideal gas
(f) density	Return density of ideal gas

(f) specificEnthalpy	Return specific enthalpy
(f) specificInternalEnergy	Return specific internal energy
(f) specificEntropy	Return specific entropy
(f) specificGibbsEnergy	Return specific Gibbs energy
(f) specificHelmholtzEnergy	Return specific Helmholtz energy
(f) dynamicViscosity	Return dynamic viscosity
(f) thermalConductivity	Return thermal conductivity
(f) specificHeatCapacityCp	Return specific heat capacity at constant pressure
(f) specificHeatCapacityCv	Return specific heat capacity at constant volume
(f) isentropicExponent	Return isentropic exponent
(f) velocityOfSound	Return velocity of sound
(f) specificEnthalpy_pTX	Return specific enthalpy from p, T, and X or Xi
(f) temperature_phX	Return temperature from p, h, and X or Xi
(f) density_phX	Return density from p, h, and X or Xi
(f) setState_pT	Return thermodynamic state from p and T
(f) setState_ph	Return thermodynamic state from p and h
(f) setState_ps	Return thermodynamic state from p and s
(f) setState_dT	Return thermodynamic state from d and T
(f) density_ph	Return density from p and h
(f) temperature_ph	Return temperature from p and h
(f) pressure_dT	Return pressure from d and T
(f) specificEnthalpy_dT	Return specific enthalpy from d and T
(f) specificEnthalpy_ps	Return specific enthalpy from p and s
(f) temperature_ps	Return temperature from p and s
(f) density_ps	Return density from p and s
(f) specificEnthalpy_pT	Return specific enthalpy from p and T
(f) density_pT	Return density from p and T
mediumName="unusablePartialMedium"	Name of the medium
substanceNames={mediumName}	Names of the mixture substances. Set substanceNames={mediumName} if only one substance.
extraPropertiesNames=fill("", 0)	Names of the additional (extra)

	transported properties. Set extraPropertiesNames=fill("",0) if unused
singleState	= true, if u and d are not a function of pressure
reducedX=true	= true if medium contains the equation sum(X) = 1.0; set reducedX=true if only one substance (see docu for details)
fixedX=false	= true if medium contains the equation X = reference_X
reference_p=101325	Reference pressure of Medium: default 1 atmosphere
reference_T=298.15	Reference temperature of Medium: default 25 deg Celsius
reference_X=if nX == 0 then fill(0, nX) else fill(1/nX, nX)	Default mass fractions of medium
p_default=101325	Default value for pressure of medium (for initialization)
T_default=Modelica.SIunits.Conversions.from_degC(20)	Default value for temperature of medium (for initialization)
h_default=specificEnthalpy_pTX(p_default, T_default, X_default)	Default value for specific enthalpy of medium (for initialization)
X_default=reference_X	Default value for mass fractions of medium (for initialization)
nS=size(substanceNames, 1)	Number of substances
nX=if nS == 1 then 0 else nS	Number of mass fractions (= 0, if only one substance)
nXi=if fixedX then 0 else if reducedX then nS - 1 else nX	Number of structurally independent mass fractions (see docu for details)
nC=size(extraPropertiesNames, 1)	Number of extra (outside of standard mass-balance) transported properties
 FluidConstants	critical, triple, molecular and other standard data of fluid
 BasePropertiesRecord	Variables contained in every instance of BaseProperties
 prandtlNumber	Return the Prandtl number
 heatCapacity_cp	alias for deprecated name
 heatCapacity_cv	alias for deprecated name
 isentropicEnthalpy	Return isentropic enthalpy
 isobaricExpansionCoefficient	Return overall the isobaric expansion coefficient beta
 beta	alias for isobaricExpansionCoefficient for user convenience
 isothermalCompressibility	Return overall the isothermal compressibility factor
 kappa	alias of isothermalCompressibility for user convenience

(f) <code>density_derP_h</code>	Return density derivative wrt pressure at const specific enthalpy
(f) <code>density_derH_p</code>	Return density derivative wrt specific enthalpy at constant pressure
(f) <code>density_derP_T</code>	Return density derivative wrt pressure at const temperature
(f) <code>density_derT_p</code>	Return density derivative wrt temperature at constant pressure
(f) <code>density_derX</code>	Return density derivative wrt mass fraction
(f) <code>molarMass</code>	Return the molar mass of the medium
(f) <code>density_pTX</code>	Return density from p, T, and X or Xi
(f) <code>temperature_psX</code>	Return temperature from p,s, and X or Xi
(f) <code>density_psX</code>	Return density from p, s, and X or Xi
(f) <code>specificEnthalpy_psX</code>	Return specific enthalpy from p, s, and X or Xi
<code>AbsolutePressure</code>	Type for absolute pressure with medium specific attributes
<code>Density</code>	Type for density with medium specific attributes
<code>DynamicViscosity</code>	Type for dynamic viscosity with medium specific attributes
<code>EnthalpyFlowRate</code>	Type for enthalpy flow rate with medium specific attributes
<code>MassFlowRate</code>	Type for mass flow rate with medium specific attributes
<code>MassFraction</code>	Type for mass fraction with medium specific attributes
<code>MoleFraction</code>	Type for mole fraction with medium specific attributes
<code>MolarMass</code>	Type for molar mass with medium specific attributes
<code>MolarVolume</code>	Type for molar volume with medium specific attributes
<code>IsentropicExponent</code>	Type for isentropic exponent with medium specific attributes
<code>SpecificEnergy</code>	Type for specific energy with medium specific attributes
<code>SpecificInternalEnergy</code>	Type for specific internal energy with medium specific attributes
<code>SpecificEnthalpy</code>	Type for specific enthalpy with medium specific attributes
<code>SpecificEntropy</code>	Type for specific entropy with medium specific attributes
<code>SpecificHeatCapacity</code>	Type for specific heat capacity with medium specific attributes

SurfaceTension	Type for surface tension with medium specific attributes
Temperature	Type for temperature with medium specific attributes
ThermalConductivity	Type for thermal conductivity with medium specific attributes
PrandtlNumber	Type for Prandtl number with medium specific attributes
VelocityOfSound	Type for velocity of sound with medium specific attributes
ExtraProperty	Type for unspecified, mass-specific property transported by flow
CumulativeExtraProperty	Type for conserved integral of unspecified, mass specific property
ExtraPropertyFlowRate	Type for flow rate of unspecified, mass-specific property
IsobaricExpansionCoefficient	Type for isobaric expansion coefficient with medium specific attributes
DipoleMoment	Type for dipole moment with medium specific attributes
DerDensityByPressure	Type for partial derivative of density with respect to pressure with medium specific attributes
DerDensityByEnthalpy	Type for partial derivative of density with respect to enthalpy with medium specific attributes
DerEnthalpyByPressure	Type for partial derivative of enthalpy with respect to pressure with medium specific attributes
DerDensityByTemperature	Type for partial derivative of density with respect to temperature with medium specific attributes
 Choices	Types, constants to define menu choices

### Types and constants

```

constant FluidConstants[nS] fluidConstants=
  FluidConstants(iupacName={"simple air"},
                 casRegistryNumber={"not a real substance"},
                 chemicalFormula={"N2, O2"},
                 structureFormula={"N2, O2"},
                 molarMass=Modelica.Media.IdealGases.Common.SingleGasesData.N2
                 .MM)
  "constant data for the fluid";

```

---

### Modelica.Media.Air.DryAirNasa

Air: Detailed dry air model as ideal gas (200..6000 K)

## Information

Ideal gas medium model for dry air based on the package [IdealGases](#) with additional functions for dynamic viscosity and thermal conductivity in a limited temperature range.

## Package Content

Name	Description
<a href="#">dynamicViscosity</a>	Simple polynomial for dry air (moisture influence small), valid from 73.15 K to 373.15 K
<a href="#">thermalConductivity</a>	Simple polynomial for dry air (moisture influence small), valid from 73.15 K to 373.15 K
<b>Inherited</b>	
<a href="#">ThermodynamicState</a>	thermodynamic state variables for ideal gases
<a href="#">FluidConstants</a>	extended fluid constants
<code>excludeEnthalpyOfFormation=true</code>	If true, enthalpy of formation Hf is not included in specific enthalpy h
<code>referenceChoice=Choices.ReferenceEnthalpy.ZeroAt0K</code>	Choice of reference enthalpy
<code>h_offset=0.0</code>	User defined offset for reference enthalpy, if referenceChoice = UserDefined
<code>data</code>	Data record of ideal gas substance
<code>fluidConstants</code>	constant data for the fluid
<a href="#">BaseProperties</a>	Base properties of ideal gas medium
<a href="#">setState_pTX</a>	Return thermodynamic state as function of p, T and composition X
<a href="#">setState_phX</a>	Return thermodynamic state as function of p, h and composition X
<a href="#">setState_psX</a>	Return thermodynamic state as function of p, s and composition X
<a href="#">setState_dTX</a>	Return thermodynamic state as function of d, T and composition X
<a href="#">pressure</a>	return pressure of ideal gas
<a href="#">temperature</a>	return temperature of ideal gas
<a href="#">density</a>	return density of ideal gas
<a href="#">specificEnthalpy</a>	Return specific enthalpy
<a href="#">specificInternalEnergy</a>	Return specific internal energy
<a href="#">specificEntropy</a>	Return specific entropy
<a href="#">specificGibbsEnergy</a>	Return specific Gibbs energy
<a href="#">specificHelmholtzEnergy</a>	Return specific Helmholtz energy
<a href="#">specificHeatCapacityCp</a>	Return specific heat capacity at constant pressure
<a href="#">specificHeatCapacityCv</a>	Compute specific heat capacity at constant volume from temperature and gas data
<a href="#">isentropicExponent</a>	Return isentropic exponent
<a href="#">velocityOfSound</a>	Return velocity of sound
<a href="#">isentropicEnthalpyApproximation</a>	approximate method of calculating h_is from upstream

	properties and downstream pressure
(f) isentropicEnthalpy	Return isentropic enthalpy
(f) isobaricExpansionCoefficient	Returns overall the isobaric expansion coefficient beta
(f) isothermalCompressibility	Returns overall the isothermal compressibility factor
(f) density_derP_T	density derivative by temperature at constant pressure
(f) density_derT_p	density derivative by temperature at constant pressure
(f) density_derX	density derivative by mass fraction
(f) cp_T	Compute specific heat capacity at constant pressure from temperature and gas data
(f) cp_Tlow	Compute specific heat capacity at constant pressure, low T region
(f) cp_Tlow_der	Compute specific heat capacity at constant pressure, low T region
(f) h_T	Compute specific enthalpy from temperature and gas data; reference is decided by the refChoice input, or by the referenceChoice package constant by default
(f) h_T_der	derivative function for h_T
(f) h_Tlow	Compute specific enthalpy, low T region; reference is decided by the refChoice input, or by the referenceChoice package constant by default
(f) h_Tlow_der	Compute specific enthalpy, low T region; reference is decided by the refChoice input, or by the referenceChoice package constant by default
(f) s0_T	Compute specific entropy from temperature and gas data
(f) s0_Tlow	Compute specific entropy, low T region
(f) dynamicViscosityLowPressure	Dynamic viscosity of low pressure gases
(f) thermalConductivityEstimate	Thermal conductivity of polyatomic gases(Eucken and Modified Eucken correlation)
(f) molarMass	return the molar mass of the medium
(f) T_h	Compute temperature from specific enthalpy
(f) T_ps	Compute temperature from pressure and specific entropy
(f) setState_pT	Return thermodynamic state from p and T
(f) setState_ph	Return thermodynamic state from p and h
(f) setState_ps	Return thermodynamic state from p and s
(f) setState_dT	Return thermodynamic state from d and T
(f) density_ph	Return density from p and h
(f) temperature_ph	Return temperature from p and h
(f) pressure_dT	Return pressure from d and T
(f) specificEnthalpy_dT	Return specific enthalpy from d and T
(f) specificEnthalpy_ps	Return specific enthalpy from p and s
(f) temperature_ps	Return temperature from p and s
(f) density_ps	Return density from p and s

 <a href="#">specificEnthalpy_pT</a>	Return specific enthalpy from p and T
 <a href="#">density_pT</a>	Return density from p and T
mediumName="unusablePartialMedium"	Name of the medium
substanceNames={mediumName}	Names of the mixture substances. Set substanceNames={mediumName} if only one substance.
extraPropertiesNames=fill("", 0)	Names of the additional (extra) transported properties. Set extraPropertiesNames=fill("",0) if unused
singleState	= true, if u and d are not a function of pressure
reducedX=true	= true if medium contains the equation sum(X) = 1.0; set reducedX=true if only one substance (see docu for details)
fixedX=false	= true if medium contains the equation X = reference_X
reference_p=101325	Reference pressure of Medium: default 1 atmosphere
reference_T=298.15	Reference temperature of Medium: default 25 deg Celsius
reference_X=if nX == 0 then fill(0, nX) else fill(1/nX, nX)	Default mass fractions of medium
p_default=101325	Default value for pressure of medium (for initialization)
T_default=Modelica.SIunits.Conversions.from_degC(20)	Default value for temperature of medium (for initialization)
h_default=specificEnthalpy_pTX(p_default, T_default, X_default)	Default value for specific enthalpy of medium (for initialization)
X_default=reference_X	Default value for mass fractions of medium (for initialization)
nS=size(substanceNames, 1)	Number of substances
nX=if nS == 1 then 0 else nS	Number of mass fractions (= 0, if only one substance)
nXi=if fixedX then 0 else if reducedX then nS - 1 else nX	Number of structurally independent mass fractions (see docu for details)
nC=size(extraPropertiesNames, 1)	Number of extra (outside of standard mass-balance) transported properties
 <a href="#">BasePropertiesRecord</a>	Variables contained in every instance of BaseProperties
 <a href="#">prandtlNumber</a>	Return the Prandtl number
 <a href="#">heatCapacity_cp</a>	alias for deprecated name
 <a href="#">heatCapacity_cv</a>	alias for deprecated name
 <a href="#">beta</a>	alias for isobaricExpansionCoefficient for user convenience
 <a href="#">kappa</a>	alias of isothermalCompressibility for user convenience
 <a href="#">density_derh_h</a>	Return density derivative wrt pressure at const specific enthalpy
 <a href="#">density_derh_p</a>	Return density derivative wrt specific enthalpy at constant pressure
 <a href="#">specificEnthalpy_pTX</a>	Return specific enthalpy from p, T, and X or Xi
 <a href="#">density_pTX</a>	Return density from p, T, and X or Xi
 <a href="#">temperature_phX</a>	Return temperature from p, h, and X or Xi
 <a href="#">density_phX</a>	Return density from p, h, and X or Xi
 <a href="#">temperature_psX</a>	Return temperature from p,s, and X or Xi
 <a href="#">density_psX</a>	Return density from p, s, and X or Xi
 <a href="#">specificEnthalpy_psX</a>	Return specific enthalpy from p, s, and X or Xi

AbsolutePressure	Type for absolute pressure with medium specific attributes
Density	Type for density with medium specific attributes
DynamicViscosity	Type for dynamic viscosity with medium specific attributes
EnthalpyFlowRate	Type for enthalpy flow rate with medium specific attributes
MassFlowRate	Type for mass flow rate with medium specific attributes
MassFraction	Type for mass fraction with medium specific attributes
MoleFraction	Type for mole fraction with medium specific attributes
MolarMass	Type for molar mass with medium specific attributes
MolarVolume	Type for molar volume with medium specific attributes
IsentropicExponent	Type for isentropic exponent with medium specific attributes
SpecificEnergy	Type for specific energy with medium specific attributes
SpecificInternalEnergy	Type for specific internal energy with medium specific attributes
SpecificEnthalpy	Type for specific enthalpy with medium specific attributes
SpecificEntropy	Type for specific entropy with medium specific attributes
SpecificHeatCapacity	Type for specific heat capacity with medium specific attributes
SurfaceTension	Type for surface tension with medium specific attributes
Temperature	Type for temperature with medium specific attributes
ThermalConductivity	Type for thermal conductivity with medium specific attributes
PrandtlNumber	Type for Prandtl number with medium specific attributes
VelocityOfSound	Type for velocity of sound with medium specific attributes
ExtraProperty	Type for unspecified, mass-specific property transported by flow
CumulativeExtraProperty	Type for conserved integral of unspecified, mass specific property
ExtraPropertyFlowRate	Type for flow rate of unspecified, mass-specific property
IsobaricExpansionCoefficient	Type for isobaric expansion coefficient with medium specific attributes
DipoleMoment	Type for dipole moment with medium specific attributes
DerDensityByPressure	Type for partial derivative of density with respect to pressure with medium specific attributes
DerDensityByEnthalpy	Type for partial derivative of density with respect to enthalpy with medium specific attributes
DerEnthalpyByPressure	Type for partial derivative of enthalpy with respect to pressure with medium specific attributes
DerDensityByTemperature	Type for partial derivative of density with respect to temperature with medium specific attributes
 Choices	Types, constants to define menu choices

### Modelica.Media.Air.DryAirNASA.dynamicViscosity

Simple polynomial for dry air (moisture influence small), valid from 73.15 K to 373.15 K



### Information

Dynamic viscosity is computed from temperature using a second order polynomial with a range of validity between 73 and 373 K.

## Inputs

Name	Default	Description
state		Thermodynamic state record

## Outputs

Name	Description
eta	Dynamic viscosity [Pa.s]

## Modelica.Media.Air.DryAirNasa.thermalConductivity

Simple polynomial for dry air (moisture influence small), valid from 73.15 K to 373.15 K



## Information

Thermal conductivity is computed from temperature using a second order polynomial with a range of validity between 73 and 373 K.

## Inputs

Name	Default	Description
state		Thermodynamic state record
method	1	Dummy for compatibility reasons

## Outputs

Name	Description
lambda	Thermal conductivity [W/(m.K)]

## Modelica.Media.Air.MoistAir

Air: Moist air model (240 ... 400 K)

## Information

### Thermodynamic Model

This package provides a full thermodynamic model of moist air including the fog region and temperatures below zero degC. The governing assumptions in this model are:

- the perfect gas law applies
- water volume other than that of steam is neglected

All extensive properties are expressed in terms of the total mass in order to comply with other media in this library. However, for moist air it is rather common to express the absolute humidity in terms of mass of dry air only, which has advantages when working with charts. In addition, care must be taken, when working with mass fractions with respect to total mass, that all properties refer to the same water content when being used in mathematical operations (which is always the case if based on dry air only). Therefore two absolute humidities are computed in the **BaseProperties** model: **X** denotes the absolute humidity in terms of the total mass while **x** denotes the absolute humidity per unit mass of dry air. In addition, the relative humidity **phi** is also computed.

At the triple point temperature of water of 0.01°C or 273.16 K and a relative humidity greater than 1 fog may be present as liquid and as ice resulting in a specific enthalpy somewhere between those of the two

isotherms for solid and liquid fog, respectively. For numerical reasons a coexisting mixture of 50% solid and 50% liquid fog is assumed in the fog region at the triple point in this model.

### Range of validity

From the assumptions mentioned above it follows that the **pressure** should be in the region around **atmospheric** conditions or below (a few bars may still be fine though). Additionally a very high water content at low temperatures would yield incorrect densities, because the volume of the liquid or solid phase would not be negligible anymore. The model does not provide information on limits for water drop size in the fog region or transport information for the actual condensation or evaporation process in combination with surfaces. All excess water which is not in its vapour state is assumed to be still present in the air regarding its energy but not in terms of its spatial extent.

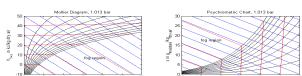
The thermodynamic model may be used for **temperatures** ranging from **240 - 400 K**. This holds for all functions unless otherwise stated in their description. However, although the model works at temperatures above the saturation temperature it is questionable to use the term "relative humidity" in this region. Please note, that although several functions compute pure water properties, they are designed to be used within the moist air medium model where properties are dominated by air and steam in their vapor states, and not for pure liquid water applications.

### Transport Properties

Several additional functions that are not needed to describe the thermodynamic system, but are required to model transport processes, like heat and mass transfer, may be called. They usually neglect the moisture influence unless otherwise stated.

### Application

The model's main area of application is all processes that involve moist air cooling under near atmospheric pressure with possible moisture condensation. This is the case in all domestic and industrial air conditioning applications. Another large domain of moist air applications covers all processes that deal with dehydration of bulk material using air as a transport medium. Engineering tasks involving moist air are often performed (or at least visualized) by using charts that contain all relevant thermodynamic data for a moist air system. These so called psychrometric charts can be generated from the medium properties in this package. The model **PsychrometricData** may be used for this purpose in order to obtain data for figures like those below (the plotting itself is not part of the model though).



**Legend:** blue - constant specific enthalpy, red - constant temperature, black - constant relative humidity

### Package Content

Name	Description
Water=1	Index of water (in substanceNames, massFractions X, etc.)
Air=2	Index of air (in substanceNames, massFractions X, etc.)
k_mair=steam.MM/dryair.MM	ratio of molar weights
dryair=IdealGases.Common.SingleGasesData.Air	
steam=IdealGases.Common.SingleGasesData.H2O	
<input type="checkbox"/> BaseProperties	Moist air base properties record
<input checked="" type="checkbox"/> setState_pTX	Return thermodynamic state as function of pressure p, temperature T and composition X

(f) <code>setState_phX</code>	Return thermodynamic state as function of pressure p, specific enthalpy h and composition X
(f) <code>setState_dTX</code>	Return thermodynamic state as function of density d, temperature T and composition X
(f) <code>Xsaturation</code>	Return absolute humidity per unit mass of moist air at saturation as a function of the thermodynamic state record
(f) <code>xsaturation</code>	Return absolute humidity per unit mass of dry air at saturation as a function of the thermodynamic state record
(f) <code>xsaturation_pT</code>	Return absolute humidity per unit mass of dry air at saturation as a function of pressure p and temperature T
(f) <code>massFraction_pTphi</code>	Return steam mass fraction as a function of relative humidity phi and temperature T
(f) <code>relativeHumidity_pTX</code>	Return relative humidity as a function of pressure p, temperature T and composition X
(f) <code>relativeHumidity</code>	Return relative humidity as a function of the thermodynamic state record
(f) <code>gasConstant</code>	Return ideal gas constant as a function from thermodynamic state, only valid for phi<1
(f) <code>gasConstant_X</code>	Return ideal gas constant as a function from composition X
(f) <code>saturationPressureLiquid</code>	Return saturation pressure of water as a function of temperature T in the range of 273.16 to 373.16 K
(f) <code>saturationPressureLiquid_der</code>	Time derivative of saturationPressureLiquid
(f) <code>sublimationPressureIce</code>	Return sublimation pressure of water as a function of temperature T between 223.16 and 273.16 K
(f) <code>sublimationPressureIce_der</code>	Derivative function for 'sublimationPressureIce'
(f) <code>saturationPressure</code>	Return saturation pressure of water as a function of temperature T between 223.16 and 373.16 K
(f) <code>saturationPressure_der</code>	Derivative function for 'saturationPressure'
(f) <code>saturationTemperature</code>	Return saturation temperature of water as a function of (partial) pressure p
(f) <code>enthalpyOfVaporization</code>	Return enthalpy of vaporization of water as a function of temperature T, 0 - 130 degC
(f) <code>HeatCapacityOfWater</code>	Return specific heat capacity of water (liquid only) as a function of temperature T
(f) <code>enthalpyOfLiquid</code>	Return enthalpy of liquid water as a function of temperature T (use enthalpyOfWater instead)
(f) <code>enthalpyOfGas</code>	Return specific enthalpy of gas (air and steam) as a function of temperature T and composition X
(f) <code>enthalpyOfCondensingGas</code>	Return specific enthalpy of steam as a function of temperature T
(f) <code>enthalpyOfWater</code>	Computes specific enthalpy of water (solid/liquid) near atmospheric pressure from temperature T
(f) <code>enthalpyOfWater_der</code>	Derivative function of enthalpyOfWater
(f) <code>pressure</code>	Returns pressure of ideal gas as a function of the thermodynamic state record
(f) <code>temperature</code>	Return temperature of ideal gas as a function of the thermodynamic state record

(f) <code>T_phX</code>	Return temperature as a function of pressure p, specific enthalpy h and composition X
(f) <code>density</code>	Returns density of ideal gas as a function of the thermodynamic state record
(f) <code>specificEnthalpy</code>	Return specific enthalpy of moist air as a function of the thermodynamic state record
(f) <code>h_pTX</code>	Return specific enthalpy of moist air as a function of pressure p, temperature T and composition X
(f) <code>h_pTX_der</code>	Derivative function of <code>h_pTX</code>
(f) <code>specificInternalEnergy</code>	Return specific internal energy of moist air as a function of the thermodynamic state record
(f) <code>specificInternalEnergy_pTX</code>	Return specific internal energy of moist air as a function of pressure p, temperature T and composition X
(f) <code>specificInternalEnergy_pTX_der</code>	Derivative function for <code>specificInternalEnergy_pTX</code>
(f) <code>specificEntropy</code>	Return specific entropy from thermodynamic state record, only valid for $\phi < 1$
(f) <code>specificGibbsEnergy</code>	Return specific Gibbs energy as a function of the thermodynamic state record, only valid for $\phi < 1$
(f) <code>specificHelmholtzEnergy</code>	Return specific Helmholtz energy as a function of the thermodynamic state record, only valid for $\phi < 1$
(f) <code>specificHeatCapacityCp</code>	Return specific heat capacity at constant pressure as a function of the thermodynamic state record
(f) <code>specificHeatCapacityCv</code>	Return specific heat capacity at constant volume as a function of the thermodynamic state record
(f) <code>dynamicViscosity</code>	Return dynamic viscosity as a function of the thermodynamic state record, valid from 73.15 K to 373.15 K
(f) <code>thermalConductivity</code>	Return thermal conductivity as a function of the thermodynamic state record, valid from 73.15 K to 373.15 K
<input type="checkbox"/> <code>Utilities</code>	utility functions
<input type="checkbox"/> <code>PsychrometricData</code>	Produces plot data for psychrometric charts
<b>Inherited</b>	
<input type="checkbox"/> <code>ThermodynamicState</code>	thermodynamic state variables
<input type="checkbox"/> <code>FluidConstants</code>	extended fluid constants
<code>fluidConstants</code>	constant data for the fluid
(f) <code>moleToMassFractions</code>	Return mass fractions X from mole fractions
(f) <code>massToMoleFractions</code>	Return mole fractions from mass fractions X
<code>mediumName="unusablePartialMedium"</code>	Name of the medium
<code>substanceNames={mediumName}</code>	Names of the mixture substances. Set <code>substanceNames={mediumName}</code> if only one substance.
<code>extraPropertiesNames=fill("", 0)</code>	Names of the additional (extra) transported properties. Set <code>extraPropertiesNames=fill("", 0)</code> if unused
<code>singleState</code>	= true, if u and d are not a function of pressure
<code>reducedX=true</code>	= true if medium contains the equation $\sum(X) = 1.0$ ; set <code>reducedX=true</code> if only one substance (see docu for details)

fixedX=false	= true if medium contains the equation X = reference_X
reference_p=101325	Reference pressure of Medium: default 1 atmosphere
reference_T=298.15	Reference temperature of Medium: default 25 deg Celsius
reference_X=if nX == 0 then fill(0, nX) else fill(1/nX, nX)	Default mass fractions of medium
p_default=101325	Default value for pressure of medium (for initialization)
T_default=Modelica.SIunits.Conversions.from_deg_C(20)	Default value for temperature of medium (for initialization)
h_default=specificEnthalpy_pTX(p_default, T_default, X_default)	Default value for specific enthalpy of medium (for initialization)
X_default=reference_X	Default value for mass fractions of medium (for initialization)
nS=size(substanceNames, 1)	Number of substances
nX=if nS == 1 then 0 else nS	Number of mass fractions (= 0, if only one substance)
nXi=if fixedX then 0 else if reducedX then nS - 1 else nX	Number of structurally independent mass fractions (see docu for details)
nC=size(extraPropertiesNames, 1)	Number of extra (outside of standard mass-balance) transported properties
 BasePropertiesRecord	Variables contained in every instance of BaseProperties
(f) setState_psX	Return thermodynamic state as function of p, s and composition X or Xi
(f) prandtlNumber	Return the Prandtl number
(f) heatCapacity_cp	alias for deprecated name
(f) heatCapacity_cv	alias for deprecated name
(f) isentropicExponent	Return isentropic exponent
(f) isentropicEnthalpy	Return isentropic enthalpy
(f) velocityOfSound	Return velocity of sound
(f) isobaricExpansionCoefficient	Return overall the isobaric expansion coefficient beta
(f) beta	alias for isobaricExpansionCoefficient for user convenience
(f) isothermalCompressibility	Return overall the isothermal compressibility factor
(f) kappa	alias of isothermalCompressibility for user convenience
(f) density_derh_h	Return density derivative wrt pressure at const specific enthalpy
(f) density_derh_p	Return density derivative wrt specific enthalpy at constant pressure
(f) density_derh_T	Return density derivative wrt pressure at const temperature
(f) density_derT_p	Return density derivative wrt temperature at constant pressure
(f) density_derX	Return density derivative wrt mass fraction
(f) molarMass	Return the molar mass of the medium
(f) specificEnthalpy_pTX	Return specific enthalpy from p, T, and X or Xi
(f) density_pTX	Return density from p, T, and X or Xi

(f) <code>temperature_phX</code>	Return temperature from p, h, and X or Xi
(f) <code>density_phX</code>	Return density from p, h, and X or Xi
(f) <code>temperature_psX</code>	Return temperature from p,s, and X or Xi
(f) <code>density_psX</code>	Return density from p, s, and X or Xi
(f) <code>specificEnthalpy_psX</code>	Return specific enthalpy from p, s, and X or Xi
<code>AbsolutePressure</code>	Type for absolute pressure with medium specific attributes
<code>Density</code>	Type for density with medium specific attributes
<code>DynamicViscosity</code>	Type for dynamic viscosity with medium specific attributes
<code>EnthalpyFlowRate</code>	Type for enthalpy flow rate with medium specific attributes
<code>MassFlowRate</code>	Type for mass flow rate with medium specific attributes
<code>MassFraction</code>	Type for mass fraction with medium specific attributes
<code>MoleFraction</code>	Type for mole fraction with medium specific attributes
<code>MolarMass</code>	Type for molar mass with medium specific attributes
<code>MolarVolume</code>	Type for molar volume with medium specific attributes
<code>IsentropicExponent</code>	Type for isentropic exponent with medium specific attributes
<code>SpecificEnergy</code>	Type for specific energy with medium specific attributes
<code>SpecificInternalEnergy</code>	Type for specific internal energy with medium specific attributes
<code>SpecificEnthalpy</code>	Type for specific enthalpy with medium specific attributes
<code>SpecificEntropy</code>	Type for specific entropy with medium specific attributes
<code>SpecificHeatCapacity</code>	Type for specific heat capacity with medium specific attributes
<code>SurfaceTension</code>	Type for surface tension with medium specific attributes
<code>Temperature</code>	Type for temperature with medium specific attributes
<code>ThermalConductivity</code>	Type for thermal conductivity with medium specific attributes
<code>PrandtlNumber</code>	Type for Prandtl number with medium specific attributes
<code>VelocityOfSound</code>	Type for velocity of sound with medium specific attributes
<code>ExtraProperty</code>	Type for unspecified, mass-specific property transported by flow
<code>CumulativeExtraProperty</code>	Type for conserved integral of unspecified, mass specific property
<code>ExtraPropertyFlowRate</code>	Type for flow rate of unspecified, mass-specific property
<code>IsobaricExpansionCoefficient</code>	Type for isobaric expansion coefficient with medium specific attributes
<code>DipoleMoment</code>	Type for dipole moment with medium specific attributes
<code>DerDensityByPressure</code>	Type for partial derivative of density with respect to pressure with medium specific attributes
<code>DerDensityByEnthalpy</code>	Type for partial derivative of density with respect to enthalpy with medium specific attributes
<code>DerEnthalpyByPressure</code>	Type for partial derivative of enthalpy with respect to pressure with medium specific attributes

DerDensityByTemperature	Type for partial derivative of density with respect to temperature with medium specific attributes
Choices	Types, constants to define menu choices

## Types and constants

```

constant Integer Water=1
"Index of water (in substanceNames, massFractions X, etc.)";

constant Integer Air=2
"Index of air (in substanceNames, massFractions X, etc.)";

constant Real k_mair = steam.MM/dryair.MM "ratio of molar weights";

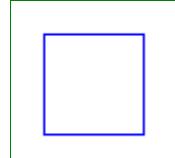
constant IdealGases.Common.DataRecord dryair =
IdealGases.Common.SingleGasesData.Air;

constant IdealGases.Common.DataRecord steam =
IdealGases.Common.SingleGasesData.H2O;

```

## Modelica.Media.Air.MoistAir.BaseProperties

### Moist air base properties record



#### Information

This model computes thermodynamic properties of moist air from three independent (thermodynamic or/and numerical) state variables. Preferred numerical states are temperature T, pressure p and the reduced composition vector  $\bar{X}_i$ , which contains the water mass fraction only. As an EOS the **ideal gas law** is used and associated restrictions apply. The model can also be used in the **fog region**, when moisture is present in its liquid state. However, it is assumed that the liquid water volume is negligible compared to that of the gas phase. Computation of thermal properties is based on property data of **dry air** and water (source: VDI-Wärmeatlas), respectively. Besides the standard thermodynamic variables **absolute and relative humidity**,  $x_{\text{water}}$  and  $\phi$ , respectively, are given by the model. Upper case X denotes absolute humidity with respect to mass of moist air while absolute humidity with respect to mass of dry air only is denoted by a lower case x throughout the model. See [package description](#) for further information.

#### Parameters

Name	Default	Description
standardOrderComponents	true	if true, last element in components is computed from $1 - \sum(X_i)$
<b>Advanced</b>		
preferredMediumStates	false	= true if StateSelect.prefer shall be used for the independent property variables of the medium

## Modelica.Media.Air.MoistAir.setState\_pTX

Return thermodynamic state as function of pressure p, temperature T and composition X



## 810 Modelica.Media.Air.MoistAir.setState\_pTX

---

### Information

The [thermodynamic state record](#) is computed from pressure p, temperature T and composition X.

### Inputs

Name	Default	Description
p		Pressure [Pa]
T		Temperature [K]
X[:]	reference_X	Mass fractions [kg/kg]

### Outputs

Name	Description
state	Thermodynamic state

---

## Modelica.Media.Air.MoistAir.setState\_phX

Return thermodynamic state as function of pressure p, specific enthalpy h and composition X



### Information

The [thermodynamic state record](#) is computed from pressure p, specific enthalpy h and composition X.

### Inputs

Name	Default	Description
p		Pressure [Pa]
h		Specific enthalpy [J/kg]
X[:]	reference_X	Mass fractions [kg/kg]

### Outputs

Name	Description
state	Thermodynamic state

---

## Modelica.Media.Air.MoistAir.setState\_dTX

Return thermodynamic state as function of density d, temperature T and composition X



### Information

The [thermodynamic state record](#) is computed from density d, temperature T and composition X.

### Inputs

Name	Default	Description
d		density [kg/m <sup>3</sup> ]
T		Temperature [K]
X[:]	reference_X	Mass fractions [kg/kg]

## Outputs

Name	Description
state	Thermodynamic state

---

## Modelica.Media.Air.MoistAir.Xsaturation

Return absolute humidity per unit mass of moist air at saturation as a function of the thermodynamic state record

## Information

Absolute humidity per unit mass of moist air at saturation is computed from pressure and temperature in the state record. Note, that unlike X\_sat in the BaseProperties model this mass fraction refers to mass of moist air at saturation.

## Inputs

Name	Default	Description
state		Thermodynamic state record

## Outputs

Name	Description
X_sat	Steam mass fraction of sat. boundary [kg/kg]

---

## Modelica.Media.Air.MoistAir.xsaturation

Return absolute humidity per unit mass of dry air at saturation as a function of the thermodynamic state record

## Information

Absolute humidity per unit mass of dry air at saturation is computed from pressure and temperature in the thermodynamic state record.

## Inputs

Name	Default	Description
state		Thermodynamic state record

## Outputs

Name	Description
x_sat	Absolute humidity per unit mass of dry air [kg/kg]

---

## Modelica.Media.Air.MoistAir.xsaturation\_pT

Return absolute humidity per unit mass of dry air at saturation as a function of pressure p and temperature T

## 812 Modelica.Media.Air.MoistAir.xsaturation\_pT

---

### Information

Absolute humidity per unit mass of dry air at saturation is computed from pressure and temperature.

### Inputs

Name	Default	Description
p		Pressure [Pa]
T		Temperature [K]

### Outputs

Name	Description
x_sat	Absolute humidity per unit mass of dry air [kg/kg]

---

## Modelica.Media.Air.MoistAir.massFraction\_pTphi

Return steam mass fraction as a function of relative humidity phi and temperature T

### Information

Absolute humidity per unit mass of moist air is computed from temperature, pressure and relative humidity.

### Inputs

Name	Default	Description
p		Pressure [Pa]
T		Temperature [K]
phi		Relative humidity (0 ... 1.0)

### Outputs

Name	Description
X_steam	Absolute humidity, steam mass fraction [kg/kg]

---

## Modelica.Media.Air.MoistAir.relativeHumidity\_pTX

Return relative humidity as a function of pressure p, temperature T and composition X

### Information

Relative humidity is computed from pressure, temperature and composition with 1.0 as the upper limit at saturation. Water mass fraction is the first entry in the composition vector.

### Inputs

Name	Default	Description
p		Pressure [Pa]
T		Temperature [K]
X[:]		Composition [1]

## Outputs

Name	Description
phi	Relative humidity

---

### Modelica.Media.Air.MoistAir.relativeHumidity

Return relative humidity as a function of the thermodynamic state record

## Information

Relative humidity is computed from the thermodynamic state record with 1.0 as the upper limit at saturation.

## Inputs

Name	Default	Description
state		Thermodynamic state

## Outputs

Name	Description
phi	Relative humidity

---

### Modelica.Media.Air.MoistAir.gasConstant

Return ideal gas constant as a function from thermodynamic state, only valid for phi<1



## Information

The ideal gas constant for moist air is computed from [thermodynamic state](#) assuming that all water is in the gas phase.

## Inputs

Name	Default	Description
state		thermodynamic state

## Outputs

Name	Description
R	mixture gas constant [J/(kg.K)]

---

### Modelica.Media.Air.MoistAir.gasConstant\_X

Return ideal gas constant as a function from composition X

## Information

The ideal gas constant for moist air is computed from the gas phase composition. The first entry in composition vector X is the steam mass fraction of the gas phase.

## Inputs

Name	Default	Description
X[.]		Gas phase composition [1]

## Outputs

Name	Description
R	Ideal gas constant [J/(kg.K)]

---

## Modelica.Media.Air.MoistAir.saturationPressureLiquid

Return saturation pressure of water as a function of temperature T in the range of 273.16 to 373.16 K



## Information

Saturation pressure of water above the triple point temperature is computed from temperature. Its range of validity is between 273.16 and 373.16 K. Outside these limits a less accurate result is returned.

## Inputs

Name	Default	Description
Tsat		saturation temperature [K]

## Outputs

Name	Description
psat	saturation pressure [Pa]

---

## Modelica.Media.Air.MoistAir.saturationPressureLiquid\_der

Time derivative of saturationPressureLiquid



## Information

Derivative function of saturationPressureLiquid

## Inputs

Name	Default	Description
Tsat		Saturation temperature [K]
dTsat		Saturation temperature derivative [K/s]

## Outputs

Name	Description
psat_der	Saturation pressure [Pa/s]

---

## Modelica.Media.Air.MoistAir.sublimationPressureIce

Return sublimation pressure of water as a function of temperature T between 223.16



and 273.16 K

## Information

Sublimation pressure of water below the triple point temperature is computed from temperature. It's range of validity is between 223.16 and 273.16 K. Outside of these limits a less accurate result is returned.

## Inputs

Name	Default	Description
Tsat		sublimation temperature [K]

## Outputs

Name	Description
psat	sublimation pressure [Pa]

## Modelica.Media.Air.MoistAir.sublimationPressureIce\_der



Derivative function for 'sublimationPressureIce'

## Information

Derivative function of [saturationPressureIce](#)

## Inputs

Name	Default	Description
Tsat		Sublimation temperature [K]
dTsat		Time derivative of sublimation temperature [K/s]

## Outputs

Name	Description
psat_der	Sublimation pressure [Pa/s]

## Modelica.Media.Air.MoistAir.saturationPressure



Return saturation pressure of water as a function of temperature T between 223.16 and 373.16 K

## Information

Saturation pressure of water in the liquid and the solid region is computed using an Antoine-type correlation. It's range of validity is between 223.16 and 373.16 K. Outside of these limits a (less accurate) result is returned. Functions for the [solid](#) and the [liquid](#) region, respectively, are combined using the first derivative continuous [spliceFunction](#).

## Inputs

Name	Default	Description
Tsat		saturation temperature [K]

## Outputs

Name	Description
psat	saturation pressure [Pa]

---

## Modelica.Media.Air.MoistAir.saturationPressure\_der

Derivative function for 'saturationPressure'

## Information

Derivative function of saturationPressure

## Inputs

Name	Default	Description
Tsat		Saturation temperature [K]
dTsat		Time derivative of saturation temperature [K/s]

## Outputs

Name	Description
psat_der	Saturation pressure [Pa/s]

---

## Modelica.Media.Air.MoistAir.saturationTemperature

Return saturation temperature of water as a function of (partial) pressure p

## Information

Computes saturation temperature from (partial) pressure via numerical inversion of the function saturationPressure. Therefore additional inputs are required (or the defaults are used) for upper and lower temperature bounds.

## Inputs

Name	Default	Description
p		Pressure [Pa]
T_min	200	Lower boundary of solution [K]
T_max	400	Upper boundary of solution [K]

## Outputs

Name	Description
T	Saturation temperature [K]

---

## Modelica.Media.Air.MoistAir.enthalpyOfVaporization

Return enthalpy of vaporization of water as a function of temperature T, 0 - 130 degC



## Information

Enthalpy of vaporization of water is computed from temperature in the region of 0 to 130 °C.

## Inputs

Name	Default	Description
T		temperature [K]

## Outputs

Name	Description
r0	vaporization enthalpy [J/kg]

## Modelica.Media.Air.MoistAir.HeatCapacityOfWater

Return specific heat capacity of water (liquid only) as a function of temperature T



## Information

The specific heat capacity of water (liquid and solid) is calculated using a polynomial approach and data from VDI-Waermeatlas 8. Edition (Db1)

## Inputs

Name	Default	Description
T		Temperature [K]

## Outputs

Name	Description
cp_fl	Specific heat capacity of liquid [J/(kg.K)]

## Modelica.Media.Air.MoistAir.enthalpyOfLiquid

Return enthalpy of liquid water as a function of temperature T (use enthalpyOfWater instead)



## Information

Specific enthalpy of liquid water is computed from temperature using a polynomial approach. Kept for compatibility reasons, better use [enthalpyOfWater](#) instead.

## Inputs

Name	Default	Description
T		temperature [K]

## Outputs

Name	Description
h	liquid enthalpy [J/kg]

**Modelica.Media.Air.MoistAir.enthalpyOfGas**

Return specific enthalpy of gas (air and steam) as a function of temperature T and composition X

**Information**

Specific enthalpy of moist air is computed from temperature, provided all water is in the gaseous state. The first entry in the composition vector X must be the mass fraction of steam. For a function that also covers the fog region please refer to [h\\_pTX](#).

**Inputs**

Name	Default	Description
T		temperature [K]
X[:]		vector of mass fractions [kg/kg]

**Outputs**

Name	Description
h	liquid enthalpy [J/kg]

**Modelica.Media.Air.MoistAir.enthalpyOfCondensingGas**

Return specific enthalpy of steam as a function of temperature T

**Information**

Specific enthalpy of steam is computed from temperature.

**Inputs**

Name	Default	Description
T		temperature [K]

**Outputs**

Name	Description
h	liquid enthalpy [J/kg]

**Modelica.Media.Air.MoistAir.enthalpyOfWater**

Computes specific enthalpy of water (solid/liquid) near atmospheric pressure from temperature T

**Information**

Specific enthalpy of water (liquid and solid) is computed from temperature using constant properties as follows:

- heat capacity of liquid water: 4200 J/kg
- heat capacity of solid water: 2050 J/kg
- enthalpy of fusion (liquid=>solid): 333000 J/kg

Pressure is assumed to be around 1 bar. This function is usually used to determine the specific enthalpy of

the liquid or solid fraction of moist air.

## Inputs

Name	Default	Description
T		Temperature [K]

## Outputs

Name	Description
h	Specific enthalpy of water [J/kg]

## Modelica.Media.Air.MoistAir.enthalpyOfWater\_der

Derivative function of enthalpyOfWater

## Information

Derivative function for enthalpyOfWater.

## Inputs

Name	Default	Description
T		Temperature [K]
dT		Time derivative of temperature [K/s]

## Outputs

Name	Description
dh	Time derivative of specific enthalpy [J/(kg.s)]

## Modelica.Media.Air.MoistAir.pressure

Returns pressure of ideal gas as a function of the thermodynamic state record



## Information

Pressure is returned from the thermodynamic state record input as a simple assignment.

## Inputs

Name	Default	Description
state		

## Outputs

Name	Description
p	Pressure [Pa]

**Modelica.Media.Air.MoistAir.temperature****Return temperature of ideal gas as a function of the thermodynamic state record****Information**

Temperature is returned from the thermodynamic state record input as a simple assignment.

**Inputs**

Name	Default	Description
state		

**Outputs**

Name	Description
T	Temperature [K]

---

**Modelica.Media.Air.MoistAir.T\_phX****Return temperature as a function of pressure p, specific enthalpy h and composition X****Information**

Temperature is computed from pressure, specific enthalpy and composition via numerical inversion of function [h\\_pTX](#).

**Inputs**

Name	Default	Description
p		Pressure [Pa]
h		Specific enthalpy [J/kg]
X[:]		Mass fractions of composition [kg/kg]

**Outputs**

Name	Description
T	Temperature [K]

---

**Modelica.Media.Air.MoistAir.density****Returns density of ideal gas as a function of the thermodynamic state record****Information**

Density is computed from pressure, temperature and composition in the thermodynamic state record applying the ideal gas law.

**Inputs**

Name	Default	Description
state		

## Outputs

Name	Description
d	Density [kg/m3]

---

## Modelica.Media.Air.MoistAir.specificEnthalpy

Return specific enthalpy of moist air as a function of the thermodynamic state record



## Information

Specific enthalpy of moist air is computed from the thermodynamic state record. The fog region is included for both, ice and liquid fog.

## Inputs

Name	Default	Description
state		

## Outputs

Name	Description
h	Specific enthalpy [J/kg]

---

## Modelica.Media.Air.MoistAir.h\_pTX

Return specific enthalpy of moist air as a function of pressure p, temperature T and composition X



## Information

Specific enthalpy of moist air is computed from pressure, temperature and composition with X[1] as the total water mass fraction. The fog region is included for both, ice and liquid fog.

## Inputs

Name	Default	Description
p		Pressure [Pa]
T		Temperature [K]
X[:]		Mass fractions of moist air [1]

## Outputs

Name	Description
h	Specific enthalpy at p, T, X [J/kg]

---

## Modelica.Media.Air.MoistAir.h\_pTX\_der

Derivative function of h\_pTX



## 822 Modelica.Media.Air.MoistAir.h\_pTX\_der

---

### Information

Derivative function for [h\\_pTX](#).

### Inputs

Name	Default	Description
p		Pressure [Pa]
T		Temperature [K]
X[:]		Mass fractions of moist air [1]
dp		Pressure derivative [Pa/s]
dT		Temperature derivative [K/s]
dX[:]		Composition derivative [1/s]

### Outputs

Name	Description
h_der	Time derivative of specific enthalpy [J/(kg.s)]

---

## Modelica.Media.Air.MoistAir.[specificInternalEnergy](#)

Return specific internal energy of moist air as a function of the thermodynamic state record



### Information

Specific internal energy is determined from the thermodynamic state record, assuming that the liquid or solid water volume is negligible.

### Inputs

Name	Default	Description
state		

### Outputs

Name	Description
u	Specific internal energy [J/kg]

---

## Modelica.Media.Air.MoistAir.[specificInternalEnergy\\_pTX](#)

Return specific internal energy of moist air as a function of pressure p, temperature T and composition X

### Information

Specific internal energy is determined from pressure p, temperature T and composition X, assuming that the liquid or solid water volume is negligible.

### Inputs

Name	Default	Description

p	Pressure [Pa]
T	Temperature [K]
X[:]	Mass fractions of moist air [1]

## Outputs

Name	Description
u	Specific internal energy [J/kg]

## Modelica.Media.Air.MoistAir.specificInternalEnergy\_pTX\_der

Derivative function for specificInternalEnergy\_pTX

### Information

Derivative function for specificInternalEnergy\_pTX.

## Inputs

Name	Default	Description
p		Pressure [Pa]
T		Temperature [K]
X[:]		Mass fractions of moist air [1]
dp		Pressure derivative [Pa/s]
dT		Temperature derivative [K/s]
dX[:]		Mass fraction derivatives

## Outputs

Name	Description
u_der	Specific internal energy derivative [J/(kg.s)]

## Modelica.Media.Air.MoistAir.specificEntropy

Return specific entropy from thermodynamic state record, only valid for phi<1



### Information

Specific entropy is calculated from the thermodynamic state record, assuming ideal gas behavior and including entropy of mixing. Liquid or solid water is not taken into account, the entire water content X[1] is assumed to be in the vapor state (relative humidity below 1.0).

## Inputs

Name	Default	Description
state		

## Outputs

Name	Description
s	Specific entropy [J/(kg.K)]

**Modelica.Media.Air.MoistAir.specificGibbsEnergy**

Return specific Gibbs energy as a function of the thermodynamic state record, only valid for  $\phi < 1$

**Information**

The Gibbs Energy is computed from the thermodynamic state record for moist air with a water content below saturation.

**Inputs**

Name	Default	Description
state		

**Outputs**

Name	Description
g	Specific Gibbs energy [J/kg]

**Modelica.Media.Air.MoistAir.specificHelmholtzEnergy**

Return specific Helmholtz energy as a function of the thermodynamic state record, only valid for  $\phi < 1$

**Information**

The Specific Helmholtz Energy is computed from the thermodynamic state record for moist air with a water content below saturation.

**Inputs**

Name	Default	Description
state		

**Outputs**

Name	Description
f	Specific Helmholtz energy [J/kg]

**Modelica.Media.Air.MoistAir.specificHeatCapacityCp**

Return specific heat capacity at constant pressure as a function of the thermodynamic state record

**Information**

The specific heat capacity at constant pressure **cp** is computed from temperature and composition for a mixture of steam ( $X[1]$ ) and dry air. All water is assumed to be in the vapor state.

## Inputs

Name	Default	Description
state		

## Outputs

Name	Description
cp	Specific heat capacity at constant pressure [J/(kg.K)]

## Modelica.Media.Air.MoistAir.specificHeatCapacityCv

Return specific heat capacity at constant volume as a function of the thermodynamic state record



## Information

The specific heat capacity at constant density **cv** is computed from temperature and composition for a mixture of steam (X[1]) and dry air. All water is assumed to be in the vapor state.

## Inputs

Name	Default	Description
state		

## Outputs

Name	Description
cv	Specific heat capacity at constant volume [J/(kg.K)]

## Modelica.Media.Air.MoistAir.dynamicViscosity

Return dynamic viscosity as a function of the thermodynamic state record, valid from 73.15 K to 373.15 K



## Information

Dynamic viscosity is computed from temperature using a simple polynomial for dry air, assuming that moisture influence is small. Range of validity is from 73.15 K to 373.15 K.

## Inputs

Name	Default	Description
state		

## Outputs

Name	Description
eta	Dynamic viscosity [Pa.s]

## Modelica.Media.Air.MoistAir.thermalConductivity

Return thermal conductivity as a function of the thermodynamic state record, valid from 73.15 K to 373.15 K



### Information

Thermal conductivity is computed from temperature using a simple polynomial for dry air, assuming that moisture influence is small. Range of validity is from 73.15 K to 373.15 K.

### Inputs

Name	Default	Description
state		

### Outputs

Name	Description
lambda	Thermal conductivity [W/(m.K)]

## Modelica.Media.Air.MoistAir.Utilities

### utility functions

#### Package Content

Name	Description
(f) spliceFunction	Spline interpolation of two functions
(f) spliceFunction_der	Derivative of spliceFunction

## Modelica.Media.Air.MoistAir.Utilities.spliceFunction

Spline interpolation of two functions

### Inputs

Name	Default	Description
pos		Returned value for x-deltax >= 0
neg		Returned value for x+deltax <= 0
x		Function argument
deltax	1	Region around x with spline interpolation

### Outputs

Name	Description
out	

## Modelica.Media.Air.MoistAir.Utilities.spliceFunction\_der

Derivative of spliceFunction

## Inputs

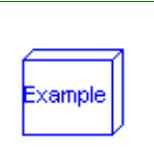
Name	Default	Description
pos		
neg		
x		
deltax	1	
dpos		
dneg		
dx		
ddeltax	0	

## Outputs

Name	Description
out	

## Modelica.Media.Air.MoistAir.PsychrometricData

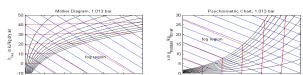
Produces plot data for psychrometric charts



### Information

This model produces psychrometric data from the moist air model in this library to be plotted in charts. The two most common chart varieties are the Mollier Diagram and the Psychrometric Chart. The first is widely used in some European countries while the second is more common in the Anglo-American world. Specific enthalpy is plotted over absolute humidity in the Mollier Diagram, it is the other way round in the Psychrometric Chart.

It must be noted that the relationship of both axis variables is not right-angled, the absolute humidity follows a slope which equals the enthalpy of vaporization at 0°C. For better reading and in order to reduce the fog region the humidity axis is rotated to obtain a right-angled plot. Both charts usually contain additional information as isochores or auxiliary scales for e.g. heat ratios. Those information are omitted in this model and the charts below. Other important features of psychrometric chart data are that all mass specific variables (like absolute humidity, specific enthalpy etc.) are expressed in terms of kg dry air and that their baseline of 0 enthalpy is found at 0°C and zero humidity.



**Legend:** blue - constant specific enthalpy, red - constant temperature, black - constant relative humidity

The model provides data for lines of constant specific enthalpy, temperature and relative humidity in a Mollier Diagram or Psychrometric Chart as they were used for the figures above. For limitations and ranges of validity please refer to the [MoistAir package description](#). Absolute humidity  $x$  is increased with time in this model. The specific enthalpies adjusted for plotting are then obtained from:

- $y_h$ : constant specific enthalpy
- $y_T$ : constant temperature
- $y_\phi$ : constant relative humidity

## Parameters

Name	Default	Description
p_const	1e5	Pressure [Pa]
n_T	11	Number of isotherms

T_min	253.15	Lowest isotherm [K]
T_step	10	Temperature step between two isotherms [K]
n_h	16	Number of lines with constant specific enthalpy
h_min	-20e3	Lowest line of constant enthalpy [J/kg]
h_step	1e4	Enthalpy step between two lines of constant enthalpy [J/kg]
n_phi	10	Number of lines with constant relative humidity
phi_min	0.1	Lowest line of constant humidity
phi_step	0.1	Step between two lines of constant humidity
x_min	0.00	Minimum diagram absolute humidity [1]
x_max	0.03	Maximum diagram absolute humidity [1]
t	1	Simulation time [s]

## Modelica.Media.CompressibleLiquids

### compressible liquid models

#### Information

##### Fluid models with linear compressibility, using PartialLinearFluid as base class.

The linear compressibility fluid models contained in this package are based on the assumptions that:

- The specific heat capacity at constant pressure ( $cp$ ) is constant
- The isobaric expansion coefficient ( $\beta$ ) is constant
- The isothermal compressibility ( $\kappa$ ) is constant
- Pressure and temperature are used as states

This results in models that are only valid for small temperature ranges, but sufficient to model compressibility and e.g. the "water hammer" effect. Another advantage is that only 3 values need to be measured to have an initial model. Hydraulic fluids can often be approximated by this type of model.

That means that the density is a linear function in temperature and in pressure. In order to define the complete model, a number of constant reference values are needed which are computed at the reference values of the states pressure  $p$  and temperature  $T$ . The model can be interpreted as a linearization of a full non-linear fluid model (but it is not linear in all thermodynamic coordinates). Reference values are needed for

1. the density (reference\_d),
2. the specific enthalpy (reference\_h),
3. the specific entropy (reference\_s).

Apart from that, a user needs to define the molar mass, MM\_const. Note that it is possible to define a fluid by computing the reference values from a full non-linear fluid model by computing the package constants using the standard functions defined in a fluid package (see example in Common, LinearWater\_pT).

#### Package Content

Name	Description
 Common	base classes for compressible liquids
 LinearColdWater	cold water model with linear compressibility
 LinearWater_pT_Ambient	liquid, linear compressibility water model at 1.01325 bar and 25 degree Celsius

**Modelica.Media.CompressibleLiquids.Common****base classes for compressible liquids****Package Content**

Name	Description
<a href="#">LinearWater_pT</a>	base class for liquid, linear compressibility water models

**Modelica.Media.CompressibleLiquids.Common.LinearWater\_pT****base class for liquid, linear compressibility water models****Package Content**

Name	Description
state=Modelica.Media.Water.StandardWater.setState_pT (reference_p, reference_T)	
<b>Inherited</b>	
cp_const	Specific heat capacity at constant pressure
beta_const	Thermal expansion coefficient at constant pressure
kappa_const	Isothermal compressibility
MM_const	Molar mass
reference_d	Density in reference conditions
reference_h	Specific enthalpy in reference conditions
reference_s	Specific entropy in reference conditions
constantJacobian	if true, entries in thermodynamic Jacobian are constant, taken at reference conditions
<a href="#">ThermodynamicState</a>	a selection of variables that uniquely defines the thermodynamic state
<a href="#">BaseProperties</a>	Base properties of medium
<a href="#">setState_pTX</a>	set the thermodynamic state record from p and T (X not needed)
<a href="#">setState_phX</a>	set the thermodynamic state record from p and h (X not needed)
<a href="#">setState_dTX</a>	set the thermodynamic state record from d and T (X not needed)
<a href="#">setState_psX</a>	set the thermodynamic state record from p and s (X not needed)
<a href="#">pressure</a>	Return the pressure from the thermodynamic state
<a href="#">temperature</a>	Return the temperature from the thermodynamic state
<a href="#">density</a>	Return the density from the thermodynamic state
<a href="#">specificEnthalpy</a>	Return the specific enthalpy from the thermodynamic state
<a href="#">specificEntropy</a>	Return the specific entropy from the thermodynamic state
<a href="#">specificInternalEnergy</a>	Return the specific internal energy from the

	thermodynamic state
(f) specificGibbsEnergy	Return specific Gibbs energy from the thermodynamic state
(f) specificHelmholtzEnergy	Return specific Helmholtz energy from the thermodynamic state
(f) velocityOfSound	Return velocity of sound from the thermodynamic state
(f) isentropicExponent	Return isentropic exponent from the thermodynamic state
(f) isentropicEnthalpy	Return isentropic enthalpy
(f) specificHeatCapacityCp	Return specific heat capacity at constant volume
(f) specificHeatCapacityCv	Return specific heat capacity at constant volume from the thermodynamic state
(f) isothermalCompressibility	Return the iso-thermal compressibility kappa
(f) isobaricExpansionCoefficient	Return the iso-baric expansion coefficient
(f) density_derP_h	Return density derivative wrt pressure at const specific enthalpy
(f) density_derH_p	Return density derivative wrt specific enthalpy at constant pressure
(f) density_derP_T	Return density derivative wrt pressure at const temperature
(f) density_derT_p	Return density derivative wrt temperature at constant pressure
(f) molarMass	Return molar mass
(f) T_ph	Return temperature from pressure and specific enthalpy
(f) T_ps	Return temperature from pressure and specific entropy
(f) setState_pT	Return thermodynamic state from p and T
(f) setState_ph	Return thermodynamic state from p and h
(f) setState_ps	Return thermodynamic state from p and s
(f) setState_dT	Return thermodynamic state from d and T
(f) density_ph	Return density from p and h
(f) temperature_ph	Return temperature from p and h
(f) pressure_dT	Return pressure from d and T
(f) specificEnthalpy_dT	Return specific enthalpy from d and T
(f) specificEnthalpy_ps	Return specific enthalpy from p and s
(f) temperature_ps	Return temperature from p and s
(f) density_ps	Return density from p and s
(f) specificEnthalpy_pT	Return specific enthalpy from p and T
(f) density_pT	Return density from p and T
mediumName="unusablePartialMedium"	Name of the medium
substanceNames={mediumName}	Names of the mixture substances. Set substanceNames={mediumName} if only one

	substance.
extraPropertiesNames=fill("", 0)	Names of the additional (extra) transported properties. Set extraPropertiesNames=fill("",0) if unused
singleState	= true, if u and d are not a function of pressure
reducedX=true	= true if medium contains the equation sum(X) = 1.0; set reducedX=true if only one substance (see docu for details)
fixedX=false	= true if medium contains the equation X = reference_X
reference_p=101325	Reference pressure of Medium: default 1 atmosphere
reference_T=298.15	Reference temperature of Medium: default 25 deg Celsius
reference_X=if nX == 0 then fill(0, nX) else fill(1/nX, nX)	Default mass fractions of medium
p_default=101325	Default value for pressure of medium (for initialization)
T_default=Modelica.SIunits.Conversions.from_degC(20)	Default value for temperature of medium (for initialization)
h_default=specificEnthalpy_pTX(p_default, T_default, X_default)	Default value for specific enthalpy of medium (for initialization)
X_default=reference_X	Default value for mass fractions of medium (for initialization)
nS=size(substanceNames, 1)	Number of substances
nX=if nS == 1 then 0 else nS	Number of mass fractions (= 0, if only one substance)
nXi=if fixedX then 0 else if reducedX then nS - 1 else nX	Number of structurally independent mass fractions (see docu for details)
nC=size(extraPropertiesNames, 1)	Number of extra (outside of standard mass-balance) transported properties
 FluidConstants	critical, triple, molecular and other standard data of fluid
 BasePropertiesRecord	Variables contained in every instance of BaseProperties
(f) dynamicViscosity	Return dynamic viscosity
(f) thermalConductivity	Return thermal conductivity
(f) prandtlNumber	Return the Prandtl number
(f) heatCapacity_cp	alias for deprecated name
(f) heatCapacity_cv	alias for deprecated name
(f) beta	alias for isobaricExpansionCoefficient for user convenience
(f) kappa	alias of isothermalCompressibility for user convenience
(f) density_derX	Return density derivative wrt mass fraction
(f) specificEnthalpy_pTX	Return specific enthalpy from p, T, and X or Xi
(f) density_pTX	Return density from p, T, and X or Xi
(f) temperature_phX	Return temperature from p, h, and X or Xi

---

**832 Modelica.Media.CompressibleLiquids.Common.LinearWater\_pT**


---

 <a href="#">density_phX</a>	Return density from p, h, and X or Xi
 <a href="#">temperature_psX</a>	Return temperature from p,s, and X or Xi
 <a href="#">density_psX</a>	Return density from p, s, and X or Xi
 <a href="#">specificEnthalpy_psX</a>	Return specific enthalpy from p, s, and X or Xi
<a href="#">AbsolutePressure</a>	Type for absolute pressure with medium specific attributes
<a href="#">Density</a>	Type for density with medium specific attributes
<a href="#">DynamicViscosity</a>	Type for dynamic viscosity with medium specific attributes
<a href="#">EnthalpyFlowRate</a>	Type for enthalpy flow rate with medium specific attributes
<a href="#">MassFlowRate</a>	Type for mass flow rate with medium specific attributes
<a href="#">MassFraction</a>	Type for mass fraction with medium specific attributes
<a href="#">MoleFraction</a>	Type for mole fraction with medium specific attributes
<a href="#">MolarMass</a>	Type for molar mass with medium specific attributes
<a href="#">MolarVolume</a>	Type for molar volume with medium specific attributes
<a href="#">IsentropicExponent</a>	Type for isentropic exponent with medium specific attributes
<a href="#">SpecificEnergy</a>	Type for specific energy with medium specific attributes
<a href="#">SpecificInternalEnergy</a>	Type for specific internal energy with medium specific attributes
<a href="#">SpecificEnthalpy</a>	Type for specific enthalpy with medium specific attributes
<a href="#">SpecificEntropy</a>	Type for specific entropy with medium specific attributes
<a href="#">SpecificHeatCapacity</a>	Type for specific heat capacity with medium specific attributes
<a href="#">SurfaceTension</a>	Type for surface tension with medium specific attributes
<a href="#">Temperature</a>	Type for temperature with medium specific attributes
<a href="#">ThermalConductivity</a>	Type for thermal conductivity with medium specific attributes
<a href="#">PrandtlNumber</a>	Type for Prandtl number with medium specific attributes
<a href="#">VelocityOfSound</a>	Type for velocity of sound with medium specific attributes
<a href="#">ExtraProperty</a>	Type for unspecified, mass-specific property transported by flow
<a href="#">CumulativeExtraProperty</a>	Type for conserved integral of unspecified, mass specific property
<a href="#">ExtraPropertyFlowRate</a>	Type for flow rate of unspecified, mass-specific property

IsobaricExpansionCoefficient	Type for isobaric expansion coefficient with medium specific attributes
DipoleMoment	Type for dipole moment with medium specific attributes
DerDensityByPressure	Type for partial derivative of density with respect to pressure with medium specific attributes
DerDensityByEnthalpy	Type for partial derivative of density with respect to enthalpy with medium specific attributes
DerEnthalpyByPressure	Type for partial derivative of enthalpy with respect to pressure with medium specific attributes
DerDensityByTemperature	Type for partial derivative of density with respect to temperature with medium specific attributes
Choices	Types, constants to define menu choices

## Types and constants

```
constant Modelica.Media.Water.StandardWater.ThermodynamicState state=
  Modelica.Media.Water.StandardWater.setState_pT(reference_p, reference_T);
```

---

## Modelica.Media.CompressibleLiquids.LinearColdWater

cold water model with linear compressibility

### Package Content

Name	Description
(f) dynamicViscosity	Dynamic viscosity of water
(f) thermalConductivity	Thermal conductivity of water
<b>Inherited</b>	
cp_const	Specific heat capacity at constant pressure
beta_const	Thermal expansion coefficient at constant pressure
kappa_const	Isothermal compressibility
MM_const	Molar mass
reference_d	Density in reference conditions
reference_h	Specific enthalpy in reference conditions
reference_s	Specific enthalpy in reference conditions
constantJacobian	if true, entries in thermodynamic Jacobian are constant, taken at reference conditions
ThermodynamicState	a selection of variables that uniquely defines the thermodynamic state
BaseProperties	Base properties of medium
(f) setState_pTX	set the thermodynamic state record from p and T (X not needed)
(f) setState_phX	set the thermodynamic state record from p and h (X not needed)
(f) setState_dTX	set the thermodynamic state record from d and T (X not needed)

(f) <code>setState_psX</code>	set the thermodynamic state record from p and s (X not needed)
(f) <code>pressure</code>	Return the pressure from the thermodynamic state
(f) <code>temperature</code>	Return the temperature from the thermodynamic state
(f) <code>density</code>	Return the density from the thermodynamic state
(f) <code>specificEnthalpy</code>	Return the specific enthalpy from the thermodynamic state
(f) <code>specificEntropy</code>	Return the specific entropy from the thermodynamic state
(f) <code>specificInternalEnergy</code>	Return the specific internal energy from the thermodynamic state
(f) <code>specificGibbsEnergy</code>	Return specific Gibbs energy from the thermodynamic state
(f) <code>specificHelmholtzEnergy</code>	Return specific Helmholtz energy from the thermodynamic state
(f) <code>velocityOfSound</code>	Return velocity of sound from the thermodynamic state
(f) <code>isentropicExponent</code>	Return isentropic exponent from the thermodynamic state
(f) <code>isentropicEnthalpy</code>	Return isentropic enthalpy
(f) <code>specificHeatCapacityCp</code>	Return specific heat capacity at constant volume
(f) <code>specificHeatCapacityCv</code>	Return specific heat capacity at constant volume from the thermodynamic state
(f) <code>isothermalCompressibility</code>	Return the iso-thermal compressibility kappa
(f) <code>isobaricExpansionCoefficient</code>	Return the iso-baric expansion coefficient
(f) <code>density_derP_h</code>	Return density derivative wrt pressure at const specific enthalpy
(f) <code>density_derH_p</code>	Return density derivative wrt specific enthalpy at constant pressure
(f) <code>density_derP_T</code>	Return density derivative wrt pressure at const temperature
(f) <code>density_derT_p</code>	Return density derivative wrt temperature at constant pressure
(f) <code>molarMass</code>	Return molar mass
(f) <code>T_ph</code>	Return temperature from pressure and specific enthalpy
(f) <code>T_ps</code>	Return temperature from pressure and specific entropy
(f) <code>setState_pT</code>	Return thermodynamic state from p and T
(f) <code>setState_ph</code>	Return thermodynamic state from p and h
(f) <code>setState_ps</code>	Return thermodynamic state from p and s
(f) <code>setState_dT</code>	Return thermodynamic state from d and T
(f) <code>density_ph</code>	Return density from p and h
(f) <code>temperature_ph</code>	Return temperature from p and h
(f) <code>pressure_dT</code>	Return pressure from d and T
(f) <code>specificEnthalpy_dT</code>	Return specific enthalpy from d and T
(f) <code>specificEnthalpy_ps</code>	Return specific enthalpy from p and s
(f) <code>temperature_ps</code>	Return temperature from p and s
(f) <code>density_ps</code>	Return density from p and s

 <a href="#">specificEnthalpy_pT</a>	Return specific enthalpy from p and T
 <a href="#">density_pT</a>	Return density from p and T
mediumName="unusablePartialMedium"	Name of the medium
substanceNames={mediumName}	Names of the mixture substances. Set substanceNames={mediumName} if only one substance.
extraPropertiesNames=fill("", 0)	Names of the additional (extra) transported properties. Set extraPropertiesNames=fill("",0) if unused
singleState	= true, if u and d are not a function of pressure
reducedX=true	= true if medium contains the equation sum(X) = 1.0; set reducedX=true if only one substance (see docu for details)
fixedX=false	= true if medium contains the equation X = reference_X
reference_p=101325	Reference pressure of Medium: default 1 atmosphere
reference_T=298.15	Reference temperature of Medium: default 25 deg Celsius
reference_X=if nX == 0 then fill(0, nX) else fill(1/nX, nX)	Default mass fractions of medium
p_default=101325	Default value for pressure of medium (for initialization)
T_default=Modelica.SIunits.Conversions.from_degC(20)	Default value for temperature of medium (for initialization)
h_default=specificEnthalpy_pTX(p_default, T_default, X_default)	Default value for specific enthalpy of medium (for initialization)
X_default=reference_X	Default value for mass fractions of medium (for initialization)
nS=size(substanceNames, 1)	Number of substances
nX=if nS == 1 then 0 else nS	Number of mass fractions (= 0, if only one substance)
nXi=if fixedX then 0 else if reducedX then nS - 1 else nX	Number of structurally independent mass fractions (see docu for details)
nC=size(extraPropertiesNames, 1)	Number of extra (outside of standard mass-balance) transported properties
 <a href="#">FluidConstants</a>	critical, triple, molecular and other standard data of fluid
 <a href="#">BasePropertiesRecord</a>	Variables contained in every instance of BaseProperties
 <a href="#">prandtlNumber</a>	Return the Prandtl number
 <a href="#">heatCapacity_cp</a>	alias for deprecated name
 <a href="#">heatCapacity_cv</a>	alias for deprecated name
 <a href="#">beta</a>	alias for isobaricExpansionCoefficient for user convenience
 <a href="#">kappa</a>	alias of isothermalCompressibility for user convenience
 <a href="#">density_derX</a>	Return density derivative wrt mass fraction
 <a href="#">specificEnthalpy_pTX</a>	Return specific enthalpy from p, T, and X or Xi
 <a href="#">density_pTX</a>	Return density from p, T, and X or Xi
 <a href="#">temperature_phX</a>	Return temperature from p, h, and X or Xi
 <a href="#">density_phX</a>	Return density from p, h, and X or Xi
 <a href="#">temperature_psX</a>	Return temperature from p,s, and X or Xi
 <a href="#">density_psX</a>	Return density from p, s, and X or Xi
 <a href="#">specificEnthalpy_psX</a>	Return specific enthalpy from p, s, and X or Xi
AbsolutePressure	Type for absolute pressure with medium specific attributes

Density	Type for density with medium specific attributes
DynamicViscosity	Type for dynamic viscosity with medium specific attributes
EnthalpyFlowRate	Type for enthalpy flow rate with medium specific attributes
MassFlowRate	Type for mass flow rate with medium specific attributes
MassFraction	Type for mass fraction with medium specific attributes
MoleFraction	Type for mole fraction with medium specific attributes
MolarMass	Type for molar mass with medium specific attributes
MolarVolume	Type for molar volume with medium specific attributes
IsentropicExponent	Type for isentropic exponent with medium specific attributes
SpecificEnergy	Type for specific energy with medium specific attributes
SpecificInternalEnergy	Type for specific internal energy with medium specific attributes
SpecificEnthalpy	Type for specific enthalpy with medium specific attributes
SpecificEntropy	Type for specific entropy with medium specific attributes
SpecificHeatCapacity	Type for specific heat capacity with medium specific attributes
SurfaceTension	Type for surface tension with medium specific attributes
Temperature	Type for temperature with medium specific attributes
ThermalConductivity	Type for thermal conductivity with medium specific attributes
PrandtlNumber	Type for Prandtl number with medium specific attributes
VelocityOfSound	Type for velocity of sound with medium specific attributes
ExtraProperty	Type for unspecified, mass-specific property transported by flow
CumulativeExtraProperty	Type for conserved integral of unspecified, mass specific property
ExtraPropertyFlowRate	Type for flow rate of unspecified, mass-specific property
IsobaricExpansionCoefficient	Type for isobaric expansion coefficient with medium specific attributes
DipoleMoment	Type for dipole moment with medium specific attributes
DerDensityByPressure	Type for partial derivative of density with respect to pressure with medium specific attributes
DerDensityByEnthalpy	Type for partial derivative of density with respect to enthalpy with medium specific attributes
DerEnthalpyByPressure	Type for partial derivative of enthalpy with respect to pressure with medium specific attributes
DerDensityByTemperature	Type for partial derivative of density with respect to temperature with medium specific attributes
 Choices	Types, constants to define menu choices

**Modelica.Media.CompressibleLiquids.LinearColdWater.dynamicViscosity****Dynamic viscosity of water****Inputs**

Name	Default	Description
state		

## Outputs

Name	Description
eta	Dynamic viscosity [Pa.s]

---

## Modelica.Media.CompressibleLiquids.LinearColdWater.thermalConductivity

Thermal conductivity of water



## Inputs

Name	Default	Description
state		

## Outputs

Name	Description
lambda	Thermal conductivity [W/(m.K)]

---

## Modelica.Media.CompressibleLiquids.LinearWater\_pT\_Ambient

liquid, linear compressibility water model at 1.01325 bar and 25 degree Celsius

## Information

Water model with linear compressibility at ambient conditions

---

## Modelica.Media.IdealGases

Data and models of ideal gases (single, fixed and dynamic mixtures) from NASA source

## Information

This package contains medium models for the 1241 ideal gases from

McBride B.J., Zehe M.J., and Gordon S. (2002): **NASA Glenn Coefficients for Calculating Thermodynamic Properties of Individual Species**. NASA report TP-2002-211556

Medium models for the following gases are available in package **IdealGases.SingleGases**:

Ag	BaOH+	C2H4O_ethylene_o	DF	In2I4	Nb	ScO2
Ag+	Ba_OH_2	CH3CHO_ethanal	DOCl	In2I6	Nb+	Sc2O
Ag-	BaS	CH3COOH	DO2	In2O	Nb-	Sc2O2
Air	Ba2	OHCH2COOH	DO2-	K	NbCl5	Si
Al	Be	C2H5	D2	K+	NbO	Si+
Al+	Be+	C2H5Br	D2+	K-	NbOC13	Si-
Al-	Be++	C2H6	D2-	KAlF4	NbO2	SiBr
AlBr	BeBr	CH3N2CH3	D2O	KBO2	Ne	SiBr2
AlBr2	BeBr2	C2H5OH	D2O2	KBr	Ne+	SiBr3
AlBr3	BeCl	CH3OCH3	D2S	KCN	Ni	SiBr4
AlC	BeCl2	CH3O2CH3	e-	KCl	Ni+	SiC
AlC2	BeF	CCN	F	KF	Ni-	SiC2
AlCl	BeF2	CNC	F+	KH	NiCl	SiCl

AlCl+	BeH	OCCN	F-	KI	NiCl2	SiCl2
AlC12	BeH+	C2N2	FCN	Kli	NiO	SiCl3
AlC13	BeH2	C2O	FCO	KNO2	NiS	SiCl4
AlF	BeI	C3	FO	KNO3	O	SiF
AlF+	BeI2	C3H3_1_propynl	FO2_FOO	KNa	O+	SiFC1
AlFC1	BeN	C3H3_2_propynl	FO2_OFO	KO	O-	SiF2
AlFC12	BeO	C3H4_allene	F2	KOH	OD	SiF3
AlF2	BeOH	C3H4_propyne	F2O	K2	OD-	SiF4
AlF2-	BeOH+	C3H4_cyclo	F2O2	K2+	OH	SiH
AlF2C1	Be_OH_2	C3H5_allyl	FS2F	K2Br2	OH+	SiH+
AlF3	BeS	C3H6_propylene	Fe	K2CO3	OH-	SiHBr3
AlF4-	Be2	C3H6_cyclo	Fe+	K2C2N2	O2	SiHCl
AlH	Be2Cl4	C3H6O_propylox	Fe_CO_5	K2C12	O2+	SiHCl3
AlHCl	Be2F4	C3H6O_acetone	FeCl	K2F2	O2-	SiHF
AlHC12	Be2O	C3H6O_propanal	FeC12	K2I2	O3	SiHF3
AlHF	Be2OF2	C3H7_n_propyl	FeC13	K2O	P	SiHI3
AlHFCl	Be2O2	C3H7_i_propyl	FeO	K2O2	P+	SiH2
AlHF2	Be3O3	C3H8_Fe_OH_2	K2O2	P-		SiH2Br2
AlH2	Be4O4	C3H8O_1propanol	Fe2C14	K2O2H2	PC1	SiH2Cl2
AlH2C1	Br	C3H8O_2propanol	Fe2C16	K2SO4	PC12	SiH2F2
AlH2F	Br+	CNCOCN	Ga	Kr	PC12-	SiH2I2
AlH3	Br-	C3O2	Ga+	Kr+	PC13	SiH3
AlI	BrCl	C4	GaBr	li	PC15	SiH3Br
AlI2	BrF	C4H2_butadiyne	GaBr2	li+	PF	SiH3Cl
AlI3	BrF3	C4H4_1_3-cyclo	GaBr3	li-	PF+	SiH3F
AlN	BrF5	C4H6_butadiene	GaCl	liaAlF4	PF-	SiH3I
AlO	Bro	C4H6_1butyne	GaCl2	libO2	PFC1	SiH4
AlO+	OBro	C4H6_2butyne	GaCl3	libr	PFC1-	SiI
AlO-	BrOO	C4H6_cyclo	GaF	licl	PFC12	SiI2
AlOC1	BrO3	C4H8_1_butene	GaF2	lif	PFC14	SiN
AlOC12	Br2	C4H8_cis2_butene	GaF3	liH	PF2	SiO
AlOF	BrBrO	C4H8_isobutene	GaH	liI	PF2-	SiO2
AlOF2	BroBr	C4H8_cyclo	GaI	lin	PF2C1	SiS
AlOF2-	C	C4H9_n_butyl	GaI2	linO2	PF2C13	SiS2
AlOH	C+	C4H9_i_butyl	GaI3	linO3	PF3	Si2
AlOHC1	C-	C4H9_s_butyl	GaO	lio	PF3C12	Si2C
AlOHC12	CBr	C4H9_t_butyl	GaOH	lioF	PF4C1	Si2F6
AlOHF	CBr2	C4H10_n_butane	Ga2Br2	lioH	PF5	Si2N
AlOHF2	CBr3	C4H10_isobutane	Ga2Br4	lion	PH	Si3
AlO2	CBr4	C4N2	Ga2Br6	li2	PH2	Sn
AlO2-	CC1	C5	Ga2C12	li2+	PH2-	Sn+
Al_OH_2	CC12	C5H6_1_3cyclo	Ga2C14	li2Br2	PH3	Sn-
Al_OH_2C1	CC12Br2	C5H8_cyclo	Ga2C16	li2F2	PN	SnBr
Al_OH_2F	CC13	C5H10_1_pentene	Ga2F2	li2I2	PO	SnBr2
Al_OH_3	CC13Br	C5H10_cyclo	Ga2F4	li2O	PO-	SnBr3
AlS	CC14	C5H11_pentyl	Ga2F6	li2O+	POC13	SnBr4
AlS2	CF	C5H11_t_pentyl	Ga2I2	li2O2	POFC12	SnCl
A12	CF+	C5H12_n_pentane	Ga2I4	li2O2H2	POF2C1	SnC12
A12Br6	CFBr3	C5H12_i_pentane	Ga2I6	li2SO4	POF3	SnC13
A12C2	CFC1	CH3C_CH3_2CH3	Ga2O	li3+	PO2	SnC14
A12C16	CFC1Br2	C6D5_phenyl	Ge	li3Br3	PO2-	SnF
A12F6	CFC12	C6D6	Ge+	li3C13	PS	SnF2
A12I6	CFC12Br	C6H2	Ge-	li3F3	P2	SnF3
A12O	CFC13	C6H5_phenyl	GeBr	li3I3	P2O3	SnF4
A12O+	CF2	C6H5O_phenoxy	GeBr2	Mg	P2O4	SnI
A12O2	CF2+	C6H6	GeBr3	Mg+	P2O5	SnI2
A12O2+	CF2Br2	C6H5OH_phenol	GeBr4	MgBr	P3	SnI3
A12O3	CF2Cl1	C6H10_cyclo	GeCl	MgBr2	P3O6	SnI4
A12S	CF2Cl1Br	C6H12_1_hexene	GeCl2	MgCl	P4	SnO

A12S2	CF2C12	C6H12_cyclo	GeCl3	MgCl+	P4O6	SnO2
Ar	CF3	C6H13_n_hexyl	GeCl4	MgCl2	P4O7	SnS
Ar+	CF3+	C6H14_n_hexane	GeF	MgF	P4O8	SnS2
B	CF3Br	C7H7_benzyl	GeF2	MgF+	P4O9	Sn2
B+	CF3Cl	C7H8	GeF3	MgF2	P4O10	Sr
B-	CF4	C7H8O_cresol_mx	GeF4	MgF2+	Pb	Sr+
BBR	CH+	C7H14_1_heptene	GeH4	MgH	Pb+	SrBr
BBR2	CHBr3	C7H15_n_heptyl	GeI	MgI	Pb-	SrBr2
BBR3	CHCl	C7H16_n_heptane	GeO	MgI2	PbBr	SrCl
BC	CHClBr2	C7H16_2_methylh	GeO2	MgN	PbBr2	SrCl+
BC2	CHCl2	C8H8_styrene	GeS	MgO	PbBr3	SrCl2
BC1	CHCl2Br	C8H10_ethylbenz	GeS2	MgOH	PbBr4	SrF
BC1+	CHCl3	C8H16_1_octene	Ge2	MgOH+	PbCl	SrF+
BC1OH	CHF	C8H17_n_octyl	H	Mg_OH_2	PbCl2	SrF2
BC1_OH_2	CHFBr2	C8H18_n_octane	H+	MgS	PbCl3	SrH
BC12	CHFC1	C8H18_isooctane	H-	Mg2	PbCl4	SrI
BC12+	CHFC1Br	C9H19_n_nonyl	HALO	Mg2F4	PbF	SrI2
BC12OH	CHFC12	C10H8_naphthale	HALO2	Mn	PbF2	SrO
BF	CHF2	C10H21_n_decyl	HBO	Mn+	PbF3	SrOH
BFC1	CHF2Br	C12H9_o_bipheny	HBO+	Mo	PbF4	SrOH+
BFC12	CHF2Cl	C12H10_biphenyl	HBO2	Mo+	PbI	Sr_OH_2
BFOH	CHF3	Ca	HBS	Mo-	PbI2	SrS
BF_OH_2	CHI3	Ca+	HBS+	MoO	PbI3	Sr2
BF2	CH2	CaBr	HCN	MoO2	PbI4	Ta
BF2+	CH2Br2	CaBr2	HCO	MoO3	PbO	Ta+
BF2-	CH2Cl1	CaCl	HCO+	MoO3-	PbO2	Ta-
BF2Cl1	CH2Cl1Br	CaCl+	HCCN	Mo2O6	PbS	TaCl5
BF2OH	CH2C12	CaCl2	HCCO	Mo3O9	PbS2	TaO
BF3	CH2F	CaF	HC1	Mo4O12	Rb	TaO2
BF4-	CH2FBr	CaF+	HD	Mo5O15	Rb+	Ti
BH	CH2FC1	CaF2	HD+	N	Rb-	Ti+
BHC1	CH2F2	CaH	HDO	N+	RbBO2	Ti-
BHC12	CH2I2	CaI	HDO2	N-	RbBr	TiCl
BHF	CH3	CaI2	HF	NCO	RbCl	TiCl2
BHFC1	CH3Br	CaO	HI	ND	RbF	TiCl3
BHF2	CH3Cl1	CaO+	HNC	ND2	RbH	TiCl4
BH2	CH3F	CaOH	HNCO	ND3	RbI	TiO
BH2C1	CH3I	CaOH+	HNO	NF	RbK	TiO+
BH2F	CH2OH	Ca_OH_2	HNO2	NF2	Rbli	TiOCl
BH3	CH2OH+	CaS	HNO3	NF3	RbNO2	TiOCl2
BH3NH3	CH3O	Ca2	HOCl	NH	RbNO3	TiO2
BH4	CH4	Cd	HOF	NH+	RbNa	U
BI	CH3OH	Cd+	HO2	NHF	RbO	UF
BI2	CH3OOH	C1	HO2-	NHF2	RbOH	UF+
BI3	CI	Cl+	HPO	NH2	Rb2Br2	UF-
BN	CI2	Cl-	HSO3F	NH2F	Rb2C12	UF2
BO	CI3	C1CN	H2	NH3	Rb2F2	UF2+
BO-	CI4	C1F	H2+	NH2OH	Rb2I2	UF2-
BOC1	CN	C1F3	H2-	NH4+	Rb2O	UF3
BOC12	CN+	C1F5	HBOH	NO	Rb2O2	UF3+
BOF	CN-	C1O	HCOOH	NOCl	Rb2O2H2	UF3-
BOF2	CNN	C1O2	H2F2	NOF	Rb2SO4	UF4
BOH	CO	C12	H2O	NOF3	Rn	UF4+
BO2	CO+	C12O	H2O+	NO2	Rn+	UF4-
BO2-	COCl	Co	H2O2	NO2-	S	UF5
B_OH_2	COCl2	Co+	H2S	NO2Cl	S+	UF5+
BS	COFCl	Co-	H2SO4	NO2F	S-	UF5-
BS2	COF2	Cr	H2BOH	NO3	SC1	UF6
B2	COHCl	Cr+	HB_OH_2	NO3-	SC12	UF6-

B2C	COHF	Cr-	H3BO3	NO3F	SC12+	UO
B2C14	COS	CrN	H3B3O3	N2	SD	UO+
B2F4	CO2	CrO	H3B3O6	N2+	SF	UOF
B2H	CO2+	CrO2	H3F3	N2-	SF+	UOF2
B2H2	COOH	CrO3	H3O+	NCN	SF-	UOF3
B2H3	CP	CrO3-	H4F4	N2D2_cis	SF2	UOF4
B2H3_db	CS	Cs	H5F5	N2F2	SF2+	UO2
B2H4	CS2	Cs+	H6F6	N2F4	SF2-	UO2+
B2H4_db	C2	Cs-	H7F7	N2H2	SF3	UO2-
B2H5	C2+	CsBO2	He	NH2NO2	SF3+	UO2F
B2H5_db	C2-	CsBr	He+	N2H4	SF3-	UO2F2
B2H6	C2Cl	CsCl	Hg	N2O	SF4	UO3
B2O	C2Cl2	CsF	Hg+	N2O+	SF4+	UO3-
B2O2	C2Cl3	CsH	HgBr2	N2O3	SF4-	V
B2O3	C2Cl4	CsI	I	N2O4	SF5	V+
B2_OH_4	C2Cl6	Csli	I+	N2O5	SF5+	V-
B2S	C2F	CsNO2	I-	N3	SF5-	VC14
B2S2	C2FC1	CsNO3	IF5	N3H	SF6	VN
B2S3	C2FC13	CsNa	IF7	Na	SF6-	VO
B3H7_C2v	C2F2	CsO	I2	Na+	SH	VO2
B3H7_Cs	C2F2C12	CsOH	In	Na-	SH-	V4O10
B3H9	C2F3	CsRb	In+	NaAlF4	SN	W
B3N3H6	C2F3C1	Cs2	InBr	NaBO2	SO	W+
B3O3C13	C2F4	Cs2Br2	InBr2	NaBr	SO-	W-
B3O3FC12	C2F6	Cs2CO3	InBr3	NaCN	SOF2	WC16
B3O3F2C1	C2H	Cs2C12	InCl	NaCl	SO2	WO
B3O3F3	C2HC1	Cs2F2	InCl2	NaF	SO2-	WOC14
B4H4	C2HC13	Cs2I2	InCl3	NaH	SO2C12	WO2
B4H10	C2HF	Cs2O	InF	NaI	SO2FC1	WO2C12
B4H12	C2HFC12	Cs2O+	InF2	Nali	SO2F2	WO3
B5H9	C2HF2C1	Cs2O2	InF3	NaNO2	SO3	WO3-
Ba	C2HF3	Cs2O2H2	InH	NaNO3	S2	Xe
Ba+	C2H2_vinylidene	Cs2SO4	InI	NaO	S2-	Xe+
BaBr	C2H2C12	Cu	InI2	NaOH	S2C12	Zn
BaBr2	C2H2FC1	Cu+	InI3	NaOH+	S2F2	Zn+
BaCl	C2H2F2	Cu-	InO	Na2	S2O	Zr
BaCl+	CH2CO_ketene	CuCl	InOH	Na2Br2	S3	Zr+
BaCl2	O_CH_2O	CuF	In2Br2	Na2C12	S4	Zr-
BaF	HO_CO_2OH	CuF2	In2Br4	Na2F2	S5	ZrN
BaF+	C2H3_vinyl	CuO	In2Br6	Na2I2	S6	ZrO
BaF2	CH2Br-COOH	Cu2	In2C12	Na2O	S7	ZrO+
BaH	C2H3C1	Cu3C13	In2C14	Na2O+	S8	ZrO2
BaI	CH2Cl-COOH	D	In2C16	Na2O2	Sc	
BaI2	C2H3F	D+	In2F2	Na2O2H2	Sc+	
BaO	CH3CN	D-	In2F4	Na2SO4	Sc-	
BaO+	CH3CO_acetyl	DBr	In2F6	Na3C13	ScO	
BaOH	C2H4	DC1	In2I2	Na3F3	ScO+	

## Package Content

Name	Description
 Common	Common packages and data for the ideal gas models
 MixtureGases	Medium models consisting of mixtures of ideal gases
 SingleGases	Media models of ideal gases from NASA tables

## Modelica.Media.IdealGases.Common

Common packages and data for the ideal gas models

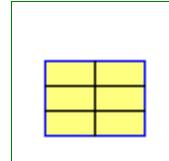
### Information

#### Package Content

Name	Description
DataRecord	Coefficient data record for properties of ideal gases based on NASA source
SingleGasNasa	Medium model of an ideal gas based on NASA source
MixtureGasNasa	Medium model of a mixture of ideal gases based on NASA source
FluidData	critical data, dipole moments and related data
SingleGasesData	Ideal gas data based on the NASA Glenn coefficients

## Modelica.Media.IdealGases.Common.DataRecord

Coefficient data record for properties of ideal gases based on NASA source



### Information

This data record contains the coefficients for the ideal gas equations according to:

McBride B.J., Zehe M.J., and Gordon S. (2002): **NASA Glenn Coefficients for Calculating Thermodynamic Properties of Individual Species**. NASA report TP-2002-211556

The equations have the following structure:

$$\begin{aligned}
 cp(T) &= R \sum_{i=1}^7 a_i T^{i-3} \\
 h(T) &= RT \left( -\frac{a_1}{T^2} + a_2 \frac{\log(T)}{T} + \sum_{i=3}^7 a_i \frac{T^{i-3}}{i-2} + \frac{b_1}{T} \right) \\
 s_0(T) &= R \left( -\frac{a_1}{2T^2} - \frac{a_2}{T} + a_3 \log(T) + \sum_{i=4}^7 a_i \frac{T^{i-3}}{i-3} + b_2 \right) \\
 s(T, p) &= s_0(T) - R \ln \left( \frac{p}{p_0} \right)
 \end{aligned}$$

The polynomials for  $h(T)$  and  $s_0(T)$  are derived via integration from the one for  $cp(T)$  and contain the integration constants  $b_1, b_2$  that define the reference specific enthalpy and entropy. For entropy differences the reference pressure  $p_0$  is arbitrary, but not for absolute entropies. It is chosen as 1 standard atmosphere (101325 Pa).

For most gases, the region of validity is from 200 K to 6000 K. The equations are splitted into two regions that are separated by  $T_{limit}$  (usually 1000 K). In both regions the gas is described by the data above. The two branches are continuous and in most gases also differentiable at  $T_{limit}$ .

**Modelica.Media.IdealGases.Common.SingleGasNasa****Medium model of an ideal gas based on NASA source****Information**

This model calculates medium properties for an ideal gas of a single substance, or for an ideal gas consisting of several substances where the mass fractions are fixed. Independent variables are temperature  $T$  and pressure  $p$ . Only density is a function of  $T$  and  $p$ . All other quantities are solely a function of  $T$ . The properties are valid in the range:

$$200 \text{ K} \leq T \leq 6000 \text{ K}$$

The following quantities are always computed:

Variable	Unit	Description
$h$	J/kg	specific enthalpy $h = h(T)$
$u$	J/kg	specific internal energy $u = u(T)$
$d$	kg/m <sup>3</sup>	density $d = d(p, T)$

For the other variables, see the functions in Modelica.Media.IdealGases.Common.SingleGasNasa. Note, dynamic viscosity and thermal conductivity are only provided for gases that use a data record from Modelica.Media.IdealGases.FluidData. Currently these are the following gases:

Ar  
C2H2\_vinylidene  
C2H4  
C2H5OH  
C2H6  
C3H6\_propylene  
C3H7OH  
C3H8  
C4H8\_1\_butene  
C4H9OH  
C4H10\_n\_butane  
C5H10\_1\_pentene  
C5H12\_n\_pentane  
C6H6  
C6H12\_1\_hexene  
C6H14\_n\_heptane  
C7H14\_1\_heptene  
C8H10\_ethylbenz  
CH3OH  
CH4  
CL2  
CO  
CO2  
F2  
H2  
H2O  
He  
N2  
N2O  
NH3  
NO  
O2  
SO2  
SO3

**Sources for model and literature:**

Original Data: Computer program for calculation of complex chemical equilibrium compositions and applications. Part 1: Analysis Document ID: 19950013764 N (95N20180) File Series: NASA Technical Reports Report Number: NASA-RP-1311 E-8017 NAS 1.61:1311 Authors: Gordon, Sanford (NASA Lewis Research Center) McBride, Bonnie J. (NASA Lewis Research Center) Published: Oct 01, 1994.

**Known limits of validity:**

The data is valid for temperatures between 200K and 6000K. A few of the data sets for monatomic gases have a discontinuous 1st derivative at 1000K, but this never caused problems so far.

This model has been copied from the ThermoFluid library and adapted to the Modelica.Media package.

**Package Content**

Name	Description
ThermodynamicState	thermodynamic state variables for ideal gases
FluidConstants	extended fluid constants
excludeEnthalpyOfFormation=true	If true, enthalpy of formation Hf is not included in specific enthalpy h
referenceChoice=Choices.ReferenceEnthalpy.ZeroAt0K	Choice of reference enthalpy
h_offset=0.0	User defined offset for reference enthalpy, if referenceChoice = UserDefined
data	Data record of ideal gas substance
fluidConstants	constant data for the fluid
BaseProperties	Base properties of ideal gas medium
setState_pTX	Return thermodynamic state as function of p, T and composition X
setState_phX	Return thermodynamic state as function of p, h and composition X
setState_psX	Return thermodynamic state as function of p, s and composition X
setState_dTX	Return thermodynamic state as function of d, T and composition X
pressure	return pressure of ideal gas
temperature	return temperature of ideal gas
density	return density of ideal gas
specificEnthalpy	Return specific enthalpy
specificInternalEnergy	Return specific internal energy
specificEntropy	Return specific entropy
specificGibbsEnergy	Return specific Gibbs energy
specificHelmholtzEnergy	Return specific Helmholtz energy
specificHeatCapacityCp	Return specific heat capacity at constant pressure
specificHeatCapacityCv	Compute specific heat capacity at constant volume from temperature and gas data
isentropicExponent	Return isentropic exponent
velocityOfSound	Return velocity of sound

(f) <code>isentropicEnthalpyApproximation</code>	approximate method of calculating $h_{is}$ from upstream properties and downstream pressure
(f) <code>isentropicEnthalpy</code>	Return isentropic enthalpy
(f) <code>isobaricExpansionCoefficient</code>	Returns overall the isobaric expansion coefficient beta
(f) <code>isothermalCompressibility</code>	Returns overall the isothermal compressibility factor
(f) <code>density_derP_T</code>	density derivative by temperature at constant pressure
(f) <code>density_derT_p</code>	density derivative by temperature at constant pressure
(f) <code>density_derX</code>	density derivative by mass fraction
(f) <code>cp_T</code>	Compute specific heat capacity at constant pressure from temperature and gas data
(f) <code>cp_Tlow</code>	Compute specific heat capacity at constant pressure, low T region
(f) <code>cp_Tlow_der</code>	Compute specific heat capacity at constant pressure, low T region
(f) <code>h_T</code>	Compute specific enthalpy from temperature and gas data; reference is decided by the refChoice input, or by the referenceChoice package constant by default
(f) <code>h_T_der</code>	derivative function for $h_T$
(f) <code>h_Tlow</code>	Compute specific enthalpy, low T region; reference is decided by the refChoice input, or by the referenceChoice package constant by default
(f) <code>h_Tlow_der</code>	Compute specific enthalpy, low T region; reference is decided by the refChoice input, or by the referenceChoice package constant by default
(f) <code>s0_T</code>	Compute specific entropy from temperature and gas data
(f) <code>s0_Tlow</code>	Compute specific entropy, low T region
(f) <code>dynamicViscosityLowPressure</code>	Dynamic viscosity of low pressure gases
(f) <code>dynamicViscosity</code>	dynamic viscosity
(f) <code>thermalConductivityEstimate</code>	Thermal conductivity of polyatomic gases(Eucken and Modified Eucken correlation)
(f) <code>thermalConductivity</code>	thermal conductivity of gas
(f) <code>molarMass</code>	return the molar mass of the medium
(f) <code>T_h</code>	Compute temperature from specific enthalpy
(f) <code>T_ps</code>	Compute temperature from pressure and specific entropy
<b>Inherited</b>	
(f) <code>setState_pT</code>	Return thermodynamic state from p and T
(f) <code>setState_ph</code>	Return thermodynamic state from p and h
(f) <code>setState_ps</code>	Return thermodynamic state from p and s
(f) <code>setState_dT</code>	Return thermodynamic state from d and T
(f) <code>density_ph</code>	Return density from p and h
(f) <code>temperature_ph</code>	Return temperature from p and h
(f) <code>pressure_dT</code>	Return pressure from d and T

(f) <code>specificEnthalpy_dT</code>	Return specific enthalpy from d and T
(f) <code>specificEnthalpy_ps</code>	Return specific enthalpy from p and s
(f) <code>temperature_ps</code>	Return temperature from p and s
(f) <code>density_ps</code>	Return density from p and s
(f) <code>specificEnthalpy_pT</code>	Return specific enthalpy from p and T
(f) <code>density_pT</code>	Return density from p and T
<code>mediumName="unusablePartialMedium"</code>	Name of the medium
<code>substanceNames={mediumName}</code>	Names of the mixture substances. Set <code>substanceNames={mediumName}</code> if only one substance.
<code>extraPropertiesNames=fill("", 0)</code>	Names of the additional (extra) transported properties. Set <code>extraPropertiesNames=fill("",0)</code> if unused
<code>singleState</code>	= true, if u and d are not a function of pressure
<code>reducedX=true</code>	= true if medium contains the equation $\sum(X) = 1.0$ ; set <code>reducedX=true</code> if only one substance (see docu for details)
<code>fixedX=false</code>	= true if medium contains the equation $X = \text{reference\_}X$
<code>reference_p=101325</code>	Reference pressure of Medium: default 1 atmosphere
<code>reference_T=298.15</code>	Reference temperature of Medium: default 25 deg Celsius
<code>reference_X=if nX == 0 then fill(0, nX) else fill(1/nX, nX)</code>	Default mass fractions of medium
<code>p_default=101325</code>	Default value for pressure of medium (for initialization)
<code>T_default=Modelica.SIunits.Conversions.from_degC(20)</code>	Default value for temperature of medium (for initialization)
<code>h_default=specificEnthalpy_pTX(p_default, T_default, X_default)</code>	Default value for specific enthalpy of medium (for initialization)
<code>X_default=reference_X</code>	Default value for mass fractions of medium (for initialization)
<code>nS=size(substanceNames, 1)</code>	Number of substances
<code>nX=if nS == 1 then 0 else nS</code>	Number of mass fractions (= 0, if only one substance)
<code>nXi=if fixedX then 0 else if reducedX then nS - 1 else nX</code>	Number of structurally independent mass fractions (see docu for details)
<code>nC=size(extraPropertiesNames, 1)</code>	Number of extra (outside of standard mass-balance) transported properties
 <code>BasePropertiesRecord</code>	Variables contained in every instance of BaseProperties
(f) <code>prandtlNumber</code>	Return the Prandtl number
(f) <code>heatCapacity_cp</code>	alias for deprecated name
(f) <code>heatCapacity_cv</code>	alias for deprecated name
(f) <code>beta</code>	alias for isobaricExpansionCoefficient for user convenience
(f) <code>kappa</code>	alias of isothermalCompressibility for user convenience
(f) <code>density_derP_h</code>	Return density derivative wrt pressure at const specific enthalpy
(f) <code>density_derH_p</code>	Return density derivative wrt specific enthalpy at constant pressure
(f) <code>specificEnthalpy_pTX</code>	Return specific enthalpy from p, T, and X or Xi
(f) <code>density_pTX</code>	Return density from p, T, and X or Xi
(f) <code>temperature_phX</code>	Return temperature from p, h, and X or Xi

 <a href="#">density_phX</a>	Return density from p, h, and X or Xi
 <a href="#">temperature_psX</a>	Return temperature from p,s, and X or Xi
 <a href="#">density_psX</a>	Return density from p, s, and X or Xi
 <a href="#">specificEnthalpy_psX</a>	Return specific enthalpy from p, s, and X or Xi
<a href="#">AbsolutePressure</a>	Type for absolute pressure with medium specific attributes
<a href="#">Density</a>	Type for density with medium specific attributes
<a href="#">DynamicViscosity</a>	Type for dynamic viscosity with medium specific attributes
<a href="#">EnthalpyFlowRate</a>	Type for enthalpy flow rate with medium specific attributes
<a href="#">MassFlowRate</a>	Type for mass flow rate with medium specific attributes
<a href="#">MassFraction</a>	Type for mass fraction with medium specific attributes
<a href="#">MoleFraction</a>	Type for mole fraction with medium specific attributes
<a href="#">MolarMass</a>	Type for molar mass with medium specific attributes
<a href="#">MolarVolume</a>	Type for molar volume with medium specific attributes
<a href="#">IsentropicExponent</a>	Type for isentropic exponent with medium specific attributes
<a href="#">SpecificEnergy</a>	Type for specific energy with medium specific attributes
<a href="#">SpecificInternalEnergy</a>	Type for specific internal energy with medium specific attributes
<a href="#">SpecificEnthalpy</a>	Type for specific enthalpy with medium specific attributes
<a href="#">SpecificEntropy</a>	Type for specific entropy with medium specific attributes
<a href="#">SpecificHeatCapacity</a>	Type for specific heat capacity with medium specific attributes
<a href="#">SurfaceTension</a>	Type for surface tension with medium specific attributes
<a href="#">Temperature</a>	Type for temperature with medium specific attributes
<a href="#">ThermalConductivity</a>	Type for thermal conductivity with medium specific attributes
<a href="#">PrandtlNumber</a>	Type for Prandtl number with medium specific attributes
<a href="#">VelocityOfSound</a>	Type for velocity of sound with medium specific attributes
<a href="#">ExtraProperty</a>	Type for unspecified, mass-specific property transported by flow
<a href="#">CumulativeExtraProperty</a>	Type for conserved integral of unspecified, mass specific property
<a href="#">ExtraPropertyFlowRate</a>	Type for flow rate of unspecified, mass-specific property
<a href="#">IsobaricExpansionCoefficient</a>	Type for isobaric expansion coefficient with medium specific attributes
<a href="#">DipoleMoment</a>	Type for dipole moment with medium specific attributes
<a href="#">DerDensityByPressure</a>	Type for partial derivative of density with resect to pressure with medium specific attributes
<a href="#">DerDensityByEnthalpy</a>	Type for partial derivative of density with resect to enthalpy with medium specific attributes
<a href="#">DerEnthalpyByPressure</a>	Type for partial derivative of enthalpy with resect to pressure with medium specific attributes
<a href="#">DerDensityByTemperature</a>	Type for partial derivative of density with resect to temperature with medium specific attributes
 <a href="#">Choices</a>	Types, constants to define menu choices

## Types and constants

```
constant Boolean excludeEnthalpyOfFormation=true
"If true, enthalpy of formation Hf is not included in specific enthalpy h";
```

```

constant ReferenceEnthalpy.Temp referenceChoice=Choices.
  ReferenceEnthalpy.ZeroAt0K "Choice of reference enthalpy";

constant SpecificEnthalpy h_offset=0.0
  "User defined offset for reference enthalpy, if referenceChoice =
UserDefined";

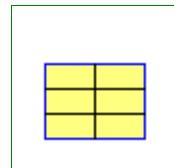
constant IdealGases.Common.DataRecord data
  "Data record of ideal gas substance";

constant FluidConstants[nS] fluidConstants "constant data for the fluid";

```

## Modelica.Media.IdealGases.Common.SingleGasNasa.ThermodynamicState

**thermodynamic state variables for ideal gases**



### Modelica definition

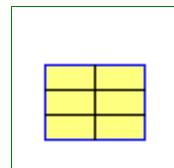
```

redeclare replaceable record extends ThermodynamicState
  "thermodynamic state variables for ideal gases"
  AbsolutePressure p "Absolute pressure of medium";
  Temperature T "Temperature of medium";
end ThermodynamicState;

```

## Modelica.Media.IdealGases.Common.SingleGasNasa.FluidConstants

**extended fluid constants**



### Modelica definition

```

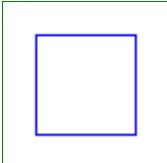
redeclare replaceable record extends FluidConstants
  "extended fluid constants"
  Temperature criticalTemperature "critical temperature";
  AbsolutePressure criticalPressure "critical pressure";
  MolarVolume criticalMolarVolume "critical molar Volume";
  Real acentricFactor "Pitzer acentric factor";
  Temperature triplePointTemperature "triple point temperature";
  AbsolutePressure triplePointPressure "triple point pressure";
  Temperature meltingPoint "melting point at 101325 Pa";
  Temperature normalBoilingPoint "normal boiling point (at 101325 Pa)";
  DipoleMoment dipoleMoment
    "dipole moment of molecule in Debye (1 debye = 3.33564e10-30 C.m)";
  Boolean hasIdealGasHeatCapacity=false
    "true if ideal gas heat capacity is available";
  Boolean hasCriticalData=false "true if critical data are known";
  Boolean hasDipoleMoment=false "true if a dipole moment known";
  Boolean hasFundamentalEquation=false "true if a fundamental equation";
  Boolean hasLiquidHeatCapacity=false
    "true if liquid heat capacity is available";
  Boolean hasSolidHeatCapacity=false "true if solid heat capacity is available";
  Boolean hasAccurateViscosityData=false
    "true if accurate data for a viscosity function is available";

```

```

Boolean hasAccurateConductivityData=false
  "true if accurate data for thermal conductivity is available";
Boolean hasVapourPressureCurve=false
  "true if vapour pressure data, e.g. Antoine coefficents are known";
Boolean hasAcentricFactor=false "true if Pitzer acentric factor is known";
SpecificEnthalpy HCRIT0=0.0
  "Critical specific enthalpy of the fundamental equation";
SpecificEntropy SCRIT0=0.0
  "Critical specific entropy of the fundamental equation";
SpecificEnthalpy deltah=0.0
  "Difference between specific enthalpy model ( $h_m$ ) and f.eq. ( $h_f$ ) ( $h_m - h_f$ )";
SpecificEntropy deltas=0.0
  "Difference between specific enthalpy model ( $s_m$ ) and f.eq. ( $s_f$ ) ( $s_m - s_f$ )";
end FluidConstants;

```

**Modelica.Media.IdealGases.Common.SingleGasNasa.BaseProperties****Base properties of ideal gas medium****Parameters**

Name	Default	Description
standardOrderComponents	true	if true, last element in components is computed from 1-sum(Xi)
<b>Advanced</b>		
preferredMediumStates	false	= true if StateSelect.prefer shall be used for the independent property variables of the medium

**Modelica.Media.IdealGases.Common.SingleGasNasa.setState\_pTX****Return thermodynamic state as function of p, T and composition X****Inputs**

Name	Default	Description
p		Pressure [Pa]
T		Temperature [K]
X[:]	reference_X	Mass fractions [kg/kg]

**Outputs**

Name	Description
state	

**Modelica.Media.IdealGases.Common.SingleGasNasa.setState\_phX****Return thermodynamic state as function of p, h and composition X**

## Inputs

Name	Default	Description
p		Pressure [Pa]
h		Specific enthalpy [J/kg]
X[:]	reference_X	Mass fractions [kg/kg]

## Outputs

Name	Description
state	

---

## **Modelica.Media.IdealGases.Common.SingleGasNasa.setState\_psX**

Return thermodynamic state as function of p, s and composition X



## Inputs

Name	Default	Description
p		Pressure [Pa]
s		Specific entropy [J/(kg.K)]
X[:]	reference_X	Mass fractions [kg/kg]

## Outputs

Name	Description
state	

---

## **Modelica.Media.IdealGases.Common.SingleGasNasa.setState\_dTX**

Return thermodynamic state as function of d, T and composition X



## Inputs

Name	Default	Description
d		density [kg/m3]
T		Temperature [K]
X[:]	reference_X	Mass fractions [kg/kg]

## Outputs

Name	Description
state	

---

## **Modelica.Media.IdealGases.Common.SingleGasNasa.pressure**

return pressure of ideal gas



---

## 850 Modelica.Media.IdealGases.Common.SingleGasNasa.pressure

---

### Inputs

Name	Default	Description
state		

### Outputs

Name	Description
p	Pressure [Pa]

---

## Modelica.Media.IdealGases.Common.SingleGasNasa.temperature

return temperature of ideal gas



### Inputs

Name	Default	Description
state		

### Outputs

Name	Description
T	Temperature [K]

---

## Modelica.Media.IdealGases.Common.SingleGasNasa.density

return density of ideal gas



### Inputs

Name	Default	Description
state		

### Outputs

Name	Description
d	Density [kg/m3]

---

## Modelica.Media.IdealGases.Common.SingleGasNasa.specificEnthalpy

Return specific enthalpy



### Inputs

Name	Default	Description
state		

### Outputs

Name	Description
h	Specific enthalpy [J/kg]

**Modelica.Media.IdealGases.Common.SingleGasNasa.specificInternalEnergy**

Return specific internal energy

**Inputs**

Name	Default	Description
state		

**Outputs**

Name	Description
u	Specific internal energy [J/kg]

**Modelica.Media.IdealGases.Common.SingleGasNasa.specificEntropy**

Return specific entropy

**Inputs**

Name	Default	Description
state		

**Outputs**

Name	Description
s	Specific entropy [J/(kg.K)]

**Modelica.Media.IdealGases.Common.SingleGasNasa.specificGibbsEnergy**

Return specific Gibbs energy

**Inputs**

Name	Default	Description
state		

**Outputs**

Name	Description
g	Specific Gibbs energy [J/kg]

**Modelica.Media.IdealGases.Common.SingleGasNasa.specificHelmholtzEnergy**

Return specific Helmholtz energy

**Inputs**

Name	Default	Description
state		

## Outputs

Name	Description
f	Specific Helmholtz energy [J/kg]

---

## Modelica.Media.IdealGases.Common.SingleGasNasa.specificHeatCapacityCp

Return specific heat capacity at constant pressure



## Inputs

Name	Default	Description
state		

## Outputs

Name	Description
cp	Specific heat capacity at constant pressure [J/(kg.K)]

---

## Modelica.Media.IdealGases.Common.SingleGasNasa.specificHeatCapacityCv

Compute specific heat capacity at constant volume from temperature and gas data



## Inputs

Name	Default	Description
state		

## Outputs

Name	Description
cv	Specific heat capacity at constant volume [J/(kg.K)]

---

## Modelica.Media.IdealGases.Common.SingleGasNasa.isentropicExponent

Return isentropic exponent



## Inputs

Name	Default	Description
state		

## Outputs

Name	Description
gamma	Isentropic exponent [1]

---

## Modelica.Media.IdealGases.Common.SingleGasNasa.velocityOfSound

Return velocity of sound



## Inputs

Name	Default	Description
state		

## Outputs

Name	Description
a	Velocity of sound [m/s]

## Modelica.Media.IdealGases.Common.SingleGasNasa.isentropicEnthalpyApproximation

approximate method of calculating h\_is from upstream properties and downstream pressure



## Inputs

Name	Default	Description
p2		downstream pressure [Pa]
state		properties at upstream location
exclEnthForm	excludeEnthalpyOfFormation	If true, enthalpy of formation Hf is not included in specific enthalpy h
refChoice	referenceChoice	Choice of reference enthalpy
h_offset	h_offset	User defined offset for reference enthalpy, if referenceChoice = UserDefined [J/kg]

## Outputs

Name	Description
h_is	isentropic enthalpy [J/kg]

## Modelica.Media.IdealGases.Common.SingleGasNasa.isentropicEnthalpy

Return isentropic enthalpy



## Inputs

Name	Default	Description
exclEnthForm	excludeEnthalpyOfFormation	If true, enthalpy of formation Hf is not included in specific enthalpy h
refChoice	referenceChoice	Choice of reference enthalpy
h_offset	h_offset	User defined offset for reference enthalpy, if referenceChoice = UserDefined [J/kg]
p_downstream		downstream pressure [Pa]
refState		reference state for entropy

## Outputs

Name	Description

h_is	Isentropic enthalpy [J/kg]
------	----------------------------

---

**Modelica.Media.IdealGases.Common.SingleGasNasa.isobaricExpansionCoefficient**



Returns overall the isobaric expansion coefficient beta

#### Inputs

Name	Default	Description
state		

#### Outputs

Name	Description
beta	Isobaric expansion coefficient [1/K]

---

**Modelica.Media.IdealGases.Common.SingleGasNasa.isotheCompressibility**



Returns overall the isothermal compressibility factor

#### Inputs

Name	Default	Description
state		

#### Outputs

Name	Description
kappa	Isothermal compressibility [1/Pa]

---

**Modelica.Media.IdealGases.Common.SingleGasNasa.density\_derP\_T**



density derivative by temperature at constant pressure

#### Inputs

Name	Default	Description
state		

#### Outputs

Name	Description
ddpT	Density derivative wrt pressure [s2/m2]

---

**Modelica.Media.IdealGases.Common.SingleGasNasa.density\_derT\_p**



density derivative by temperature at constant pressure

## Inputs

Name	Default	Description
state		

## Outputs

Name	Description
ddTp	Density derivative wrt temperature [kg/(m <sup>3</sup> .K)]

---

## Modelica.Media.IdealGases.Common.SingleGasNasa.density\_derX

density derivative by mass fraction



## Inputs

Name	Default	Description
state		

## Outputs

Name	Description
dddX[nX]	Derivative of density wrt mass fraction [kg/m <sup>3</sup> ]

---

## Modelica.Media.IdealGases.Common.SingleGasNasa.cp\_T

Compute specific heat capacity at constant pressure from temperature and gas data



## Inputs

Name	Default	Description
data		Ideal gas data
T		Temperature [K]

## Outputs

Name	Description
cp	Specific heat capacity at temperature T [J/(kg.K)]

---

## Modelica.Media.IdealGases.Common.SingleGasNasa.cp\_Tlow

Compute specific heat capacity at constant pressure, low T region



## Inputs

Name	Default	Description
data		Ideal gas data
T		Temperature [K]

---

## 856 Modelica.Media.IdealGases.Common.SingleGasNasa.cp\_Tlow

---

### Outputs

Name	Description
cp	Specific heat capacity at temperature T [J/(kg.K)]

---

### Modelica.Media.IdealGases.Common.SingleGasNasa.cp\_Tlow\_der

Compute specific heat capacity at constant pressure, low T region



### Inputs

Name	Default	Description
data		Ideal gas data
T		Temperature [K]
dT		Temperature derivative

### Outputs

Name	Description
cp_der	Derivative of specific heat capacity

---

### Modelica.Media.IdealGases.Common.SingleGasNasa.h\_T

Compute specific enthalpy from temperature and gas data; reference is decided by the refChoice input, or by the referenceChoice package constant by default



### Inputs

Name	Default	Description
data		Ideal gas data
T		Temperature [K]
exclEnthForm	excludeEnthalpyOfFormation	If true, enthalpy of formation Hf is not included in specific enthalpy h
refChoice	referenceChoice	Choice of reference enthalpy
h_offset	h_offset	User defined offset for reference enthalpy, if referenceChoice = UserDefined [J/kg]

### Outputs

Name	Description
h	Specific enthalpy at temperature T [J/kg]

---

### Modelica.Media.IdealGases.Common.SingleGasNasa.h\_T\_der

derivative function for h\_T



### Inputs

Name	Default	Description
data		Ideal gas data

T		Temperature [K]
exclEnthForm	excludeEnthalpyOfFormation	If true, enthalpy of formation Hf is not included in specific enthalpy h
refChoice	referenceChoice	Choice of reference enthalpy
h_off	h_offset	User defined offset for reference enthalpy, if referenceChoice = UserDefined [J/kg]
dT		Temperature derivative

## Outputs

Name	Description
h_der	Specific enthalpy at temperature T

---

## Modelica.Media.IdealGases.Common.SingleGasNasa.h\_Tlow

Compute specific enthalpy, low T region; reference is decided by the refChoice input, or by the referenceChoice package constant by default



## Inputs

Name	Default	Description
data		Ideal gas data
T		Temperature [K]
exclEnthForm	excludeEnthalpyOfFormation	If true, enthalpy of formation Hf is not included in specific enthalpy h
refChoice	referenceChoice	Choice of reference enthalpy
h_off	h_offset	User defined offset for reference enthalpy, if referenceChoice = UserDefined [J/kg]

## Outputs

Name	Description
h	Specific enthalpy at temperature T [J/kg]

---

## Modelica.Media.IdealGases.Common.SingleGasNasa.h\_Tlow\_der

Compute specific enthalpy, low T region; reference is decided by the refChoice input, or by the referenceChoice package constant by default



## Inputs

Name	Default	Description
data		Ideal gas data
T		Temperature [K]
exclEnthForm	excludeEnthalpyOfFormation	If true, enthalpy of formation Hf is not included in specific enthalpy h
refChoice	referenceChoice	Choice of reference enthalpy
h_off	h_offset	User defined offset for reference enthalpy, if referenceChoice = UserDefined [J/kg]
dT		Temperature derivative [K/s]

## Outputs

Name	Description
h_der	Derivative of specific enthalpy at temperature T [J/(kg.s)]

---

## Modelica.Media.IdealGases.Common.SingleGasNasa.s0\_T

Compute specific entropy from temperature and gas data



## Inputs

Name	Default	Description
data		Ideal gas data
T		Temperature [K]

## Outputs

Name	Description
s	Specific entropy at temperature T [J/(kg.K)]

---

## Modelica.Media.IdealGases.Common.SingleGasNasa.s0\_Tlow

Compute specific entropy, low T region



## Inputs

Name	Default	Description
data		Ideal gas data
T		Temperature [K]

## Outputs

Name	Description
s	Specific entropy at temperature T [J/(kg.K)]

---

## Modelica.Media.IdealGases.Common.SingleGasNasa.dynamicViscosityLowPressure

Dynamic viscosity of low pressure gases



## Information

The used formula are based on the method of Chung et al (1984, 1988) referred to in ref [1] chapter 9. The formula 9-4.10 is the one being used. The Formula is given in non-SI units, the following conversion constants were used to transform the formula to SI units:

- **Const1\_SI:** The factor  $10^{-9.5} = 10^{(-2.5)*1e-7}$  where the factor  $10^{(-2.5)}$  originates from the conversion of g/mol->kg/mol + cm<sup>3</sup>/mol->m<sup>3</sup>/mol and the factor 1e-7 is due to conversion from microPoise->Pa.s.
- **Const2\_SI:** The factor  $1/3.335641e-27 = 1e-3/3.335641e-30$  where the factor 3.335641e-30 comes from debye->C.m and 1e-3 is due to conversion from cm<sup>3</sup>/mol->m<sup>3</sup>/mol

**References:**

[1] Bruce E. Poling, John E. Prausnitz, John P. O'Connell, "The Properties of Gases and Liquids" 5th Ed. Mc Graw Hill.

**Author**

T. Skoglund, Lund, Sweden, 2004-08-31

**Inputs**

Name	Default	Description
T		Gas temperature [K]
Tc		Critical temperature of gas [K]
M		Molar mass of gas [kg/mol]
Vc		Critical molar volume of gas [m <sup>3</sup> /mol]
w		Acentric factor of gas
mu		Dipole moment of gas molecule [C.m]
k	0.0	Special correction for highly polar substances

**Outputs**

Name	Description
eta	Dynamic viscosity of gas [Pa.s]

**Modelica.Media.IdealGases.Common.SingleGasNasa.dynamicViscosity**

dynamic viscosity

**Inputs**

Name	Default	Description
state		

**Outputs**

Name	Description
eta	Dynamic viscosity [Pa.s]

**Modelica.Media.IdealGases.Common.SingleGasNasa.thermalConductivityEstimat**

e

Thermal conductivity of polyatomic gases(Eucken and Modified Eucken correlation)

**Information**

This function provides two similar methods for estimating the thermal conductivity of polyatomic gases. The Eucken method (input method == 1) gives good results for low temperatures, but it tends to give an underestimated value of the thermal conductivity ( $\lambda$ ) at higher temperatures.

The Modified Eucken method (input method == 2) gives good results for high-temperatures, but it tends to give an overestimated value of the thermal conductivity ( $\lambda$ ) at low temperatures.

## Inputs

Name	Default	Description
Cp		Constant pressure heat capacity [J/(kg.K)]
eta		Dynamic viscosity [Pa.s]
method	1	1: Eucken Method, 2: Modified Eucken Method

## Outputs

Name	Description
lambda	Thermal conductivity [W/(m.k)] [W/(m.K)]

---

## Modelica.Media.IdealGases.Common.SingleGasNasa.thermalConductivity

thermal conductivity of gas



## Inputs

Name	Default	Description
method	1	1: Eucken Method, 2: Modified Eucken Method
state		

## Outputs

Name	Description
lambda	Thermal conductivity [W/(m.K)]

---

## Modelica.Media.IdealGases.Common.SingleGasNasa.molarMass

return the molar mass of the medium



## Inputs

Name	Default	Description
state		

## Outputs

Name	Description
MM	Mixture molar mass [kg/mol]

---

## Modelica.Media.IdealGases.Common.SingleGasNasa.T\_h

Compute temperature from specific enthalpy

## Inputs

Name	Default	Description
h		Specific enthalpy [J/kg]

## Outputs

Name	Description
T	Temperature [K]

---

## Modelica.Media.IdealGases.Common.SingleGasNasa.T\_ps

Compute temperature from pressure and specific entropy

## Inputs

Name	Default	Description
p		Pressure [Pa]
s		Specific entropy [J/(kg.K)]

## Outputs

Name	Description
T	Temperature [K]

---

## Modelica.Media.IdealGases.Common.MixtureGasNasa

Medium model of a mixture of ideal gases based on NASA source

## Information

This model calculates the medium properties for single component ideal gases.

### Sources for model and literature:

Original Data: Computer program for calculation of complex chemical equilibrium compositions and applications. Part 1: Analysis Document ID: 19950013764 N (95N20180) File Series: NASA Technical Reports Report Number: NASA-RP-1311 E-8017 NAS 1.61:1311 Authors: Gordon, Sanford (NASA Lewis Research Center) McBride, Bonnie J. (NASA Lewis Research Center) Published: Oct 01, 1994.

### Known limits of validity:

The data is valid for temperatures between 200 K and 6000 K. A few of the data sets for monatomic gases have a discontinuous 1st derivative at 1000 K, but this never caused problems so far.

This model has been copied from the ThermoFluid library. It has been developed by Hubertus Tummescheit.

## Package Content

Name	Description
data	Data records of ideal gas substances
excludeEnthalpyOfFormation=true	If true, enthalpy of formation Hf is not included in specific enthalpy h
referenceChoice=Choices.ReferenceEnthalpy.ZeroAt0K	Choice of reference enthalpy
h_offset=0.0	User defined offset for reference enthalpy, if referenceChoice = UserDefined
MMX=data[:,].MM	molar masses of components
<input checked="" type="checkbox"/> BaseProperties	

(f) <code>setState_pTX</code>	Return thermodynamic state as function of p, T and composition X
(f) <code>setState_phX</code>	Return thermodynamic state as function of p, h and composition X
(f) <code>setState_psX</code>	Return thermodynamic state as function of p, s and composition X
(f) <code>setState_dTX</code>	Return thermodynamic state as function of d, T and composition X
(f) <code>pressure</code>	Return pressure of ideal gas
(f) <code>temperature</code>	Return temperature of ideal gas
(f) <code>density</code>	Return density of ideal gas
(f) <code>specificEnthalpy</code>	Return specific enthalpy
(f) <code>specificInternalEnergy</code>	Return specific internal energy
(f) <code>specificEntropy</code>	Return specific entropy
(f) <code>specificGibbsEnergy</code>	Return specific Gibbs energy
(f) <code>specificHelmholtzEnergy</code>	Return specific Helmholtz energy
(f) <code>h_TX</code>	Return specific enthalpy
(f) <code>h_TX_der</code>	Return specific enthalpy derivative
(f) <code>gasConstant</code>	Return gasConstant
(f) <code>specificHeatCapacityCp</code>	Return specific heat capacity at constant pressure
(f) <code>specificHeatCapacityCv</code>	Return specific heat capacity at constant volume from temperature and gas data
(f) <code>MixEntropy</code>	Return mixing entropy of ideal gases / R
(f) <code>s_TX</code>	Return temperature dependent part of the entropy, expects full entropy vector
(f) <code>isentropicExponent</code>	Return isentropic exponent
(f) <code>velocityOfSound</code>	Return velocity of sound
(f) <code>isentropicEnthalpyApproximation</code>	Approximate method of calculating h_is from upstream properties and downstream pressure
(f) <code>isentropicEnthalpy</code>	Return isentropic enthalpy
(f) <code>gasMixtureViscosity</code>	Return viscosities of gas mixtures at low pressures (Wilke method)
(f) <code>dynamicViscosity</code>	Return mixture dynamic viscosity
(f) <code>mixtureViscosityChung</code>	Return the viscosity of gas mixtures without access to component viscosities (Chung, et. al. rules)
(f) <code>lowPressureThermalConductivity</code>	Return thermal conductivities of low-pressure gas mixtures (Mason and Saxena Modification)
(f) <code>thermalConductivity</code>	Return thermal conductivity for low pressure gas mixtures
(f) <code>isobaricExpansionCoefficient</code>	Return isobaric expansion coefficient beta
(f) <code>isothermalCompressibility</code>	Return isothermal compressibility factor
(f) <code>density_derP_T</code>	Return density derivative by temperature at constant pressure
(f) <code>density_derT_p</code>	Return density derivative by temperature at constant

	pressure
(f) density_derX	Return density derivative by mass fraction
(f) molarMass	Return molar mass of mixture
(f) T_hX	Return temperature from specific enthalpy and mass fraction
(f) T_psX	Return temperature from pressure, specific entropy and mass fraction
<b>Inherited</b>	
ThermodynamicState	thermodynamic state variables
FluidConstants	extended fluid constants
fluidConstants	constant data for the fluid
(f) moleToMassFractions	Return mass fractions X from mole fractions
(f) massToMoleFractions	Return mole fractions from mass fractions X
mediumName="unusablePartialMedium"	Name of the medium
substanceNames={mediumName}	Names of the mixture substances. Set substanceNames={mediumName} if only one substance.
extraPropertiesNames=fill("", 0)	Names of the additional (extra) transported properties. Set extraPropertiesNames=fill("",0) if unused
singleState	= true, if u and d are not a function of pressure
reducedX=true	= true if medium contains the equation sum(X) = 1.0; set reducedX=true if only one substance (see docu for details)
fixedX=false	= true if medium contains the equation X = reference_X
reference_p=101325	Reference pressure of Medium: default 1 atmosphere
reference_T=298.15	Reference temperature of Medium: default 25 deg Celsius
reference_X=if nX == 0 then fill(0, nX) else fill(1/nX, nX)	Default mass fractions of medium
p_default=101325	Default value for pressure of medium (for initialization)
T_default=Modelica.Sunits.Conversions.from_degC(20)	Default value for temperature of medium (for initialization)
h_default=specificEnthalpy_pTX(p_default, T_default, X_default)	Default value for specific enthalpy of medium (for initialization)
X_default=reference_X	Default value for mass fractions of medium (for initialization)
nS=size(substanceNames, 1)	Number of substances
nX=if nS == 1 then 0 else nS	Number of mass fractions (= 0, if only one substance)
nXi=if fixedX then 0 else if reducedX then nS - 1 else nX	Number of structurally independent mass fractions (see docu for details)
nC=size(extraPropertiesNames, 1)	Number of extra (outside of standard mass-balance) transported properties
BasePropertiesRecord	Variables contained in every instance of BaseProperties
(f) prandtlNumber	Return the Prandtl number
(f) heatCapacity_cp	alias for deprecated name
(f) heatCapacity_cv	alias for deprecated name
(f) beta	alias for isobaricExpansionCoefficient for user convenience
(f) kappa	alias of isothermalCompressibility for user convenience
(f) density_derP_h	Return density derivative wrt pressure at const specific

	enthalpy
(f) density_derh_p	Return density derivative wrt specific enthalpy at constant pressure
(f) specificEnthalpy_pTX	Return specific enthalpy from p, T, and X or Xi
(f) density_pTX	Return density from p, T, and X or Xi
(f) temperature_phX	Return temperature from p, h, and X or Xi
(f) density_phX	Return density from p, h, and X or Xi
(f) temperature_psX	Return temperature from p,s, and X or Xi
(f) density_psX	Return density from p, s, and X or Xi
(f) specificEnthalpy_psX	Return specific enthalpy from p, s, and X or Xi
AbsolutePressure	Type for absolute pressure with medium specific attributes
Density	Type for density with medium specific attributes
DynamicViscosity	Type for dynamic viscosity with medium specific attributes
EnthalpyFlowRate	Type for enthalpy flow rate with medium specific attributes
MassFlowRate	Type for mass flow rate with medium specific attributes
MassFraction	Type for mass fraction with medium specific attributes
MoleFraction	Type for mole fraction with medium specific attributes
MolarMass	Type for molar mass with medium specific attributes
MolarVolume	Type for molar volume with medium specific attributes
IsentropicExponent	Type for isentropic exponent with medium specific attributes
SpecificEnergy	Type for specific energy with medium specific attributes
SpecificInternalEnergy	Type for specific internal energy with medium specific attributes
SpecificEnthalpy	Type for specific enthalpy with medium specific attributes
SpecificEntropy	Type for specific entropy with medium specific attributes
SpecificHeatCapacity	Type for specific heat capacity with medium specific attributes
SurfaceTension	Type for surface tension with medium specific attributes
Temperature	Type for temperature with medium specific attributes
ThermalConductivity	Type for thermal conductivity with medium specific attributes
PrandtlNumber	Type for Prandtl number with medium specific attributes
VelocityOfSound	Type for velocity of sound with medium specific attributes
ExtraProperty	Type for unspecified, mass-specific property transported by flow
CumulativeExtraProperty	Type for conserved integral of unspecified, mass specific property
ExtraPropertyFlowRate	Type for flow rate of unspecified, mass-specific property
IsobaricExpansionCoefficient	Type for isobaric expansion coefficient with medium specific attributes
DipoleMoment	Type for dipole moment with medium specific attributes
DerDensityByPressure	Type for partial derivative of density with respect to pressure with medium specific attributes
DerDensityByEnthalpy	Type for partial derivative of density with respect to enthalpy with medium specific attributes
DerEnthalpyByPressure	Type for partial derivative of enthalpy with respect to

	pressure with medium specific attributes
DerDensityByTemperature	Type for partial derivative of density with respect to temperature with medium specific attributes
Choices	Types, constants to define menu choices

## Types and constants

```

constant Modelica.Media.IdealGases.Common.DataRecord [:] data
  "Data records of ideal gas substances";

constant Boolean excludeEnthalpyOfFormation=true
  "If true, enthalpy of formation Hf is not included in specific enthalpy h";

constant Choices.ReferenceEnthalpy.Temp referenceChoice=Choices.
  ReferenceEnthalpy.ZeroAt0K "Choice of reference enthalpy";

constant SpecificEnthalpy h_offset=0.0
  "User defined offset for reference enthalpy, if referenceChoice =
UserDefined";

constant MolarMass[nX] MMX=data[:].MM "molar masses of components";

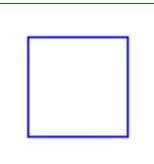
```

---

## Modelica.Media.IdealGases.Common.MixtureGasNasa.BaseProperties

### Parameters

Name	Default	Description
standardOrderComponents	true	if true, last element in components is computed from 1-sum(Xi)
<b>Advanced</b>		
preferredMediumStates	false	= true if StateSelect.prefer shall be used for the independent property variables of the medium




---

## Modelica.Media.IdealGases.Common.MixtureGasNasa.setState\_pTX

Return thermodynamic state as function of p, T and composition X



### Inputs

Name	Default	Description
p		Pressure [Pa]
T		Temperature [K]
X[:]	reference_X	Mass fractions [kg/kg]

### Outputs

Name	Description
state	

---

**866 Modelica.Media.IdealGases.Common.MixtureGasNasa.setState\_phX****Modelica.Media.IdealGases.Common.MixtureGasNasa.setState\_phX**

Return thermodynamic state as function of p, h and composition X

**Inputs**

Name	Default	Description
p		Pressure [Pa]
h		Specific enthalpy [J/kg]
X[:]	reference_X	Mass fractions [kg/kg]

**Outputs**

Name	Description
state	

---

**Modelica.Media.IdealGases.Common.MixtureGasNasa.setState\_psX**

Return thermodynamic state as function of p, s and composition X

**Inputs**

Name	Default	Description
p		Pressure [Pa]
s		Specific entropy [J/(kg.K)]
X[:]	reference_X	Mass fractions [kg/kg]

**Outputs**

Name	Description
state	

---

**Modelica.Media.IdealGases.Common.MixtureGasNasa.setState\_dTX**

Return thermodynamic state as function of d, T and composition X

**Inputs**

Name	Default	Description
d		density [kg/m3]
T		Temperature [K]
X[:]	reference_X	Mass fractions [kg/kg]

**Outputs**

Name	Description
state	

---

**Modelica.Media.IdealGases.Common.MixtureGasNasa.pressure**

Return pressure of ideal gas

**Inputs**

Name	Default	Description
state		

**Outputs**

Name	Description
p	Pressure [Pa]

**Modelica.Media.IdealGases.Common.MixtureGasNasa.temperature**

Return temperature of ideal gas

**Inputs**

Name	Default	Description
state		

**Outputs**

Name	Description
T	Temperature [K]

**Modelica.Media.IdealGases.Common.MixtureGasNasa.density**

Return density of ideal gas

**Inputs**

Name	Default	Description
state		

**Outputs**

Name	Description
d	Density [kg/m3]

**Modelica.Media.IdealGases.Common.MixtureGasNasa.specificEnthalpy**

Return specific enthalpy

**Inputs**

Name	Default	Description
state		

**Outputs**

Name	Description
h	Specific enthalpy [J/kg]

**Modelica.Media.IdealGases.Common.MixtureGasNasa.specificInternalEnergy**

Return specific internal energy

**Inputs**

Name	Default	Description
state		

**Outputs**

Name	Description
u	Specific internal energy [J/kg]

---

**Modelica.Media.IdealGases.Common.MixtureGasNasa.specificEntropy**

Return specific entropy

**Inputs**

Name	Default	Description
state		

**Outputs**

Name	Description
s	Specific entropy [J/(kg.K)]

---

**Modelica.Media.IdealGases.Common.MixtureGasNasa.specificGibbsEnergy**

Return specific Gibbs energy

**Inputs**

Name	Default	Description
state		

**Outputs**

Name	Description
g	Specific Gibbs energy [J/kg]

---

**Modelica.Media.IdealGases.Common.MixtureGasNasa.specificHelmholtzEnergy**

Return specific Helmholtz energy

**Inputs**

Name	Default	Description
state		

## Outputs

Name	Description
f	Specific Helmholtz energy [J/kg]

---

## Modelica.Media.IdealGases.Common.MixtureGasNasa.h\_TX

Return specific enthalpy



## Inputs

Name	Default	Description
T		Temperature [K]
X[:]	reference_X	Independent Mass fractions of gas mixture [kg/kg]
exclEnthForm	excludeEnthalpyOfFormation	If true, enthalpy of formation Hf is not included in specific enthalpy h
refChoice	referenceChoice	Choice of reference enthalpy
h_offset	h_offset	User defined offset for reference enthalpy, if referenceChoice = UserDefined [J/kg]

## Outputs

Name	Description
h	Specific enthalpy at temperature T [J/kg]

---

## Modelica.Media.IdealGases.Common.MixtureGasNasa.h\_TX\_der

Return specific enthalpy derivative



## Inputs

Name	Default	Description
T		Temperature [K]
X[nX]		Independent Mass fractions of gas mixture [kg/kg]
exclEnthForm	excludeEnthalpyOfFormation	If true, enthalpy of formation Hf is not included in specific enthalpy h
refChoice	referenceChoice	Choice of reference enthalpy
h_offset	h_offset	User defined offset for reference enthalpy, if referenceChoice = UserDefined [J/kg]
dT		Temperature derivative
dX[nX]		independent mass fraction derivative

## Outputs

Name	Description
h_der	Specific enthalpy at temperature T

---

## 870 Modelica.Media.IdealGases.Common.MixtureGasNasa.gasConstant

---

### Modelica.Media.IdealGases.Common.MixtureGasNasa.gasConstant



Return gasConstant

#### Inputs

Name	Default	Description
state		thermodynamic state

#### Outputs

Name	Description
R	mixture gas constant [J/(kg.K)]

---

### Modelica.Media.IdealGases.Common.MixtureGasNasa.specificHeatCapacityCp



Return specific heat capacity at constant pressure

#### Inputs

Name	Default	Description
state		

#### Outputs

Name	Description
cp	Specific heat capacity at constant pressure [J/(kg.K)]

---

### Modelica.Media.IdealGases.Common.MixtureGasNasa.specificHeatCapacityCv



Return specific heat capacity at constant volume from temperature and gas data

#### Inputs

Name	Default	Description
state		

#### Outputs

Name	Description
cv	Specific heat capacity at constant volume [J/(kg.K)]

---

### Modelica.Media.IdealGases.Common.MixtureGasNasa.MixEntropy



Return mixing entropy of ideal gases / R

#### Inputs

Name	Default	Description
x[:]		mole fraction of mixture [1]

## Outputs

Name	Description
smix	mixing entropy contribution, divided by gas constant

---

## Modelica.Media.IdealGases.Common.MixtureGasNasa.s\_TX

Return temperature dependent part of the entropy, expects full entropy vector

## Inputs

Name	Default	Description
T		temperature [K]
X[nX]		mass fraction [kg/kg]

## Outputs

Name	Description
s	specific entropy [J/(kg.K)]

---

## Modelica.Media.IdealGases.Common.MixtureGasNasa.isentropicExponent

Return isentropic exponent



## Inputs

Name	Default	Description
state		

## Outputs

Name	Description
gamma	Isentropic exponent [1]

---

## Modelica.Media.IdealGases.Common.MixtureGasNasa.velocityOfSound

Return velocity of sound



## Inputs

Name	Default	Description
state		properties at upstream location

## Outputs

Name	Description
a	Velocity of sound [m/s]

## Modelica.Media.IdealGases.Common.MixtureGasNasa.isentropicEnthalpyApproximation



Approximate method of calculating  $h_{is}$  from upstream properties and downstream pressure

### Inputs

Name	Default	Description
p2		downstream pressure [Pa]
state		thermodynamic state at upstream location

### Outputs

Name	Description
h_is	isentropic enthalpy [J/kg]

## Modelica.Media.IdealGases.Common.MixtureGasNasa.isentropicEnthalpy



Return isentropic enthalpy

### Inputs

Name	Default	Description
exact	false	flag whether exact or approximate version should be used
p_downstream		downstream pressure [Pa]
refState		reference state for entropy

### Outputs

Name	Description
h_is	Isentropic enthalpy [J/kg]

## Modelica.Media.IdealGases.Common.MixtureGasNasa.gasMixtureViscosity



Return viscosities of gas mixtures at low pressures (Wilke method)

### Information

Simplification of the kinetic theory (Chapman and Enskog theory) approach neglecting the second-order effects.

This equation has been extensively tested (Amdur and Mason, 1958; Bromley and Wilke, 1951; Cheung, 1958; Dahler, 1959; Gandhi and Saxena, 1964; Ranz and Brodowsky, 1962; Saxena and Gambhir, 1963a; Strunk, et al., 1964; Vanderslice, et al. 1962; Wright and Gray, 1962). In most cases, only nonpolar mixtures were compared, and very good results obtained. For some systems containing hydrogen as one component, less satisfactory agreement was noted. Wilke's method predicted mixture viscosities that were larger than experimental for the H<sub>2</sub>-N<sub>2</sub> system, but for H<sub>2</sub>-NH<sub>3</sub>, it underestimated the viscosities.

Gururaja, et al. (1967) found that this method also overpredicted in the H<sub>2</sub>-O<sub>2</sub> case but was quite accurate for the H<sub>2</sub>-CO<sub>2</sub> system.

Wilke's approximation has proved reliable even for polar-polar gas mixtures of aliphatic alcohols (Reid and Belenyessy, 1960). The principal reservation appears to lie in those cases where  $M_i > M_j$  and  $\eta_{i\bar{i}} > \eta_{j\bar{j}}$ .

## Inputs

Name	Default	Description
yi[:]		Mole fractions [mol/mol]
M[:]		Mole masses [kg/mol]
eta[:]		Pure component viscosities [Pa.s]

## Outputs

Name	Description
etam	Viscosity of the mixture [Pa.s]

## Modelica.Media.IdealGases.Common.MixtureGasNasa.dynamicViscosity



Return mixture dynamic viscosity

## Inputs

Name	Default	Description
state		

## Outputs

Name	Description
eta	Dynamic viscosity [Pa.s]

## Modelica.Media.IdealGases.Common.MixtureGasNasa.mixtureViscosityChung



Return the viscosity of gas mixtures without access to component viscosities (Chung, et. al. rules)

## Information

Equation to estimate the viscosity of gas mixtures at low pressures.

It is a simplification of an extension of the rigorous kinetic theory of Chapman and Enskog to determine the viscosity of multicomponent mixtures, at low pressures and with a factor to correct for molecule shape and polarity.

The input argument Kappa is a special correction for highly polar substances such as alcohols and acids.  
Values of kappa for a few such materials:

Compound	Kappa	Compound	Kappa
Methanol	0.215	n-Pentanol	0.122
Ethanol	0.175	n-Hexanol	0.114
n-Propanol	0.143	n-Heptanol	0.109
i-Propanol	0.143	Acetic Acid	0.0916
n-Butanol	0.132	Water	0.076
i-Butanol	0.132		

Chung, et al. (1984) suggest that for other alcohols not shown in the table:

$$\text{kappa} = 0.0682 + 4.704 * [(\text{number of } -\text{OH groups})] / [\text{molecular weight}]$$

S.I. units relation for the debyes:

$$1 \text{ debye} = 3.162e-25 (\text{J}\cdot\text{m}^3)^{(1/2)}$$

## References

- [1] THE PROPERTIES OF GASES AND LIQUIDS, Fifth Edition,  
Bruce E. Poling, John M. Prausnitz, John P. O'Connell.
- [2] Chung, T.-H., M. Ajlan, L. L. Lee, and K. E. Starling: Ind. Eng. Chem. Res., 27: 671 (1988).
- [3] Chung, T.-H., L. L. Lee, and K. E. Starling; Ing. Eng. Chem. Fundam., 23: 3 ()1984).

## Inputs

Name	Default	Description
T		Temperature [K]
Tc[:]		Critical temperatures [K]
Vcrit[:]		Critical volumes (m <sup>3</sup> /mol) [m <sup>3</sup> /mol]
w[:]		Acentric factors
mu[:]		Dipole moments (debyes)
MolecularWeights[:]		Molecular weights (kg/mol) [kg/mol]
y[:]		Molar Fractions [mol/mol]
kappa[:]	zeros(nX)	Association Factors

## Outputs

Name	Description
etaMixture	Mixture viscosity (Pa.s) [Pa.s]

## Modelica.Media.IdealGases.Common.MixtureGasNasa.lowPressureThermalConductivity

Return thermal conductivities of low-pressure gas mixtures (Mason and Saxena Modification)



## Information

This function applies the Masson and Saxena modification of the Wassiljewa Equation for the thermal conductivity for gas mixtures of n elements at low pressure.

For nonpolar gas mixtures errors will generally be less than 3 to 4%. For mixtures of nonpolar-polar and polar-polar gases, errors greater than 5 to 8% may be expected. For mixtures in which the sizes and polarities of the constituent molecules are not greatly different, the thermal conductivity can be estimated satisfactorily by a mole fraction average of the pure component conductivities.

## Inputs

Name	Default	Description
y[:]		Mole fraction of the components in the gass mixture [mol/mol]
T		Temperature [K]
Tc[:]		Critical temperatures [K]
Pc[:]		Critical pressures [Pa]
M[:]		Molecular weights [kg/mol]
lambda[:]		Thermal conductivities of the pure gases [W/(m.K)]

## Outputs

Name	Description
lambda_m	Thermal conductivity of the gas mixture [W/(m.K)]

---

## Modelica.Media.IdealGases.Common.MixtureGasNasa.thermalConductivity

Return thermal conductivity for low pressure gas mixtures



## Inputs

Name	Default	Description
method	1	method to compute single component thermal conductivity
state		

## Outputs

Name	Description
lambda	Thermal conductivity [W/(m.K)]

---

## Modelica.Media.IdealGases.Common.MixtureGasNasa.isobaricExpansionCoefficient

nt

Return isobaric expansion coefficient beta



## Inputs

Name	Default	Description
state		

## Outputs

Name	Description
beta	Isobaric expansion coefficient [1/K]

---

## Modelica.Media.IdealGases.Common.MixtureGasNasa.isoThermalCompressibility

Return isothermal compressibility factor



## Inputs

Name	Default	Description
state		

## Outputs

Name	Description
kappa	Isothermal compressibility [1/Pa]

---

**876 Modelica.Media.IdealGases.Common.MixtureGasNasa.density\_derP\_T****Modelica.Media.IdealGases.Common.MixtureGasNasa.density\_derP\_T**

Return density derivative by temperature at constant pressure

**Inputs**

Name	Default	Description
state		

**Outputs**

Name	Description
ddpT	Density derivative wrt pressure [s <sup>2</sup> /m <sup>2</sup> ]

---

**Modelica.Media.IdealGases.Common.MixtureGasNasa.density\_derT\_p**

Return density derivative by temperature at constant pressure

**Inputs**

Name	Default	Description
state		

**Outputs**

Name	Description
ddTp	Density derivative wrt temperature [kg/(m <sup>3</sup> .K)]

---

**Modelica.Media.IdealGases.Common.MixtureGasNasa.density\_derX**

Return density derivative by mass fraction

**Inputs**

Name	Default	Description
state		

**Outputs**

Name	Description
dddX[nX]	Derivative of density wrt mass fraction [kg/m <sup>3</sup> ]

---

**Modelica.Media.IdealGases.Common.MixtureGasNasa.molarMass**

Return molar mass of mixture

**Inputs**

Name	Default	Description
state		

## Outputs

Name	Description
MM	Mixture molar mass [kg/mol]

---

## Modelica.Media.IdealGases.Common.MixtureGasNasa.T\_hX

Return temperature from specific enthalpy and mass fraction

## Inputs

Name	Default	Description
h		specific enthalpy [J/kg]
X[:]		mass fractions of composition [kg/kg]

## Outputs

Name	Description
T	temperature [K]

---

## Modelica.Media.IdealGases.Common.MixtureGasNasa.T\_psX

Return temperature from pressure, specific entropy and mass fraction

## Inputs

Name	Default	Description
p		pressure [Pa]
s		specific entropy [J/(kg.K)]
X[:]		mass fractions of composition [kg/kg]

## Outputs

Name	Description
T	temperature [K]

---

## Modelica.Media.IdealGases.Common.FluidData

critical data, dipole moments and related data

## Information

### Package Content

Name	Description
N2	
O2	
CL2	
F2	

CO2	
CO	
H2	
H2O	
N2O	
NO	
NO2	
NH3	
SO2	
SO3	
Ar	
He	
Ne	
CH4	
C2H6	
C3H8	
C4H10_n_butane	
C5H12_n_pentane	
C6H14_n_hexane	
C7H16_n_heptane	
C2H4	
C3H6_propylene	
C4H8_1_butene	
C5H10_1_pentene	
C6H12_1_hexene	
C7H14_1_heptene	
C2H2_vinylidene	
C6H6	
C8H18_n_octane	
C8H10_ethylbenz	
CH3OH	
C2H5OH	
C3H7OH	
C4H9OH	

### Types and constants

```
constant SingleGasNasa.FluidConstants N2 (
    chemicalFormula = "N2",
    iupacName = "unknown",
    structureFormula = "unknown",
    casRegistryNumber = "7727-37-9",
    meltingPoint = 63.15,
    normalBoilingPoint = 77.35,
    criticalTemperature = 126.20,
    criticalPressure = 33.98e5,
    criticalMolarVolume = 90.10e-6,
    acentricFactor = 0.037,
    dipoleMoment = 0.0,
```

```
molarMass = SingleGasesData.N2.MM,
hasDipoleMoment = true,
hasIdealGasHeatCapacity=true,
hasCriticalData = true,
hasAcentricFactor = true);

constant SingleGasNasa.FluidConstants O2(
    chemicalFormula = "O2",
    iupacName = "unknown",
    structureFormula = "unknown",
    casRegistryNumber = "7782-44-7",
    meltingPoint = 54.36,
    normalBoilingPoint = 90.17,
    criticalTemperature = 154.58,
    criticalPressure = 50.43e5,
    criticalMolarVolume = 73.37e-6,
    acentricFactor = 0.022,
    dipoleMoment = 0.0,
    molarMass = SingleGasesData.O2.MM,
    hasDipoleMoment = true,
    hasIdealGasHeatCapacity=true,
    hasCriticalData = true,
    hasAcentricFactor = true);

constant SingleGasNasa.FluidConstants CL2(
    chemicalFormula = "CL2",
    iupacName = "unknown",
    structureFormula = "unknown",
    casRegistryNumber = "7782-50-5",
    meltingPoint = 172.19,
    normalBoilingPoint = 239.12,
    criticalTemperature = 417.00,
    criticalPressure = 77.00e5,
    criticalMolarVolume = 124.00e-6,
    acentricFactor = 0.069,
    dipoleMoment = 0.0,
    molarMass = SingleGasesData.CL2.MM,
    hasDipoleMoment = true,
    hasIdealGasHeatCapacity=true,
    hasCriticalData = true,
    hasAcentricFactor = true);

constant SingleGasNasa.FluidConstants F2(
    chemicalFormula = "F2",
    iupacName = "unknown",
    structureFormula = "unknown",
    casRegistryNumber = "7782-41-4",
    meltingPoint = 53.48,
    normalBoilingPoint = 84.95,
    criticalTemperature = 144.30,
    criticalPressure = 52.15e5,
    criticalMolarVolume = 66.20e-6,
    acentricFactor = 0.051,
    dipoleMoment = 0.0,
    molarMass = SingleGasesData.F2.MM,
    hasDipoleMoment = true,
    hasIdealGasHeatCapacity=true,
    hasCriticalData = true,
```

```
hasAcentricFactor = true);

constant SingleGasNasa.FluidConstants CO2(
    chemicalFormula = "CO2",
    iupacName = "unknown",
    structureFormula = "unknown",
    casRegistryNumber = "124-38-9",
    meltingPoint = 216.58,
    normalBoilingPoint = -1.0,
    criticalTemperature = 304.12,
    criticalPressure = 73.74e5,
    criticalMolarVolume = 94.07e-6,
    acentricFactor = 0.225,
    dipoleMoment = 0.0,
    molarMass = SingleGasesData.CO2.MM,
    hasDipoleMoment = true,
    hasIdealGasHeatCapacity=true,
    hasCriticalData = true,
    hasAcentricFactor = true);

constant SingleGasNasa.FluidConstants CO(
    chemicalFormula = "CO",
    iupacName = "unknown",
    structureFormula = "unknown",
    casRegistryNumber = "630-08-0",
    meltingPoint = 68.15,
    normalBoilingPoint = 81.66,
    criticalTemperature = 132.85,
    criticalPressure = 34.94e5,
    criticalMolarVolume = 93.10e-6,
    acentricFactor = 0.045,
    dipoleMoment = 0.1,
    molarMass = SingleGasesData.CO.MM,
    hasDipoleMoment = true,
    hasIdealGasHeatCapacity=true,
    hasCriticalData = true,
    hasAcentricFactor = true);

constant SingleGasNasa.FluidConstants H2 (
    chemicalFormula = "H2",
    iupacName = "unknown",
    structureFormula = "unknown",
    casRegistryNumber = "800000-51-5",
    meltingPoint = 13.56,
    normalBoilingPoint = 20.38,
    criticalTemperature = 33.25,
    criticalPressure = 12.97e5,
    criticalMolarVolume = 65.00e-6,
    acentricFactor = -0.216,
    dipoleMoment = 0.0,
    molarMass = SingleGasesData.H2.MM,
    hasDipoleMoment = true,
    hasIdealGasHeatCapacity=true,
    hasCriticalData = true,
    hasAcentricFactor = true);

constant SingleGasNasa.FluidConstants H2O(
    chemicalFormula = "H2O",
```

```

iupacName = "unknown",
structureFormula = "unknown",
casRegistryNumber = "7732-18-5",
meltingPoint = 273.15,
normalBoilingPoint = 373.15,
criticalTemperature = 647.14,
criticalPressure = 220.64e5,
criticalMolarVolume = 55.95e-6,
acentricFactor = 0.344,
dipoleMoment = 1.8,
molarMass = SingleGasesData.H2O.MM,
hasDipoleMoment = true,
hasIdealGasHeatCapacity=true,
hasCriticalData = true,
hasAcentricFactor = true);

constant SingleGasNasa.FluidConstants N2O(
    chemicalFormula = "N2O",
    iupacName = "unknown",
    structureFormula = "unknown",
    casRegistryNumber = "10024-97-2",
    meltingPoint = 182.33,
    normalBoilingPoint = 184.67,
    criticalTemperature = 309.60,
    criticalPressure = 72.55e5,
    criticalMolarVolume = 97.00e-6,
    acentricFactor = 0.142,
    dipoleMoment = 0.2,
    molarMass = SingleGasesData.N2O.MM,
    hasDipoleMoment = true,
    hasIdealGasHeatCapacity=true,
    hasCriticalData = true,
    hasAcentricFactor = true);

constant SingleGasNasa.FluidConstants NO(
    chemicalFormula = "NO",
    iupacName = "unknown",
    structureFormula = "unknown",
    casRegistryNumber = "10102-43-9",
    meltingPoint = 109.51,
    normalBoilingPoint = 121.38,
    criticalTemperature = 180.00,
    criticalPressure = 64.80e5,
    criticalMolarVolume = 58.00e-6,
    acentricFactor = 0.582,
    dipoleMoment = 0.2,
    molarMass = SingleGasesData.NO.MM,
    hasDipoleMoment = true,
    hasIdealGasHeatCapacity=true,
    hasCriticalData = true,
    hasAcentricFactor = true);

constant SingleGasNasa.FluidConstants NO2(
    chemicalFormula = "NO2",
    iupacName = "unknown",
    structureFormula = "unknown",
    casRegistryNumber = "10102-44-0",
    meltingPoint = 261.95,

```

```
normalBoilingPoint = 294.0,
criticalTemperature = 431.35,
criticalPressure = 101.33e5,
criticalMolarVolume = 82.5e-6,
acentricFactor = 0.849,
dipoleMoment = 0.32,
molarMass = SingleGasesData.NO2.MM,
hasDipoleMoment = true,
hasIdealGasHeatCapacity=true,
hasCriticalData = true,
hasAcentricFactor = true);

constant SingleGasNasa.FluidConstants NH3(
    chemicalFormula = "NH3",
    iupacName = "unknown",
    structureFormula = "unknown",
    casRegistryNumber = "7664-41-7",
    meltingPoint = 195.41,
    normalBoilingPoint = 239.82,
    criticalTemperature = 405.40,
    criticalPressure = 113.53e5,
    criticalMolarVolume = 72.47e-6,
    acentricFactor = 0.257,
    dipoleMoment = 1.5,
    molarMass = SingleGasesData.NH3.MM,
    hasDipoleMoment = true,
    hasIdealGasHeatCapacity=true,
    hasCriticalData = true,
    hasAcentricFactor = true);

constant SingleGasNasa.FluidConstants SO2(
    chemicalFormula = "SO2",
    iupacName = "unknown",
    structureFormula = "unknown",
    casRegistryNumber = "7446-09-5",
    meltingPoint = 197.67,
    normalBoilingPoint = 263.13,
    criticalTemperature = 430.80,
    criticalPressure = 78.84e5,
    criticalMolarVolume = 122.00e-6,
    acentricFactor = 0.245,
    dipoleMoment = 1.6,
    molarMass = SingleGasesData.SO2.MM,
    hasDipoleMoment = true,
    hasIdealGasHeatCapacity=true,
    hasCriticalData = true,
    hasAcentricFactor = true);

constant SingleGasNasa.FluidConstants SO3(
    chemicalFormula = "SO3",
    iupacName = "unknown",
    structureFormula = "unknown",
    casRegistryNumber = "7446-11-9",
    meltingPoint = 289.95,
    normalBoilingPoint = 317.90,
    criticalTemperature = 490.90,
    criticalPressure = 82.10e5,
    criticalMolarVolume = 126.50e-6,
```

```

acentricFactor = 0.422,
dipoleMoment = 0.0,
molarMass = SingleGasesData.SO3.MM,
hasDipoleMoment = true,
hasIdealGasHeatCapacity=true,
hasCriticalData = true,
hasAcentricFactor = true);

constant SingleGasNasa.FluidConstants Ar(
    chemicalFormula = "Ar",
    iupacName = "unknown",
    structureFormula = "unknown",
    casRegistryNumber = "7440-37-1",
    meltingPoint = 83.80,
    normalBoilingPoint = 87.27,
    criticalTemperature = 150.86,
    criticalPressure = 48.98e5,
    criticalMolarVolume = 74.57e-6,
    acentricFactor = -0.002,
    dipoleMoment = 0.0,
    molarMass = SingleGasesData.Ar.MM,
    hasDipoleMoment = true,
    hasIdealGasHeatCapacity=true,
    hasCriticalData = true,
    hasAcentricFactor = true);

constant SingleGasNasa.FluidConstants He(
    chemicalFormula = "He",
    iupacName = "unknown",
    structureFormula = "unknown",
    casRegistryNumber = "7440-59-7",
    meltingPoint = 2.15,
    normalBoilingPoint = 4.30,
    criticalTemperature = 5.19,
    criticalPressure = 2.27e5,
    criticalMolarVolume = 57.30e-6,
    acentricFactor = -0.390,
    dipoleMoment = 0.0,
    molarMass = SingleGasesData.He.MM,
    hasDipoleMoment = true,
    hasIdealGasHeatCapacity=true,
    hasCriticalData = true,
    hasAcentricFactor = true);

constant SingleGasNasa.FluidConstants Ne(
    chemicalFormula = "Ne",
    iupacName = "unknown",
    structureFormula = "unknown",
    casRegistryNumber = "7440-01-9",
    meltingPoint = 24.56,
    normalBoilingPoint = 27.07,
    criticalTemperature = 44.40,
    criticalPressure = 27.60e5,
    criticalMolarVolume = 41.70e-6,
    acentricFactor = -0.016,
    dipoleMoment = 0.0,
    molarMass = SingleGasesData.Ne.MM,
    hasDipoleMoment = true,

```

```
hasIdealGasHeatCapacity=true,
hasCriticalData =      true,
hasAcentricFactor =    true);
```

```
constant SingleGasNasa.FluidConstants CH4(
    chemicalFormula =      "CH4",
    iupacName =            "unknown",
    structureFormula =     "unknown",
    casRegistryNumber =    "74-82-8",
    meltingPoint =         90.69,
    normalBoilingPoint =  111.66,
    criticalTemperature = 190.56,
    criticalPressure =    45.99e5,
    criticalMolarVolume = 98.60e-6,
    acentricFactor =       0.011,
    dipoleMoment =         0.0,
    molarMass =             SingleGasesData.CH4.MM,
    hasDipoleMoment =      true,
    hasIdealGasHeatCapacity=true,
    hasCriticalData =      true,
    hasAcentricFactor =    true);
```

```
constant SingleGasNasa.FluidConstants C2H6(
    chemicalFormula =      "C2H6",
    iupacName =            "unknown",
    structureFormula =     "unknown",
    casRegistryNumber =    "74-84-0",
    meltingPoint =         90.35,
    normalBoilingPoint =  184.55,
    criticalTemperature = 305.32,
    criticalPressure =    48.72e5,
    criticalMolarVolume = 145.50e-6,
    acentricFactor =       0.099,
    dipoleMoment =         0.0,
    molarMass =             SingleGasesData.C2H6.MM,
    hasDipoleMoment =      true,
    hasIdealGasHeatCapacity=true,
    hasCriticalData =      true,
    hasAcentricFactor =    true);
```

```
constant SingleGasNasa.FluidConstants C3H8(
    chemicalFormula =      "C3H8",
    iupacName =            "unknown",
    structureFormula =     "unknown",
    casRegistryNumber =    "74-98-6",
    meltingPoint =         91.45,
    normalBoilingPoint =  231.02,
    criticalTemperature = 369.83,
    criticalPressure =    42.48e5,
    criticalMolarVolume = 200.00e-6,
    acentricFactor =       0.152,
    dipoleMoment =         0.0,
    molarMass =             SingleGasesData.C3H8.MM,
    hasDipoleMoment =      true,
    hasIdealGasHeatCapacity=true,
    hasCriticalData =      true,
    hasAcentricFactor =    true);
```

```

constant SingleGasNasa.FluidConstants C4H10_n_butane(
    chemicalFormula = "C4H10",
    iupacName = "unknown",
    structureFormula = "unknown",
    casRegistryNumber = "106-97-8",
    meltingPoint = 134.79,
    normalBoilingPoint = 272.66,
    criticalTemperature = 425.12,
    criticalPressure = 37.96e5,
    criticalMolarVolume = 255.00e-6,
    acentricFactor = 0.20,
    dipoleMoment = 0.0,
    molarMass =
SingleGasesData.C4H10_n_butane.MM,
    hasDipoleMoment = true,
    hasIdealGasHeatCapacity=true,
    hasCriticalData = true,
    hasAcentricFactor = true);

constant SingleGasNasa.FluidConstants C5H12_n_pentane(
    chemicalFormula = "C5H12",
    iupacName = "unknown",
    structureFormula = "unknown",
    casRegistryNumber = "109-66-0",
    meltingPoint = 143.43,
    normalBoilingPoint = 309.22,
    criticalTemperature = 469.70,
    criticalPressure = 33.70e5,
    criticalMolarVolume = 311.00e-6,
    acentricFactor = 0.252,
    dipoleMoment = 0.0,
    molarMass =
SingleGasesData.C5H12_n_pentane.MM,
    hasDipoleMoment = true,
    hasIdealGasHeatCapacity=true,
    hasCriticalData = true,
    hasAcentricFactor = true);

constant SingleGasNasa.FluidConstants C6H14_n_hexane(
    chemicalFormula = "C6H14",
    iupacName = "unknown",
    structureFormula = "unknown",
    casRegistryNumber = "110-54-3",
    meltingPoint = 177.84,
    normalBoilingPoint = 341.88,
    criticalTemperature = 507.60,
    criticalPressure = 30.25e5,
    criticalMolarVolume = 368.00e-6,
    acentricFactor = 0.300,
    dipoleMoment = 0.0,
    molarMass =
SingleGasesData.C6H14_n_hexane.MM,
    hasDipoleMoment = true,
    hasIdealGasHeatCapacity=true,
    hasCriticalData = true,
    hasAcentricFactor = true);

constant SingleGasNasa.FluidConstants C7H16_n_heptane (

```

```
chemicalFormula = "C7H16",
iupacName = "unknown",
structureFormula = "unknown",
casRegistryNumber = "142-82-5",
meltingPoint = 182.59,
normalBoilingPoint = 371.57,
criticalTemperature = 540.20,
criticalPressure = 27.40e5,
criticalMolarVolume = 428.00e-6,
acentricFactor = 0.350,
dipoleMoment = 0.0,
molarMass =
SingleGasesData.C7H16_n_heptane.MM,
hasDipoleMoment = true,
hasIdealGasHeatCapacity=true,
hasCriticalData = true,
hasAcentricFactor = true);

constant SingleGasNasa.FluidConstants C2H4(
    chemicalFormula = "C2H4",
    iupacName = "unknown",
    structureFormula = "unknown",
    casRegistryNumber = "74-85-1",
    meltingPoint = 103.99,
    normalBoilingPoint = 169.42,
    criticalTemperature = 282.34,
    criticalPressure = 50.41e5,
    criticalMolarVolume = 131.10e-6,
    acentricFactor = 0.087,
    dipoleMoment = 0.0,
    molarMass = SingleGasesData.C2H4.MM,
    hasDipoleMoment = true,
    hasIdealGasHeatCapacity=true,
    hasCriticalData = true,
    hasAcentricFactor = true);

constant SingleGasNasa.FluidConstants C3H6_propylene(
    chemicalFormula = "C3H6",
    iupacName = "unknown",
    structureFormula = "unknown",
    casRegistryNumber = "115-07-1",
    meltingPoint = 87.89,
    normalBoilingPoint = 225.46,
    criticalTemperature = 364.90,
    criticalPressure = 46.00e5,
    criticalMolarVolume = 184.60e-6,
    acentricFactor = 0.142,
    dipoleMoment = 0.4,
    molarMass =
SingleGasesData.C3H6_propylene.MM,
    hasDipoleMoment = true,
    hasIdealGasHeatCapacity=true,
    hasCriticalData = true,
    hasAcentricFactor = true);

constant SingleGasNasa.FluidConstants C4H8_1_butene(
    chemicalFormula = "C4H8",
    iupacName = "unknown",
```

```

structureFormula =      "unknown",
casRegistryNumber =    "106-98-9",
meltingPoint =         87.79,
normalBoilingPoint =   266.92,
criticalTemperature =  419.50,
criticalPressure =     40.20e5,
criticalMolarVolume = 240.80e-6,
acentricFactor =       0.194,
dipoleMoment =         0.3,
molarMass =
SingleGasesData.C4H8_1_butene.MM,
hasDipoleMoment =      true,
hasIdealGasHeatCapacity=true,
hasCriticalData =       true,
hasAcentricFactor =    true);

constant SingleGasNasa.FluidConstants C5H10_1_pentene(
chemicalFormula =      "C5H10",
iupacName =             "unknown",
structureFormula =      "unknown",
casRegistryNumber =     "109-67-1",
meltingPoint =          106.95,
normalBoilingPoint =    303.11,
criticalTemperature =   464.80,
criticalPressure =      35.60e5,
criticalMolarVolume =  298.40e-6,
acentricFactor =        0.237,
dipoleMoment =          0.4,
molarMass =
SingleGasesData.C5H10_1_pentene.MM,
hasDipoleMoment =      true,
hasIdealGasHeatCapacity=true,
hasCriticalData =       true,
hasAcentricFactor =    true);

constant SingleGasNasa.FluidConstants C6H12_1_hexene(
chemicalFormula =      "C6H12",
iupacName =             "unknown",
structureFormula =      "unknown",
casRegistryNumber =     "592-41-6",
meltingPoint =          133.34,
normalBoilingPoint =    336.63,
criticalTemperature =   504.00,
criticalPressure =      31.43e5,
criticalMolarVolume =  355.10e-6,
acentricFactor =        0.281,
dipoleMoment =          0.4,
molarMass =
SingleGasesData.C6H12_1_hexene.MM,
hasDipoleMoment =      true,
hasIdealGasHeatCapacity=true,
hasCriticalData =       true,
hasAcentricFactor =    true);

constant SingleGasNasa.FluidConstants C7H14_1_heptene(
chemicalFormula =      "C7H14",
iupacName =             "unknown",
structureFormula =      "unknown",

```

```
casRegistryNumber = "592-76-7",
meltingPoint = 153.45,
normalBoilingPoint = 366.79,
criticalTemperature = 537.30,
criticalPressure = 29.20e5,
criticalMolarVolume = 409.00e-6,
acentricFactor = 0.343,
dipoleMoment = 0.3,
molarMass =
SingleGasesData.C7H14_1_heptene.MM,
hasDipoleMoment = true,
hasIdealGasHeatCapacity=true,
hasCriticalData = true,
hasAcentricFactor = true);

constant SingleGasNasa.FluidConstants C2H2_vinylidene(
chemicalFormula = "C2H2",
iupacName = "unknown",
structureFormula = "unknown",
casRegistryNumber = "74-86-2",
meltingPoint = 192.35,
normalBoilingPoint = 188.40,
criticalTemperature = 308.30,
criticalPressure = 61.14e5,
criticalMolarVolume = 112.20e-6,
acentricFactor = 0.189,
dipoleMoment = 0.0,
molarMass =
SingleGasesData.C2H2_vinylidene.MM,
hasDipoleMoment = true,
hasIdealGasHeatCapacity=true,
hasCriticalData = true,
hasAcentricFactor = true);

constant SingleGasNasa.FluidConstants C6H6(
chemicalFormula = "C6H6",
iupacName = "unknown",
structureFormula = "unknown",
casRegistryNumber = "71-43-2",
meltingPoint = 278.68,
normalBoilingPoint = 353.24,
criticalTemperature = 562.05,
criticalPressure = 48.95e5,
criticalMolarVolume = 256.00e-6,
acentricFactor = 0.210,
dipoleMoment = 0.0,
molarMass =
SingleGasesData.C6H6.MM,
hasDipoleMoment = true,
hasIdealGasHeatCapacity=true,
hasCriticalData = true,
hasAcentricFactor = true);

constant SingleGasNasa.FluidConstants C8H18_n_octane(
chemicalFormula = "C8H18",
iupacName = "unknown",
structureFormula = "unknown",
casRegistryNumber = "111-65-9",
meltingPoint = 216.39,
```

```

normalBoilingPoint = 398.82,
criticalTemperature = 568.70,
criticalPressure = 24.90e5,
criticalMolarVolume = 492.00e-6,
acentricFactor = 0.399,
dipoleMoment = 0.0,
molarMass =
SingleGasesData.C8H18_n_octane.MM,
hasDipoleMoment = true,
hasIdealGasHeatCapacity=true,
hasCriticalData = true,
hasAcentricFactor = true);

constant SingleGasNasa.FluidConstants C8H10_ethylbenz(
chemicalFormula = "C8H10",
iupacName = "unknown",
structureFormula = "unknown",
casRegistryNumber = "100-41-4",
meltingPoint = 178.18,
normalBoilingPoint = 409.36,
criticalTemperature = 617.15,
criticalPressure = 36.09e5,
criticalMolarVolume = 374.00e-6,
acentricFactor = 0.304,
dipoleMoment = 0.4,
molarMass =
SingleGasesData.C8H10_ethylbenz.MM,
hasDipoleMoment = true,
hasIdealGasHeatCapacity=true,
hasCriticalData = true,
hasAcentricFactor = true);

constant SingleGasNasa.FluidConstants CH3OH(
chemicalFormula = "CH3OH",
iupacName = "unknown",
structureFormula = "unknown",
casRegistryNumber = "67-56-1",
meltingPoint = 175.49,
normalBoilingPoint = 337.69,
criticalTemperature = 512.64,
criticalPressure = 80.97e5,
criticalMolarVolume = 118.00e-6,
acentricFactor = 0.565,
dipoleMoment = 1.7,
molarMass = SingleGasesData.CH3OH.MM,
hasDipoleMoment = true,
hasIdealGasHeatCapacity=true,
hasCriticalData = true,
hasAcentricFactor = true);

constant SingleGasNasa.FluidConstants C2H5OH(
chemicalFormula = "C2H5OH",
iupacName = "unknown",
structureFormula = "unknown",
casRegistryNumber = "64-17-5",
meltingPoint = 159.05,
normalBoilingPoint = 351.80,
criticalTemperature = 513.92,

```

```
criticalPressure = 61.48e5,
criticalMolarVolume = 167.00e-6,
acentricFactor = 0.649,
dipoleMoment = 1.7,
molarMass = SingleGasesData.C2H5OH.MM,
hasDipoleMoment = true,
hasIdealGasHeatCapacity=true,
hasCriticalData = true,
hasAcentricFactor = true);

constant SingleGasNasa.FluidConstants C3H7OH(
    chemicalFormula = "C3H7OH",
    iupacName = "unknown",
    structureFormula = "unknown",
    casRegistryNumber = "71-23-8",
    meltingPoint = 147.00,
    normalBoilingPoint = 370.93,
    criticalTemperature = 536.78,
    criticalPressure = 51.75e5,
    criticalMolarVolume = 219.00e-6,
    acentricFactor = 0.629,
    dipoleMoment = 1.7,
    molarMass = 60.1e-3,
    hasDipoleMoment = true,
    hasIdealGasHeatCapacity=true,
    hasCriticalData = true,
    hasAcentricFactor = true);

constant SingleGasNasa.FluidConstants C4H9OH(
    chemicalFormula = "C4H9OH",
    iupacName = "unknown",
    structureFormula = "unknown",
    casRegistryNumber = "71-36-3",
    meltingPoint = 183.35,
    normalBoilingPoint = 390.88,
    criticalTemperature = 563.05,
    criticalPressure = 44.23e5,
    criticalMolarVolume = 275.00e-6,
    acentricFactor = 0.589,
    dipoleMoment = 1.8,
    molarMass = 74.12e-3,
    hasDipoleMoment = true,
    hasIdealGasHeatCapacity=true,
    hasCriticalData = true,
    hasAcentricFactor = true);
```

---

## Modelica.Media.IdealGases.Common.SingleGasesData

### Ideal gas data based on the NASA Glenn coefficients

#### Information

This package contains ideal gas models for the 1241 ideal gases from

McBride B.J., Zehe M.J., and Gordon S. (2002): **NASA Glenn Coefficients for Calculating Thermodynamic Properties of Individual Species**. NASA report TP-2002-211556

## Package Content

Name	Description
Ag	
Agplus	
Agminus	
Air	
AL	
ALplus	
ALminus	
ALBr	
ALBr2	
ALBr3	
ALC	
ALC2	
ALCL	
ALCLplus	
ALCL2	
ALCL3	
ALF	
ALFplus	
ALFCL	
ALFCL2	
ALF2	
ALF2minus	
ALF2CL	
ALF3	
ALF4minus	
ALH	
ALHCL	
ALHCL2	
ALHF	
ALHFCL	
ALHF2	
ALH2	
ALH2CL	
ALH2F	
ALH3	
ALI	
ALI2	
ALI3	
ALN	
ALO	
ALOplus	
ALOminus	
ALOCL	

ALOC2	
ALOF	
ALOF2	
ALOF2minus	
ALOH	
ALOHCL	
ALOHCL2	
ALOHF	
ALOHF2	
ALO2	
ALO2minus	
AL_OH_2	
AL_OH_2CL	
AL_OH_2F	
AL_OH_3	
ALS	
ALS2	
AL2	
AL2Br6	
AL2C2	
AL2CL6	
AL2F6	
AL2I6	
AL2O	
AL2Oplus	
AL2O2	
AL2O2plus	
AL2O3	
AL2S	
AL2S2	
Ar	
Arplus	
B	
Bplus	
Bminus	
BBr	
BBr2	
BBr3	
BC	
BC2	
BCL	
BCLplus	
BCLOH	
BCL_OH_2	
BCL2	
BCL2plus	

BCL2OH	
BF	
BFCL	
BFCL2	
BFOH	
BF_OH_2	
BF2	
BF2plus	
BF2minus	
BF2CL	
BF2OH	
BF3	
BF4minus	
BH	
BHCL	
BHCL2	
BHF	
BHFCL	
BHF2	
BH2	
BH2CL	
BH2F	
BH3	
BH3NH3	
BH4	
BI	
BI2	
BI3	
BN	
BO	
BOminus	
BOCL	
BOCL2	
BOF	
BOF2	
BOH	
BO2	
BO2minus	
B_OH_2	
BS	
BS2	
B2	
B2C	
B2CL4	
B2F4	
B2H	

B2H2	
B2H3	
B2H3_db	
B2H4	
B2H4_db	
B2H5	
B2H5_db	
B2H6	
B2O	
B2O2	
B2O3	
B2_OH_4	
B2S	
B2S2	
B2S3	
B3H7_C2v	
B3H7_Cs	
B3H9	
B3N3H6	
B3O3CL3	
B3O3FCL2	
B3O3F2CL	
B3O3F3	
B4H4	
B4H10	
B4H12	
B5H9	
Ba	
Baplus	
BaBr	
BaBr2	
BaCL	
BaCLplus	
BaCL2	
BaF	
BaFplus	
BaF2	
BaH	
Bal	
Bal2	
BaO	
BaOplus	
BaOH	
BaOHplus	
Ba_OH_2	
BaS	

Ba2	
Be	
Beplus	
Beplusplus	
BeBr	
BeBr2	
BeCL	
BeCL2	
BeF	
BeF2	
BeH	
BeHplus	
BeH2	
Bel	
Bel2	
BeN	
BeO	
BeOH	
BeOHplus	
Be_OH_2	
BeS	
Be2	
Be2CL4	
Be2F4	
Be2O	
Be2OF2	
Be2O2	
Be3O3	
Be4O4	
Br	
Brplus	
Brminus	
BrCL	
BrF	
BrF3	
BrF5	
BrO	
OBrO	
BrOO	
BrO3	
Br2	
BrBrO	
BrOBr	
C	
Cplus	
Cminus	

CBr	
CBr2	
CBr3	
CBr4	
CCL	
CCL2	
CCL2Br2	
CCL3	
CCL3Br	
CCL4	
CF	
CFplus	
CFBr3	
CFCL	
CFCLBr2	
CFCL2	
CFCL2Br	
CFCL3	
CF2	
CF2plus	
CF2Br2	
CF2CL	
CF2CLBr	
CF2CL2	
CF3	
CF3plus	
CF3Br	
CF3CL	
CF4	
CHplus	
CHBr3	
CHCL	
CHCLBr2	
CHCL2	
CHCL2Br	
CHCL3	
CHF	
CHFBr2	
CHFCL	
CHFCLBr	
CHFCL2	
CHF2	
CHF2Br	
CHF2CL	
CHF3	
CHI3	

CH2	
CH2Br2	
CH2CL	
CH2CLBr	
CH2CL2	
CH2F	
CH2FBr	
CH2FCL	
CH2F2	
CH2I2	
CH3	
CH3Br	
CH3CL	
CH3F	
CH3I	
CH2OH	
CH2OHplus	
CH3O	
CH4	
CH3OH	
CH3OOH	
CI	
CI2	
CI3	
CI4	
CN	
CNplus	
CNminus	
CNN	
CO	
COplus	
COCL	
COCL2	
COFCL	
COF2	
COHCL	
COHF	
COS	
CO2	
CO2plus	
COOH	
CP	
CS	
CS2	
C2	
C2plus	

C2minus	
C2CL	
C2CL2	
C2CL3	
C2CL4	
C2CL6	
C2F	
C2FCL	
C2FCL3	
C2F2	
C2F2CL2	
C2F3	
C2F3CL	
C2F4	
C2F6	
C2H	
C2HCL	
C2HCL3	
C2HF	
C2HFCL2	
C2HF2CL	
C2HF3	
C2H2_vinylidene	
C2H2CL2	
C2H2FCL	
C2H2F2	
CH2CO_ketene	
O_CH_2O	
HO_CO_2OH	
C2H3_vinyl	
CH2BrminusCOOH	
C2H3CL	
CH2CLminusCOOH	
C2H3F	
CH3CN	
CH3CO_acetyl	
C2H4	
C2H4O_ethyleng_o	
CH3CHO_ethanal	
CH3COOH	
OHCH2COOH	
C2H5	
C2H5Br	
C2H6	
CH3N2CH3	
C2H5OH	

CH3OCH3	
CH3O2CH3	
CCN	
CNC	
OCCN	
C2N2	
C2O	
C3	
C3H3_1_propynl	
C3H3_2_propynl	
C3H4_allene	
C3H4_propyne	
C3H4_cyclo	
C3H5_allyl	
C3H6_propylene	
C3H6_cyclo	
C3H6O_propylox	
C3H6O_acetone	
C3H6O_propanal	
C3H7_n_propyl	
C3H7_i_propyl	
C3H8	
C3H8O_1propanol	
C3H8O_2propanol	
CNCOCN	
C3O2	
C4	
C4H2_butadiyne	
C4H4_1_3minuscyclo	
C4H6_butadiene	
C4H6_1butyne	
C4H6_2butyne	
C4H6_cyclo	
C4H8_1_butene	
C4H8_cis2_butene	
C4H8_isobutene	
C4H8_cyclo	
C4H9_n_butyl	
C4H9_i_butyl	
C4H9_s_butyl	
C4H9_t_butyl	
C4H10_n_butane	
C4H10_isobutane	
C4N2	
C5	
C5H6_1_3cyclo	

C5H8_cyclo	
C5H10_1_pentene	
C5H10_cyclo	
C5H11_pentyl	
C5H11_t_pentyl	
C5H12_n_pentane	
C5H12_i_pentane	
CH3C_CH3_2CH3	
C6D5_phenyl	
C6D6	
C6H2	
C6H5_phenyl	
C6H5O_phenoxy	
C6H6	
C6H5OH_phenol	
C6H10_cyclo	
C6H12_1_hexene	
C6H12_cyclo	
C6H13_n_hexyl	
C6H14_n_hexane	
C7H7_benzyl	
C7H8	
C7H8O_cresol_mx	
C7H14_1_heptene	
C7H15_n_heptyl	
C7H16_n_heptane	
C7H16_2_methylh	
C8H8_styrene	
C8H10_ethylbenz	
C8H16_1_octene	
C8H17_n_octyl	
C8H18_n_octane	
C8H18_isooctane	
C9H19_n_nonyl	
C10H8_naphthale	
C10H21_n_decyl	
C12H9_o_bipheny	
C12H10_biphenyl	
Ca	
Caplus	
CaBr	
CaBr2	
CaCL	
CaCLplus	
CaCL2	
CaF	

CaFplus	
CaF2	
CaH	
Cal	
Cal2	
CaO	
CaOplus	
CaOH	
CaOHplus	
Ca_OH_2	
CaS	
Ca2	
Cd	
Cdplus	
CL	
CLplus	
CLminus	
CLCN	
CLF	
CLF3	
CLF5	
CLO	
CLO2	
CL2	
CL2O	
Co	
Coplus	
Cominus	
Cr	
Crplus	
Crminus	
CrN	
CrO	
CrO2	
CrO3	
CrO3minus	
Cs	
Csplus	
Csminus	
CsBO2	
CsBr	
CsCL	
CsF	
CsH	
CsI	
CsLi	

CsNO2	
CsNO3	
CsNa	
CsO	
CsOH	
CsRb	
Cs2	
Cs2Br2	
Cs2CO3	
Cs2CL2	
Cs2F2	
Cs2I2	
Cs2O	
Cs2Oplus	
Cs2O2	
Cs2O2H2	
Cs2SO4	
Cu	
Cuplus	
Cuminus	
CuCL	
CuF	
CuF2	
CuO	
Cu2	
Cu3CL3	
D	
Dplus	
Dminus	
DBr	
DCL	
DF	
DOCL	
DO2	
DO2minus	
D2	
D2plus	
D2minus	
D2O	
D2O2	
D2S	
eminus	
F	
Fplus	
Fminus	
FCN	

FCO	
FO	
FO2_FOO	
FO2_OFO	
F2	
F2O	
F2O2	
FS2F	
Fe	
Feplus	
Fe_CO_5	
FeCL	
FeCL2	
FeCL3	
FeO	
Fe_OH_2	
Fe2CL4	
Fe2CL6	
Ga	
Gaplus	
GaBr	
GaBr2	
GaBr3	
GaCL	
GaCL2	
GaCL3	
GaF	
GaF2	
GaF3	
GaH	
Gal	
Gal2	
Gal3	
GaO	
GaOH	
Ga2Br2	
Ga2Br4	
Ga2Br6	
Ga2CL2	
Ga2CL4	
Ga2CL6	
Ga2F2	
Ga2F4	
Ga2F6	
Ga2I2	
Ga2I4	

Ga2I6	
Ga2O	
Ge	
Geplus	
Geminus	
GeBr	
GeBr2	
GeBr3	
GeBr4	
GeCL	
GeCL2	
GeCL3	
GeCL4	
GeF	
GeF2	
GeF3	
GeF4	
GeH4	
Gel	
GeO	
GeO2	
GeS	
GeS2	
Ge2	
H	
Hplus	
Hminus	
HALO	
HALO2	
HBO	
HBOplus	
HBO2	
HBS	
HBSplus	
HCN	
HCO	
HCOplus	
HCCN	
HCCO	
HCL	
HD	
HDplus	
HDO	
HDO2	
HF	
HI	

HNC	
HNCO	
HNO	
HNO2	
HNO3	
HOCL	
HOF	
HO2	
HO2minus	
HPO	
HSO3F	
H2	
H2plus	
H2minus	
HBOH	
HCOOH	
H2F2	
H2O	
H2Oplus	
H2O2	
H2S	
H2SO4	
H2BOH	
HB_OH_2	
H3BO3	
H3B3O3	
H3B3O6	
H3F3	
H3Oplus	
H4F4	
H5F5	
H6F6	
H7F7	
He	
Heplus	
Hg	
Hgplus	
HgBr2	
I	
Iplus	
Iminus	
IF5	
IF7	
I2	
In	
Inplus	

InBr	
InBr2	
InBr3	
InCL	
InCL2	
InCL3	
InF	
InF2	
InF3	
InH	
InI	
InI2	
InI3	
InO	
InOH	
In2Br2	
In2Br4	
In2Br6	
In2CL2	
In2CL4	
In2CL6	
In2F2	
In2F4	
In2F6	
In2I2	
In2I4	
In2I6	
In2O	
K	
Kplus	
Kminus	
KALF4	
KBO2	
KBr	
KCN	
KCL	
KF	
KH	
KI	
KLi	
KNO2	
KNO3	
KNa	
KO	
KOH	
K2	

K2plus	
K2Br2	
K2CO3	
K2C2N2	
K2CL2	
K2F2	
K2I2	
K2O	
K2Oplus	
K2O2	
K2O2H2	
K2SO4	
Kr	
Krplus	
Li	
Liplus	
Liminus	
LiALF4	
LiBO2	
LiBr	
LiCL	
LiF	
LiH	
LiI	
LiN	
LiNO2	
LiNO3	
LiO	
LiOF	
LiOH	
LiON	
Li2	
Li2plus	
Li2Br2	
Li2F2	
Li2I2	
Li2O	
Li2Oplus	
Li2O2	
Li2O2H2	
Li2SO4	
Li3plus	
Li3Br3	
Li3CL3	
Li3F3	
Li3I3	

Mg	
Mgplus	
MgBr	
MgBr2	
MgCL	
MgCLplus	
MgCL2	
MgF	
MgFplus	
MgF2	
MgF2plus	
MgH	
MgI	
MgI2	
MgN	
MgO	
MgOH	
MgOHplus	
Mg_OH_2	
MgS	
Mg2	
Mg2F4	
Mn	
Mnplus	
Mo	
Moplus	
Mominus	
MoO	
MoO2	
MoO3	
MoO3minus	
Mo2O6	
Mo3O9	
Mo4O12	
Mo5O15	
N	
Nplus	
Nminus	
NCO	
ND	
ND2	
ND3	
NF	
NF2	
NF3	
NH	

NHplus	
NHF	
NHF2	
NH2	
NH2F	
NH3	
NH2OH	
NH4plus	
NO	
NOCL	
NOF	
NOF3	
NO2	
NO2minus	
NO2CL	
NO2F	
NO3	
NO3minus	
NO3F	
N2	
N2plus	
N2minus	
NCN	
N2D2_cis	
N2F2	
N2F4	
N2H2	
NH2NO2	
N2H4	
N2O	
N2Oplus	
N2O3	
N2O4	
N2O5	
N3	
N3H	
Na	
Naplus	
Naminus	
NaALF4	
NaBO2	
NaBr	
NaCN	
NaCL	
NaF	
NaH	

---

**910 Modelica.Media.IdealGases.Common.SingleGasesData**

---

Nal	
NaLi	
NaNO2	
NaNO3	
NaO	
NaOH	
NaOHplus	
Na2	
Na2Br2	
Na2CL2	
Na2F2	
Na2I2	
Na2O	
Na2Oplus	
Na2O2	
Na2O2H2	
Na2SO4	
Na3CL3	
Na3F3	
Nb	
Nbplus	
Nbminus	
NbCL5	
NbO	
NbOCL3	
NbO2	
Ne	
Neplus	
Ni	
Niplus	
Niminus	
NiCL	
NiCL2	
NiO	
NiS	
O	
Oplus	
Ominus	
OD	
ODminus	
OH	
OHplus	
OHminus	
O2	
O2plus	
O2minus	

O3	
P	
Pplus	
Pminus	
PCL	
PCL2	
PCL2minus	
PCL3	
PCL5	
PF	
PFplus	
PFminus	
PFCL	
PFCLminus	
PFCL2	
PFCL4	
PF2	
PF2minus	
PF2CL	
PF2CL3	
PF3	
PF3CL2	
PF4CL	
PF5	
PH	
PH2	
PH2minus	
PH3	
PN	
PO	
POminus	
POCL3	
POFCL2	
POF2CL	
POF3	
PO2	
PO2minus	
PS	
P2	
P2O3	
P2O4	
P2O5	
P3	
P3O6	
P4	
P4O6	

P4O7	
P4O8	
P4O9	
P4O10	
Pb	
Pbplus	
Pbminus	
PbBr	
PbBr2	
PbBr3	
PbBr4	
PbCL	
PbCL2	
PbCL3	
PbCL4	
PbF	
PbF2	
PbF3	
PbF4	
Pbl	
Pbl2	
Pbl3	
Pbl4	
PbO	
PbO2	
PbS	
PbS2	
Rb	
Rbplus	
Rbminus	
RbBO2	
RbBr	
RbCL	
RbF	
RbH	
Rbl	
RbK	
RbLi	
RbNO2	
RbNO3	
RbNa	
RbO	
RbOH	
Rb2Br2	
Rb2CL2	
Rb2F2	

Rb2I2	
Rb2O	
Rb2O2	
Rb2O2H2	
Rb2SO4	
Rn	
Rnplus	
S	
Splus	
Sminus	
SCL	
SCL2	
SCL2plus	
SD	
SF	
SFplus	
SFminus	
SF2	
SF2plus	
SF2minus	
SF3	
SF3plus	
SF3minus	
SF4	
SF4plus	
SF4minus	
SF5	
SF5plus	
SF5minus	
SF6	
SF6minus	
SH	
SHminus	
SN	
SO	
SOminus	
SOF2	
SO2	
SO2minus	
SO2CL2	
SO2FCL	
SO2F2	
SO3	
S2	
S2minus	
S2CL2	

S2F2	
S2O	
S3	
S4	
S5	
S6	
S7	
S8	
Sc	
Scplus	
Scminus	
ScO	
ScOplus	
ScO2	
Sc2O	
Sc2O2	
Si	
Siplus	
Siminus	
SiBr	
SiBr2	
SiBr3	
SiBr4	
SiC	
SiC2	
SiCL	
SiCL2	
SiCL3	
SiCL4	
SiF	
SiFCL	
SiF2	
SiF3	
SiF4	
SiH	
SiHplus	
SiHBr3	
SiHCL	
SiHCL3	
SiHF	
SiHF3	
SiHI3	
SiH2	
SiH2Br2	
SiH2CL2	
SiH2F2	

SiH2I2	
SiH3	
SiH3Br	
SiH3CL	
SiH3F	
SiH3I	
SiH4	
Sil	
Sil2	
SiN	
SiO	
SiO2	
SiS	
SiS2	
Si2	
Si2C	
Si2F6	
Si2N	
Si3	
Sn	
Snplus	
Snminus	
SnBr	
SnBr2	
SnBr3	
SnBr4	
SnCL	
SnCL2	
SnCL3	
SnCL4	
SnF	
SnF2	
SnF3	
SnF4	
SnI	
SnI2	
SnI3	
SnI4	
SnO	
SnO2	
SnS	
SnS2	
Sn2	
Sr	
Srplus	
SrBr	

SrBr2	
SrCL	
SrCLplus	
SrCL2	
SrF	
SrFplus	
SrF2	
SrH	
SrI	
SrI2	
SrO	
SrOH	
SrOHplus	
Sr_OH_2	
SrS	
Sr2	
Ta	
Taplus	
Taminus	
TaCL5	
TaO	
TaO2	
Ti	
Tiplus	
Timinus	
TiCL	
TiCL2	
TiCL3	
TiCL4	
TiO	
TiOplus	
TiOCL	
TiOCL2	
TiO2	
U	
UF	
UFplus	
UFminus	
UF2	
UF2plus	
UF2minus	
UF3	
UF3plus	
UF3minus	
UF4	
UF4plus	

UF4minus	
UF5	
UF5plus	
UF5minus	
UF6	
UF6minus	
UO	
UOplus	
UOF	
UOF2	
UOF3	
UOF4	
UO2	
UO2plus	
UO2minus	
UO2F	
UO2F2	
UO3	
UO3minus	
V	
Vplus	
Vminus	
VCL4	
VN	
VO	
VO2	
V4O10	
W	
Wplus	
Wminus	
WCL6	
WO	
WOCL4	
WO2	
WO2CL2	
WO3	
WO3minus	
Xe	
Xeplus	
Zn	
Znplus	
Zr	
Zrplus	
Zrminus	
ZrN	
ZrO	

ZrOplus		
ZrO2		

## Types and constants

```
constant IdealGases.Common.DataRecord Ag(
  name="Ag",
  MM=0.1078682,
  Hf=2641186.188329832,
  H0=57453.70739476509,
  Tlimit=1000,
  allow={0,0,2.5,0,0,0,0},
  blow={33520.0237,6.56281935},
  ahigh={-330992.637,982.0086420000001,1.381179917,0.0006170899989999999,-
1.6881146e-007,
         2.008826848e-011,-5.627285655e-016},
  bhigh={27267.19171,14.56862733},
  R=77.07991789980736);

constant IdealGases.Common.DataRecord Agplus(
  name="Agplus",
  MM=0.1078676514,
  Hf=9475442.514362559,
  H0=57454.94519963192,
  Tlimit=1000,
  allow={3691132.75,-43169.8299999999,202.8445385,-0.465841374,0.000558620051,
         -3.154880975e-007,6.578702060000001e-011},
  blow={337157.447,-1142.924427},
  ahigh={-53274323.49999999,131071.0631,-109.8820208,0.0482600276,-
1.093661557e-005,
         1.26383591e-009,-5.852542535e-014},
  bhigh={-743912.2509999999,844.626618999999},
  R=77.08030991764042);

constant IdealGases.Common.DataRecord Agminus(
  name="Agminus",
  MM=0.1078687486,
  Hf=1419120.273357839,
  H0=57453.41519610434,
  Tlimit=1000,
  allow={0,0,2.5,0,0,0,0},
  blow={17665.65919,5.8696798},
  ahigh={0,0,2.5,0,0,0,0},
  bhigh={17665.65919,5.8696798},
  R=77.07952588596174);

constant IdealGases.Common.DataRecord Air(
  name="Air",
  MM=0.0289651159,
  Hf=-4333.833858403446,
  H0=298609.6803431054,
  Tlimit=1000,
  allow={10099.5016,-196.827561,5.00915511,-0.00576101373,1.06685993e-005,-
7.94029797e-009,
         2.18523191e-012},
  blow={-176.796731,-3.921504225},
  ahigh={241521.443,-1257.8746,5.14455867,-0.000213854179,7.06522784e-008,-
```

```

1.07148349e-011,
       6.57780015e-016},
bhigh={6462.26319,-8.147411905},
R=287.0512249529787);

constant IdealGases.Common.DataRecord AL(
  name="AL",
  MM=0.026981538,
  Hf=12230585.22460803,
  H0=256422.4100197698,
  Tlimit=1000,
  allow={5006.60889,18.61304407,2.412531111,0.0001987604647,-2.432362152e-007,
         1.538281506e-010,-3.944375734e-014},
  blow={38874.1268,6.086585765},
  ahighe={-29208.20938,116.7751876,2.356906505,7.73723152e-005,-1.529455262e-
008,
         -9.97167026e-013,5.053278264e-016},
  bhigh={38232.8865,6.600920155},
  R=308.1541163442944);

constant IdealGases.Common.DataRecord ALplus(
  name="ALplus",
  MM=0.0269809894,
  Hf=33839201.16732265,
  H0=229696.0985426279,
  Tlimit=1000,
  allow={0,0,2.5,0,0,0,0},
  blow={109064.4788,3.79100578},
  ahighe={-4181.18325,-9.94855727,2.548615878,-5.878760040000001e-005,
         3.132291294e-008,-7.74889463e-012,7.27444769e-016},
  bhigh={109101.1485,3.48866729},
  R=308.1603819910326);

constant IdealGases.Common.DataRecord ALminus(
  name="ALminus",
  MM=0.0269820866,
  Hf=10417656.61666804,
  H0=250396.2388142361,
  Tlimit=1000,
  allow={29108.01723,-383.698375,4.6551428,-0.00604582184,8.57725964e-006,-
5.47626e-009,
         1.322714061e-012},
  blow={34906.1698,-5.9570977},
  ahighe={633981.432,-2383.438463,5.46997113,-0.001299840355,2.88830547e-007,-
3.25324051e-011,
         1.472436088e-015},
  bhigh={47803.0654,-15.36906816},
  R=308.1478509523426);

constant IdealGases.Common.DataRecord ALBr(
  name="ALBr",
  MM=0.106885538,
  Hf=134022.0133429089,
  H0=89545.4631102666,
  Tlimit=1000,
  allow={8176.158640000001,-251.6942718,5.40329633,-0.00174721292,
         2.07730477e-006,-1.261267579e-009,3.16633597e-013},
  blow={-1.261267579e-009,3.16633597e-013});

```

```
blow={1635.024429,-2.323594886},
ahigh={-610339.5599999999,2010.066834,1.769617961,0.001929888914,-
6.64104783e-007,
1.172854627e-010,-7.17874276e-015},
bhigh={-12178.12867,22.04450764},
R=77.7885591968485);

constant IdealGases.Common.DataRecord ALBr2 (
  name="ALBr2",
  MM=0.186789538,
  Hf=-753051.4101919348,
  H0=71727.1167510463,
  Tlimit=1000,
  alow={31993.7587,-711.917897,9.478258110000001,-0.00487553167,
  5.51651299e-006,-3.34005304e-009,8.36847684e-013},
  blow={-15405.91306,-17.42171366},
  ahigh={-352378.29,467.154417,7.11190819,-0.00055517092,3.16630113e-007,-
5.52102833e-011,
  3.17672595e-015},
  bhigh={-22650.04078,-2.69561036},
  R=44.51251440003026);

constant IdealGases.Common.DataRecord ALBr3 (
  name="ALBr3",
  MM=0.266693538,
  Hf=-1539132.395476339,
  H0=67280.73028901059,
  Tlimit=1000,
  alow={47189.4884,-1053.853163,13.50747168,-0.006639869929999999,
  7.26927683e-006,-4.27804949e-009,1.045573281e-012},
  blow={-46994.4033,-36.67936779999999},
  ahigh={-89355.2744,-26.78035134,10.02063997,-8.458764399999999e-006,
  1.904243317e-009,-2.215195785e-013,1.038219389e-017},
  bhigh={-52492.4764,-15.79666403},
  R=31.17612845947546);

constant IdealGases.Common.DataRecord ALC (
  name="ALC",
  MM=0.038992238,
  Hf=17497931.48574852,
  H0=232305.4398672885,
  Tlimit=1000,
  alow={41399.2188,-585.867789,5.96244572,-0.002006064322,1.665566136e-006,-
7.12164921e-010,
  1.271188744e-013},
  blow={83834.3611,-8.00216505},
  ahigh={1937001.487,-6749.117789999999,13.47525643,-0.00585080265,
  1.926461091e-006,-2.585922351e-010,1.222659806e-014},
  bhigh={122536.3649,-61.52244872},
  R=213.2340287828567);

constant IdealGases.Common.DataRecord ALC2 (
  name="ALC2",
  MM=0.051002938,
  Hf=13246605.04851701,
  H0=240432.7374238716,
  Tlimit=1000,
```

```

alow={15240.87465,-264.9987059,6.2804149,-0.0001786037647,3.79190774e-006,-
3.99945007e-009,
      1.299752388e-012},
blow={80927.63279999999,-6.247291862},
ahigh={146601.7895,-1362.764489,8.457759149999999,-0.000367890168,
      7.9031745e-008,-8.880023720000001e-012,4.05363395e-016},
bhigh={87059.24230000001,-21.27956728},
R=163.019471544953};

constant IdealGases.Common.DataRecord ALCL(
  name="ALCL",
  MM=0.06243453800000001,
  Hf=-816974.7808496637,
  H0=149326.1950620985,
  Tlimit=1000,
  alow={23808.81392,-445.715942,5.99285399,-0.002777300261,3.101582418e-006,-
1.815268376e-009,
      4.414662339999999e-013},
  blow={-5202.73603,-7.383298016},
  ahigh={-899346.1220000001,2765.942061,1.003500696,0.002267806833,-
7.2805416e-007,
      1.182380673e-010,-6.66049813e-015},
  bhigh={-24961.64448,26.1692548},
  R=133.1710342759323);

constant IdealGases.Common.DataRecord ALCLplus(
  name="ALCLplus",
  MM=0.0624339894,
  Hf=13804168.45507553,
  H0=146625.821735492,
  Tlimit=1000,
  alow={34699.5522,-547.990003,6.10601113,-0.002674711519,2.696426573e-006,-
1.463885068e-009,
      3.35872906e-013},
  blow={105171.951,-7.816586372},
  ahigh={-754724.0819999999,1251.545253,4.44242491,-0.0009786116429999999,
      6.347511329999999e-007,-1.158320344e-010,6.870022460000001e-015},
  bhigh={93183.32699999999,4.047521108},
  R=133.1722044338881);

constant IdealGases.Common.DataRecord ALCL2(
  name="ALCL2",
  MM=0.09788753800000001,
  Hf=-2460724.387613059,
  H0=131241.0063883719,
  Tlimit=1000,
  alow={53405.4595,-967.805798,10.06252671,-0.00553952845,5.82342056e-006,-
3.30654245e-009,
      7.8315621e-013},
  blow={-26076.2711,-23.93364216},
  ahigh={430345.306,-1552.370585,8.760657419999999,-0.0009467450059999999,
      2.40184488e-007,-2.427836709e-011,8.390123470000001e-016},
  bhigh={-21458.84709,-18.03641668},
  R=84.93902461823076);

constant IdealGases.Common.DataRecord ALCL3 (
  name="ALCL3",

```

```
MM=0.133340538,
Hf=-4384854.5368851,
H0=122999.3762287055,
Tlimit=1000,
alow={77506.0097,-1440.779717,14.01744141,-0.00638163124,5.87167472e-006,-
2.908872278e-009,
      5.99405089999999e-013},
blow={-65793.43180000001,-44.94017799},
ahigh={-137863.0916,-55.7920729,10.04190387,-1.682165339e-005,
      3.72466466e-009,-4.27552678e-013,1.982341329e-017},
bhigh={-73434.0747,-20.45130429},
R=62.35517063835456);

constant IdealGases.Common.DataRecord ALF(
  name="ALF",
  MM=0.0459799412,
  Hf=-5742948.753488184,
  H0=193391.0041624847,
  Tlimit=1000,
  alow={30207.71686,-308.0986949,3.88641314,0.00356434369,-5.85847128e-006,
        4.38810992e-009,-1.256906273e-012},
  blow={-31175.72862,2.032567172},
  ahigh={-711122.905,1903.013316,2.191607179,0.001421471467,-4.17963477e-007,
        6.1538044e-011,-2.945403999e-015},
  bhigh={-45407.2896,16.13163743},
  R=180.8282434254179);

constant IdealGases.Common.DataRecord ALFplus(
  name="ALFplus",
  MM=0.0459793926,
  Hf=15055300.25205248,
  H0=191666.0160491116,
  Tlimit=1000,
  alow={36430.3812,-252.4634478,3.105310486,0.00573327043,-8.839786990000001e-
006,
        6.34357959e-009,-1.645011685e-012},
  blow={83702.3744999999,6.76401558},
  ahigh={1505101.126,-2236.668821,3.098216724,0.00311541393,-1.168427068e-006,
        1.773587168e-010,-9.66584137999999e-015},
  bhigh={98850.8385,5.99189459},
  R=180.8304009653229);

constant IdealGases.Common.DataRecord ALFCL(
  name="ALFCL",
  MM=0.0814329412,
  Hf=-5359132.392481975,
  H0=148996.1534141419,
  Tlimit=1000,
  alow={41047.7948,-576.964642,6.5626427,0.00410628584,-7.21392112e-006,
        5.48640145e-009,-1.580656193e-012},
  blow={-51148.287,-6.041227053},
  ahigh={69706.36350000001,-622.4485910000001,7.57347541,-0.0002236155999,
        2.382317949e-008,4.44378576e-012,-5.48697784999999e-016},
  bhigh={-51009.4319999999,-10.91164505},
  R=102.1020716859482);

constant IdealGases.Common.DataRecord ALFCL2(
```

```

name="ALFCL2",
MM=0.1168859412,
Hf=-6770663.741722942,
H0=134646.3042383407,
Tlimit=1000,
alow={69556.07340000001,-1188.857902,11.5667259,0.000457495949,-3.40737283e-
006,
      3.35682254e-009,-1.085329228e-012},
blow={-91620.4926,-32.06065081},
ahigh={-172036.0526,-119.3340759,10.08894431,-3.55099381e-005,
      7.83095261e-009,-8.961734229999999e-013,4.14533026e-017},
bhigh={-98041.8547,-21.27046859},
R=71.13320827671959);

constant IdealGases.Common.DataRecord ALF2 (
  name="ALF2",
  MM=0.0649783444,
  Hf=-9722688.08683282,
  H0=178535.3583123919,
  Tlimit=1000,
  alow={29949.51991,-219.4199915,3.41674876,0.01272953936,-1.883312206e-005,
        1.330702256e-008,-3.68011863e-012},
  blow={-76075.3765,8.759626916},
  ahigh={-214689.6245,30.9684015,6.81639962,0.0001820129379,-7.79367744e-008,
        1.472534201e-011,-8.80369529999999e-016},
  bhigh={-78840.02680000001,-7.915370474},
  R=127.9575845887511);

constant IdealGases.Common.DataRecord ALF2minus (
  name="ALF2minus",
  MM=0.064978893,
  Hf=-13130889.82602397,
  H0=174392.244570864,
  Tlimit=1000,
  alow={123336.9037,-1348.195504,8.61117805,0.0002245141297,-2.424673201e-006,
        2.21413243e-009,-6.60271396e-013},
  blow={-97084.37460000001,-21.92617549},
  ahigh={-159007.6965,-206.0486516,7.15528938,-6.260167690000001e-005,
        1.391935746e-008,-1.603901265e-012,7.461508519999999e-017},
  bhigh={-104047.9429,-11.22696648},
  R=127.9565042759347);

constant IdealGases.Common.DataRecord ALF2CL (
  name="ALF2CL",
  MM=0.1004313444,
  Hf=-9948369.714345874,
  H0=147638.3402869194,
  Tlimit=1000,
  alow={56774.12089999999,-886.9829470000001,8.75274655,0.0081957912,-
1.380480934e-005,
        1.032703706e-008,-2.948968416e-012},
  blow={-117793.6778,-18.5834736},
  ahigh={-211542.2988,-195.9188681,10.1457685,-5.81321935e-005,
        1.281071982e-008,-1.465366277e-012,6.77600245e-017},
  bhigh={-122715.311,-23.59853398},
  R=82.78762023621782);

```

```
constant IdealGases.Common.DataRecord ALF3 (
  name="ALF3",
  MM=0.0839767476,
  Hf=-14400140.18832994,
  H0=167233.459277244,
  Tlimit=1000,
  alow={44102.6352,-566.7495739999999,5.83307285,0.01603535069,-2.413263602e-005,
         1.71410616e-008,-4.7475432e-012},
  blow={-144334.9991,-5.461613658},
  ahighelement1={-249397.216,-291.1202519,10.21662932,-8.64298597e-005,
                 1.905713108e-008,-2.181058411e-012,1.009055287e-016},
  bhighelement1={-147569.4276,-27.03983244},
  R=99.00921668940654);

constant IdealGases.Common.DataRecord ALF4minus (
  name="ALF4minus",
  MM=0.1029756994,
  Hf=-18952051.96343634,
  H0=163017.4409866645,
  Tlimit=1000,
  alow={231201.0037,-3235.37678,20.17865547,-0.0091081935,6.67955263e-006,-2.606075106e-009,
         4.11006051e-013},
  blow={-221178.4563,-86.77268282},
  ahighelement1={-321741.161,-261.2097043,13.19792654,-8.01013541e-005,
                 1.786204421e-008,-2.062751964e-012,9.61258177e-017},
  bhighelement1={-238157.7915,-42.31634321999999},
  R=80.74207845584199);

constant IdealGases.Common.DataRecord ALH (
  name="ALH",
  MM=0.027989478,
  Hf=8905160.860806337,
  H0=309691.5205063846,
  Tlimit=1000,
  alow={-37591.1403,508.900223,1.128086896,0.003988660910000001,-2.150790303e-007,
         -2.176790819e-009,1.020805902e-012},
  blow={26444.31827,16.50021856},
  ahighelement1={6802018.430000001,-21784.16933,30.32713047,-0.01503343597,
                 4.49214236e-006,-6.17845037e-010,3.11520526e-014},
  bhighelement1={165830.1221,-187.6766425},
  R=297.0570583702919);

constant IdealGases.Common.DataRecord ALHCL (
  name="ALHCL",
  MM=0.063442478,
  Hf=165850.4890051741,
  H0=171597.5217739761,
  Tlimit=1000,
  alow={35871.6976,-591.624999,6.94454238,-0.002511543148,6.47133255e-006,-5.64227714e-009,
         1.715950436e-012},
  blow={2750.930479,-9.903745880000001},
  ahighelement1={-168391.533,-886.230179,8.062379399999999,-0.0009200551470000001,
                 3.94981722e-007,-6.40108676e-011,3.5783541e-015},
  bhighelement1={3641.50485,-18.00686894},
```

```

R=131.0552844420737);

constant IdealGases.Common.DataRecord ALHCL2 (
  name="ALHCL2",
  MM=0.098895478,
  Hf=-3552022.520180347,
  H0=138289.3968114498,
  Tlimit=1000,
  allow={118482.8615,-2054.157118,15.74696513,-0.01476257296,2.065336978e-005,
         -1.42019885e-008,3.82627986e-012},
  blow={-34342.7746,-57.61184061},
  ahigh={96080.63939999999,-1485.540161,11.03592709,-0.00039569552,
         8.46584373e-008,-9.48333652e-012,4.31903353e-016},
  bhigh={-36692.1639,-32.36862171},
  R=84.07332840840307);

constant IdealGases.Common.DataRecord ALHF (
  name="ALHF",
  MM=0.0469878812,
  Hf=-3886416.653322091,
  H0=224996.7593771817,
  Tlimit=1000,
  allow={-9289.545099999999,133.5515868,2.77244868,0.00675141503,-4.13384614e-
006,
         5.86570407e-010,2.276158011e-013},
  blow={-23846.85845,12.28752581},
  ahigh={734216.9250000001,-3462.16413,10.39714588,-0.001750737965,
         4.615687860000001e-007,-5.39041474e-011,2.340478515e-015},
  bhigh={-2965.279722,-37.13172313},
  R=176.9492853829723);

constant IdealGases.Common.DataRecord ALHFCL (
  name="ALHFCL",
  MM=0.08244088120000001,
  Hf=-6735062.943504781,
  H0=152019.6754034696,
  Tlimit=1000,
  allow={129037.3237,-2142.099873,14.8543442,-0.0114032693,1.564521768e-005,-
1.067574855e-008,
         2.857255247e-012},
  blow={-58183.0307,-54.83424986},
  ahigh={40689.8713,-1502.129854,11.05214126,-0.000403158773,8.64525222e-008,
         -9.70065952e-012,4.42362950999999e-016},
  bhigh={-61298.8811,-34.03564016},
  R=100.8537497292059);

constant IdealGases.Common.DataRecord ALHF2 (
  name="ALHF2",
  MM=0.0659862844,
  Hf=-11597852.32580848,
  H0=186401.9941695641,
  Tlimit=1000,
  allow={105018.1431,-1398.654202,9.16633483999999,0.00386728687,-4.83084109e-
006,
         3.099287181e-009,-8.440702990000001e-013},
  blow={-86590.48800000001,-25.57299683},
  ahigh={-4419.9473,-1612.826324,11.13336052,-0.000435181266,9.34439616e-008,
         1.13336052,-0.000435181266,9.34439616e-008});

```

```
-1.049431425e-011, 4.78833398e-016},  
bhigh={-86069.1618999999, -36.76383785},  
R=126.0030334425073);  
  
constant IdealGases.Common.DataRecord ALH2 (  
  name="ALH2",  
  MM=0.028997418,  
  Hf=9544813.058873035,  
  H0=347990.7762822194,  
  Tlimit=1000,  
  allow={14551.82996, -215.3768996, 5.14437023, -0.00396522203, 1.340900203e-005, -  
1.216744854e-008,  
  3.74310713e-012},  
  blow={33110.3794, -3.608799711},  
  ahighe={143291.0601, -2365.907684, 9.085667190000001, -0.001308536612,  
  4.77719136e-007, -7.32472189e-011, 3.99790094e-015},  
  bhigh={44859.5185, -32.20814285},  
  R=286.7314600217164);  
  
constant IdealGases.Common.DataRecord ALH2CL (  
  name="ALH2CL",  
  MM=0.064450418,  
  Hf=-1650034.387674569,  
  H0=174603.2275539315,  
  Tlimit=1000,  
  allow={93038.2898, -1328.188016, 9.42476359, -0.00327766059, 1.004231609e-005, -  
8.94812309e-009,  
  2.708031665e-012},  
  blow={-7647.43793, -26.82512215},  
  ahighe={316692.993, -3094.037914, 12.1587202, -0.0008248269589999999,  
  1.765027123e-007, -1.977372723e-011, 9.00617628e-016},  
  bhigh={2411.845642, -47.18390855},  
  R=129.0057110257997);  
  
constant IdealGases.Common.DataRecord ALH2F (  
  name="ALH2F",  
  MM=0.0479958212,  
  Hf=-6597571.331897535,  
  H0=224286.2134839355,  
  Tlimit=1000,  
  allow={89060.39290000001, -975.080393, 5.87077232, 0.00689666369, -4.04581728e-  
006,  
  7.21931866e-010, 7.329846739999999e-014},  
  blow={-34253.02600000001, -9.387539141},  
  ahighe={271151.4962, -3169.62348, 12.21401118, -0.000846560026, 1.812341995e-007,  
  -2.030947985e-011, 9.251825820000001e-016},  
  bhigh={-22592.17746, -49.41337333},  
  R=173.2332480645211);  
  
constant IdealGases.Common.DataRecord ALH3 (  
  name="ALH3",  
  MM=0.030005358,  
  Hf=4295768.77569666,  
  H0=346957.3667476322,  
  Tlimit=1000,  
  allow={14812.07909, -28.3649534, 2.507597126, 0.00731592051, 2.331766687e-006, -  
6.38920989e-009,
```

```

    2.456885069e-012},
blow={14631.89428,8.3131877},
ahigh={588545.855,-4595.78566,13.20893344,-0.001226849004,2.626580824e-007,
-2.943725869e-011,1.341177648e-015},
bhigh={39917.1695,-61.81829295},
R=277.0995766822712);

constant IdealGases.Common.DataRecord ALI(
  name="ALI",
  MM=0.153886008,
  Hf=437954.1575995656,
  H0=63365.84545100423,
  Tlimit=1000,
  alow={1870.854131,-154.3184003,5.07439188,-0.00112058483,1.386302156e-006,-
8.54078228e-010,
2.180358492e-013},
blow={7517.44369,0.6688267621999999},
ahigh={475281.792,-1198.028588,5.43368536,-0.0001004479306,-1.005928846e-
007,
4.568173190000001e-011,-3.960201360000001e-015},
bhigh={14646.27816,-3.141199746},
R=54.03007140194319);

constant IdealGases.Common.DataRecord ALI2(
  name="ALI2",
  MM=0.280790478,
  Hf=-120420.5863419628,
  H0=49585.17859711753,
  Tlimit=1000,
  alow={14234.13595,-464.657569,8.72267491,-0.00356051929,4.18868229e-006,-
2.616026962e-009,
6.72019551e-013},
blow={-3846.13333,-10.73076578},
ahigh={-335599.902,474.334794,7.10650139,-0.0005529965340000001,
3.161480314e-007,-5.51548916e-011,3.17415582e-015},
bhigh={-9785.48885,-0.5602275943999999},
R=29.61094713475291);

constant IdealGases.Common.DataRecord ALI3(
  name="ALI3",
  MM=0.407694948,
  Hf=-469298.0227952199,
  H0=46612.22095889204,
  Tlimit=1000,
  alow={27060.32439,-755.993681,12.71396349,-0.00546688409,6.29761892e-006,-
3.86420961e-009,
9.77413244e-013},
blow={-22209.66921,-28.47883861},
ahigh={-63574.2747,-15.7877228,10.01249733,-5.22867312e-006,1.19617352e-009,
-1.40919449e-013,6.67127639e-018},
bhigh={-26114.55525,-12.47999813},
R=20.39385584930034);

constant IdealGases.Common.DataRecord ALN(
  name="ALN",
  MM=0.040988238,
  Hf=10706217.98868251,

```

---

**928 Modelica.Media.IdealGases.Common.SingleGasesData**

---

```
H0=226043.1638949691,
Tlimit=1000,
alow={27166.45079,-254.349111,3.53898772,0.00529625616,-9.845700609999999e-
006,
     8.45220904e-009,-2.592304568e-012},
blow={53099.94910000001,5.39943522},
ahigh={3821214.22,-10677.76595,14.49174222,-0.00372722713,
       7.922770960000001e-007,-8.086892210000001e-011,2.893539736e-015},
bhigh={120523.634,-72.88585119000001},
R=202.8501932676394);

constant IdealGases.Common.DataRecord ALO (
  name="ALO",
  MM=0.042980938,
  Hf=1566252.602490899,
  H0=204465.1980373253,
  Tlimit=1000,
  alow={-7683.3911,295.7969549,0.480810844,0.01169224855,-1.595428871e-005,
        1.060766814e-008,-2.647888708e-012},
  blow={5843.67217,21.60997839},
  ahigh={15657.21161,3855.74101,-5.92607978,0.009050960419999999,-
2.930661549e-006,
        4.238529070000001e-010,-2.280655341e-014},
  bhigh={-13316.94655,68.30663436},
  R=193.4455688240215);

constant IdealGases.Common.DataRecord ALOplus (
  name="ALOplus",
  MM=0.0429803894,
  Hf=23103397.01110293,
  H0=211485.1942220886,
  Tlimit=1000,
  alow={28291.78513,-417.90871,5.31329987,-0.001410883046,2.526524021e-006,-
2.005474816e-009,
        5.76255355e-013},
  blow={120364.8109,-3.41056593},
  ahigh={27108.46446,-699.8653420000001,5.78781584,-0.000687189546,
        2.075047303e-007,-2.655281004e-011,1.28312966e-015},
  bhigh={121868.2226,-7.05103588},
  R=193.4480379556543);

constant IdealGases.Common.DataRecord ALOminus (
  name="ALOminus",
  MM=0.04298148659999999,
  Hf=-6349752.034867962,
  H0=203453.1304461675,
  Tlimit=1000,
  alow={20780.75267,2.859351226,1.90191237,0.00730104566,-9.13976987e-006,
        5.65298319e-009,-1.396787103e-012},
  blow={-33592.6773,12.96339858},
  ahigh={-69719.4829,-232.0553036,4.67278433,-4.26644704e-005,
        1.632506336e-008,-1.727134051e-012,8.136840640000001e-017},
  bhigh={-33077.1893,-2.155922595},
  R=193.4430997554189);

constant IdealGases.Common.DataRecord ALOCL (
  name="ALOCL",
```

```

MM=0.078433938,
Hf=-3844822.505278264,
H0=151797.0065458144,
Tlimit=1000,
alow={-5144.627790000001,-58.86297070000001,4.45231693,0.009288235090000001,
      -1.246894823e-005,8.17807354e-009,-2.130739934e-012},
blow={-37596.8365,2.476862818},
ahigh={-125802.7706,-242.0565961,7.68113002,-7.268741699999999e-005,
      1.611337534e-008,1.852889749e-012,8.60727822e-017},
bhigh={-37543.2605,-14.7581832},
R=106.0060505950881);

constant IdealGases.Common.DataRecord ALOCL2 (
  name="ALOCL2",
  MM=0.113886938,
  Hf=-3532526.873274967,
  H0=138316.950799046,
  Tlimit=1000,
  alow={69026.3713,-1189.2975,11.61842739,0.0002818279805,-3.145010615e-006,
        3.16838065e-009,-1.032375462e-012},
  blow={-44833.2307,-31.65475132},
  ahigh={-170771.3922,-118.0013804,10.08798517,-3.51385804e-005,
        7.751187029999999e-009,-8.8724706e-013,4.10481544e-017},
  bhigh={-51249.4414,-20.5925296},
  R=73.00637058132162);

constant IdealGases.Common.DataRecord ALOF (
  name="ALOF",
  MM=0.0619793412,
  Hf=-9233554.147555217,
  H0=177526.9434454718,
  Tlimit=1000,
  alow={7030.57836,-115.457543,3.37882359,0.01283929158,-1.749748388e-005,
        1.162605473e-008,-3.064218613e-012},
  blow={-69593.817,5.994804586},
  ahigh={-164102.0297,-327.777029,7.74451974,-9.78864944e-005,
        2.165743208e-008,-2.486541919e-012,1.153633944e-016},
  bhigh={-69740.85090000001,-17.17253748},
  R=134.1490864378533);

constant IdealGases.Common.DataRecord ALOF2 (
  name="ALOF2",
  MM=0.08097774440000001,
  Hf=-9553856.763142934,
  H0=173570.9867463288,
  Tlimit=1000,
  alow={47560.8529,-612.431836,6.05022366,0.01559291479,-2.366136797e-005,
        1.688366194e-008,-4.690583099999999e-012},
  blow={-91718.2749,-4.956258224},
  ahigh={-249668.1413,-283.7821503,10.21108125,-8.418303430000001e-005,
        1.855520828e-008,-2.122974022e-012,9.8193191e-017},
  bhigh={-95218.0419,-25.22534812},
  R=102.6760137813866);

constant IdealGases.Common.DataRecord ALOF2minus (
  name="ALOF2minus",
  MM=0.08097829299999999,

```

---

**930 Modelica.Media.IdealGases.Common.SingleGasesData**

---

```
Hf=-12006793.21555963,
H0=173836.4749180376,
Tlimit=1000,
alow={154060.5112,-1928.054597,12.55806423,-0.000490403537,-2.341783347e-
006,
      2.4151386e-009,-7.47180925e-013},
blow={-109143.0174,-42.6582175},
ahigh={-238988.1459,-288.0143725,10.2170024,-8.747061999999999e-005,
      1.944846334e-008,-2.241045051e-012,1.042591053e-016},
bhigh={-119057.6566,-26.00779629},
R=102.6753181868134);

constant IdealGases.Common.DataRecord ALOH(
  name="ALOH",
  MM=0.043988878,
  Hf=-4382066.098617018,
  H0=235330.6442596694,
  Tlimit=1000,
  alow={58764.9318,-944.942269,7.82059918,0.000585888847,-4.08366681e-006,
        4.587229340000001e-009,-1.563936726e-012},
  blow={-19932.83011,-20.65043885},
  ahigh={788206.811,-2263.671626,7.82395488,0.0001821171456,-8.26372932e-008,
        1.265414876e-011,-6.87597253e-016},
  bhigh={-10398.08093,-22.09032458},
  R=189.0130500714294);

constant IdealGases.Common.DataRecord ALOHCL (
  name="ALOHCL",
  MM=0.07944187800000001,
  Hf=-4705147.327962211,
  H0=175208.823739036,
  Tlimit=1000,
  alow={16363.60341,-191.5959416,4.71117047,0.01388068475,-1.999018454e-005,
        1.43964735e-008,-4.03333045e-012},
  blow={-45680.8371999999,4.644207506},
  ahigh={796711.551,-2843.470502,10.9555997,-0.0001099328162,-1.344506283e-
008,
        4.41589377e-012,-2.973546112e-016},
  bhigh={-29698.61461,-32.96113996},
  R=104.661070575396);

constant IdealGases.Common.DataRecord ALOHCL2 (
  name="ALOHCL2",
  MM=0.114894878,
  Hf=-6311381.504752545,
  H0=149265.1917868784,
  Tlimit=1000,
  alow={39719.2986,-775.155496,9.49405153,0.01074401359,-1.676464791e-005,
        1.259706368e-008,-3.61317798e-012},
  blow={-85847.9693,-19.93481642},
  ahigh={738592.976,-2863.204228,13.97062327,-0.0001160311527,-1.208248796e-
008,
        4.25832763e-012,-2.900049181e-016},
  bhigh={-72930.48,-46.81834857},
  R=72.36590651151568);

constant IdealGases.Common.DataRecord ALOHF (
```

```

name="ALOHF",
MM=0.0629872812,
Hf=-9116309.897179686,
H0=211393.1693244763,
Tlimit=1000,
alow={-1556.003336,288.6581999,0.7703793290000001,0.02450406329,-
3.41781101e-005,
2.388932902e-008,-6.570480510000001e-012},
blow={-71772.22559999999,24.53071801},
ahigh={755711.3130000001,-2936.702803,11.02455104,-0.000137307803,-
7.432816159999999e-009,
3.72998122e-012,-2.657027781e-016},
bhigh={-53401.309,-35.27397302},
R=132.0023954296348);

constant IdealGases.Common.DataRecord ALOHF2 (
  name="ALOHF2",
  MM=0.0819856844,
  Hf=-13923293.06456335,
  H0=188185.2681100507,
  Tlimit=1000,
  alow={17382.63538,-130.7687299,3.31238192,0.02784734099,-3.98236321e-005,
  2.809344456e-008,-7.7645742e-012},
  blow={-138413.3568,10.04739945},
  ahigh={653011.992,-3034.017232,14.09719788,-0.0001663681799,-1.011564183e-
009,
  2.993805003e-012,-2.315956955e-016},
  bhigh={-122305.2213,-51.5121069},
  R=101.4137048540635);

constant IdealGases.Common.DataRecord ALO2 (
  name="ALO2",
  MM=0.058980338,
  Hf=-655436.5795597849,
  H0=226543.7848118131,
  Tlimit=1000,
  alow={43384.8045,-473.5292259999999,6.00171767,0.007094420880000001,-
1.129107996e-005,
  8.252691679999999e-009,-2.327652976e-012},
  blow={-3826.1458,-4.83002248},
  ahigh={118721.6642,-833.56254,8.309301189999999,-0.000353866722,
  5.96706946e-008,4.0148977e-014,-3.51570252e-016},
  bhigh={-2033.107586,-17.15063884},
  R=140.9702331648218);

constant IdealGases.Common.DataRecord ALO2minus (
  name="ALO2minus",
  MM=0.0589808866,
  Hf=-7673201.423187831,
  H0=180476.720741597,
  Tlimit=1000,
  alow={117867.8641,-1507.186304,9.52474975,-0.000520798902,-1.586902345e-006,
  1.700455028e-009,-5.30141977e-013},
  blow={-48254.6927,-30.81215859},
  ahigh={-187241.0758,-233.8853263,7.67607002,-7.093570299999999e-005,
  1.576717569e-008,-1.816494162e-012,8.449707959999999e-017},
  bhigh={-55946.40560000001,-17.73751567},
  R=140.9689219558121);

```

```
constant IdealGases.Common.DataRecord AL_OH_2(
  name="AL_OH_2",
  MM=0.060996218,
  Hf=-8322822.080542764,
  H0=229973.8485425441,
  Tlimit=1000,
  allow={4397.31691,132.7680339,1.093947024,0.03054059173,-4.37594335e-005,
         3.17461743e-008,-9.004259400000001e-012},
  blow={-63154.3739,21.01977655},
  ahighe={1669643.735,-5924.73828,15.49480264,-0.000538348808,
          5.413044529999999e-008,-1.196022328e-012,-1.082001733e-016},
  bhigh={-26975.25061,-66.17911543},
  R=136.3112709709314);

constant IdealGases.Common.DataRecord AL_OH_2CL(
  name="AL_OH_2CL",
  MM=0.096449218,
  Hf=-8906829.332716828,
  H0=178560.6701342047,
  Tlimit=1000,
  allow={24778.79709,-445.457721,5.9284702,0.02730162734,-4.04140537e-005,
         2.97421041e-008,-8.477150599999999e-012},
  blow={-103377.803,-3.679940779},
  ahighe={1603911.487,-5767.23224,18.17772266,-0.000369181615,1.137918702e-008,
          4.0830794e-012,-3.64132343e-016},
  bhigh={-71186.9985,-77.57541774000001},
  R=86.20569634893255);

constant IdealGases.Common.DataRecord AL_OH_2F(
  name="AL_OH_2F",
  MM=0.0799946211999999,
  Hf=-13371257.86652266,
  H0=205914.4821602081,
  Tlimit=1000,
  allow={21328.12662,-193.2900224,3.069174664,0.0355501733,-5.18115833e-005,
         3.75352028e-008,-1.059220701e-011},
  blow={-129579.7992,10.01897011},
  ahighe={1559238.236,-5841.27303,18.23229919,-0.000390780542,1.610928938e-008,
          3.54478406e-012,-3.39345388e-016},
  bhigh={-96231.0089,-79.71351802},
  R=103.9378882639174);

constant IdealGases.Common.DataRecord AL_OH_3(
  name="AL_OH_3",
  MM=0.078003558,
  Hf=-12982325.01137961,
  H0=225563.6467249353,
  Tlimit=1000,
  allow={-14024.75452,369.607346,-0.297960065,0.0492985935,-6.98157228e-005,
         5.0148017e-008,-1.412480165e-011},
  blow={-125526.076,27.11490074},
  ahighe={2477063.616,-8968.61724999999,22.80320998,-0.000832029224999999,
          8.66349443e-008,-2.420287082e-012,-1.331746442e-016},
  bhigh={-70152.7864,-112.1899198},
  R=106.5909326854039);

constant IdealGases.Common.DataRecord ALS(
```

```

name="ALS",
MM=0.059046538,
Hf=3940654.556241722,
H0=153914.3412607866,
Tlimit=1000,
alow={26117.42801,-347.067389,4.43620459,0.00334016821,-8.41019967e-006,
      8.50818511e-009,-2.780868507e-012},
blow={28637.79062,0.7522705491},
ahigh={8909844.290000001,-25076.77454,28.93456134,-0.00983976682,
       2.045319809e-006,-2.08797089e-010,8.15279252000001e-015},
bhigh={188576.1615,-178.6036457},
R=140.8121844501705);

constant IdealGases.Common.DataRecord ALS2 (
  name="ALS2",
  MM=0.09111153799999999,
  Hf=2727813.452122826,
  H0=158363.90556814,
  Tlimit=1000,
  alow={42523.3541,-776.46033,9.80106896,-0.00391068009,3.87569597e-006,-
2.082025593e-009,
      4.68293589e-013},
  blow={31679.6154,-21.78097896},
  ahigh={-68525.6354,-26.71667739,7.52019386,-8.148565100000001e-006,
         1.811865152e-009,-2.086958125e-013,9.70324637999999e-018},
  bhigh={27583.62737,-7.873490263},
  R=91.25597243238283);

constant IdealGases.Common.DataRecord AL2 (
  name="AL2",
  MM=0.053963076,
  Hf=9289717.324490547,
  H0=187889.9935207548,
  Tlimit=1000,
  alow={-5281.50965,-17.27374523,4.60407701,-0.000261646777,6.30231997e-007,-
3.29093859e-010,
      8.88836513999999e-014},
  blow={59007.0639,3.060188921},
  ahigh={-2320724.102,9218.70789,-9.44695187,0.00999992001,-3.154798085e-006,
         4.36154481e-010,-2.24115724e-014},
  bhigh={2904.589544,99.6032074500001},
  R=154.0770581721472);

constant IdealGases.Common.DataRecord AL2Br6 (
  name="AL2Br6",
  MM=0.533387076,
  Hf=-1766864.728083513,
  H0=71650.00938267952,
  Tlimit=1000,
  alow={64833.197,-1947.687811,28.86989822,-0.01365825785,1.557883731e-005,-
9.48694200000001e-009,
      2.385519396e-012},
  blow={-110152.6833,-102.5934782},
  ahigh={-173182.949,-43.4229978000001,22.0341421,-1.421187503e-005,
         3.23863134e-009,-3.80387976e-013,1.796525455e-017},
  bhigh={-120233.8355,-62.0105495},
  R=15.58806422973773);

```

```
constant IdealGases.Common.DataRecord AL2C2 (
  name="AL2C2",
  MM=0.077984476,
  Hf=6988288.284452922,
  H0=207284.8575657545,
  Tlimit=1000,
  allow={11109.87147,-350.243423,8.598856,0.001832528671,1.035743392e-006,-
2.09164989e-009,
  7.716068700000001e-013},
  blow={64927.6419,-16.45080239},
  ahighe={159765.1967,-1644.007203,11.64643675,-0.000437921543,9.36964011e-008,
-1.049615854e-011,4.78046893e-016},
  bhigh={72048.89810000001,-36.48164252},
  R=106.6170143914284);

constant IdealGases.Common.DataRecord AL2CL6 (
  name="AL2CL6",
  MM=0.266681076,
  Hf=-4863023.134044952,
  H0=128044.6686063318,
  Tlimit=1000,
  allow={134093.5279,-3001.037953,31.43000879,-0.01700408496,1.786540089e-005,
-1.015419943e-008,2.409630451e-012},
  blow={-147183.0046,-127.4445324},
  ahighe={-275297.5554,-92.48235969999999,22.07058224,-2.870537915e-005,
6.42332569e-009,-7.4365852e-013,3.47206467e-017},
  bhigh={-162915.7014,-70.91602129},
  R=31.17758531917728);

constant IdealGases.Common.DataRecord AL2F6 (
  name="AL2F6",
  MM=0.1679534952,
  Hf=-15673929.42829332,
  H0=155186.3447019232,
  Tlimit=1000,
  allow={191506.1289,-3168.24025,22.45374802,0.01271532036,-2.488697302e-005,
1.955252096e-008,-5.71208489e-012},
  blow={-304996.255,-93.96473100999999},
  ahighe={-585298.2609999999,-545.0571990000001,22.4068632,-0.0001627323704,
3.59522504e-008,-4.12120144e-012,1.909095384e-016},
  bhigh={-321968.391,-85.89867941},
  R=49.50460834470327);

constant IdealGases.Common.DataRecord AL2I6 (
  name="AL2I6",
  MM=0.815389896,
  Hf=-598176.4520172568,
  H0=50185.76536297918,
  Tlimit=1000,
  allow={25875.71514,-1280.719742,26.7771582,-0.00991301847000002,
1.168760413e-005,-7.30391853e-009,1.874532749e-012},
  blow={-58911.0555,-83.30209321},
  ahighe={-121011.2399,-24.67542774,22.0199265,-8.46202259e-006,
1.957831901e-009,-2.326573151e-013,1.108910373e-017},
  bhigh={-65485.2375,-55.28536481},
  R=10.19692792465017);
```

```

constant IdealGases.Common.DataRecord AL2O (
  name="AL2O",
  MM=0.069962476,
  Hf=-2124157.040982941,
  H0=182619.5230711961,
  Tlimit=1000,
  allow={7776.5307,-129.4235361,4.91250952,0.00860422344999999,-1.217703648e-
005,
         8.31463487e-009,-2.237722201e-012},
  blow={-18865.12879,-0.02806368311},
  ahighe={-117107.4351,-178.3009166,7.63321536,-5.33593177e-005,
          1.180702791e-008,-1.35544579e-012,6.28732389e-017},
  bhigh={-19475.80149,-14.15764167},
  R=118.8418774658576);

constant IdealGases.Common.DataRecord AL2Oplus (
  name="AL2Oplus",
  MM=0.0699619274,
  Hf=9276048.732756898,
  H0=185545.8887772151,
  Tlimit=1000,
  allow={68289.25719999999,-909.850417,8.89656301,-0.000777255438,-
4.03465561999999e-007,
         6.97731505e-010,-2.417262484e-013},
  blow={80850.07550000001,-21.76216731},
  ahighe={-110222.5105,-121.4571732,7.59159595,-3.69453916e-005,
          8.21852032e-009,-9.47364971999999e-013,4.40862135e-017},
  bhigh={76149.43580000001,-12.82233856},
  R=118.842809353477);

constant IdealGases.Common.DataRecord AL2O2 (
  name="AL2O2",
  MM=0.08596187600000001,
  Hf=-4689236.737923215,
  H0=184298.1300221973,
  Tlimit=1000,
  allow={-19405.60042,250.8489836,3.62140379,0.01951385302,-2.560329071e-005,
         1.662721576e-008,-4.3123962e-012},
  blow={-51726.9728,9.923995945},
  ahighe={-194061.1656,-460.975243,10.84375637,-0.0001376042893,
          3.044733119e-008,-3.49619392e-012,1.622305079e-016},
  bhigh={-49630.5578,-29.4653809},
  R=96.72278441201074);

constant IdealGases.Common.DataRecord AL2O2plus (
  name="AL2O2plus",
  MM=0.0859613274,
  Hf=6484762.995877144,
  H0=174207.8147574069,
  Tlimit=1000,
  allow={82920.3499,-1757.427015,15.25328567,-0.008983131330000001,
         8.9539545e-006,-4.8405862e-009,1.096696856e-012},
  blow={73116.7714,-55.1709255},
  ahighe={-165200.5184,-60.2160697,10.04626917,-1.892194088e-005,
          4.2533012e-009,-4.94239041999999e-013,2.314507373e-017},
  bhigh={63862.19,-23.45753361},
  R=96.72340169097949);

```

```
constant IdealGases.Common.DataRecord AL2O3 (
  name="AL2O3",
  MM=0.101961276,
  Hf=-5363708.178779559,
  H0=192210.9134844487,
  Tlimit=1000,
  alow={-7443.37432, 88.2900421000001, 5.26466264, 0.02507678848, -3.43454165e-
005,
        2.30251698e-008, -6.12252928e-012},
  blow={-68726.85950000001, 2.202324298},
  ahigh={-277778.4969, -491.746593, 13.86703888, -0.000146938194, 3.25040649e-008,
         -3.73086735e-012, 1.730444284e-016},
  bhigh={-67907.5785, -43.75559873},
  R=81.54538983996238);

constant IdealGases.Common.DataRecord AL2S (
  name="AL2S",
  MM=0.086028076,
  Hf=2565198.959000316,
  H0=162783.2639195604,
  Tlimit=1000,
  alow={40946.2437, -779.7172469999999, 9.814423100000001, -0.00394033982,
        3.91246543e-006, -2.105891677e-009, 4.74600108e-013},
  blow={28339.63305, -24.65067552},
  ahigh={-70434.3737, -26.76376869, 7.52023382, -8.16612306e-006, 1.81602604e-009,
         -2.091989298e-013, 9.72753769999999e-018},
  bhigh={24227.18377, -10.66580982},
  R=96.64835466040181);

constant IdealGases.Common.DataRecord AL2S2 (
  name="AL2S2",
  MM=0.118093076,
  Hf=1145599.09507311,
  H0=153264.8789671632,
  Tlimit=1000,
  alow={66145.8732, -1111.319537, 12.57007304, -0.001456053256, -5.85162711e-007,
        1.390055053e-009, -5.4964682e-013},
  blow={19144.56895, -36.00277092},
  ahigh={-137780.6456, -77.050325, 10.55723566, -2.277554487e-005,
        5.00768285e-009, -5.715862119999999e-013, 2.63799924e-017},
  bhigh={13133.36091, -22.56851594},
  R=70.40609222508525);

constant IdealGases.Common.DataRecord Ar (
  name="Ar",
  MM=0.039948,
  Hf=0,
  H0=155137.3785921698,
  Tlimit=1000,
  alow={0,0,2.5,0,0,0,0},
  blow={-745.375, 4.37967491},
  ahigh={20.10538475, -0.05992661069999999, 2.500069401, -3.99214116e-008,
        1.20527214e-011, -1.819015576e-015, 1.078576636e-019},
  bhigh={-744.993961, 4.37918011},
  R=208.1323720837088);

constant IdealGases.Common.DataRecord Arplus (
```

```

name="Arplus",
MM=0.0399474514,
Hf=38219669.92868035,
H0=155353.7906050247,
Tlimit=1000,
alow={-57312.0917, 793.079147, -1.717121217, 0.01044184018, -1.180207501e-005,
      6.52813478e-009, -1.44755813e-012},
blow={179057.223, 29.4915095},
ahigh={-383596.54, 816.20197, 2.301342628, -4.95298377e-006, 1.205108477e-008, -
2.185050286e-012,
      1.265493898e-016},
bhigh={177181.1455, 7.94750748},
R=208.135230374171);

constant IdealGases.Common.DataRecord B (
  name="B",
  MM=0.010811,
  Hf=53241953.5658126,
  H0=584225.326056794,
  Tlimit=1000,
  alow={118.2394638, -0.0700991691, 2.500236159, -4.5842137e-007, 5.12318583e-010,
        -3.057217674e-013, 7.533815325e-017},
  blow={68483.59080000001, 4.20950192},
  ahigh={-107265.961, 322.530716, 2.126407232, 0.0002106579339, -5.93712916e-008,
        7.37742799e-012, -2.282443381e-016},
  bhigh={66434.13099999999, 6.87706967},
  R=769.0752011839794);

constant IdealGases.Common.DataRecord Bplus (
  name="Bplus",
  MM=0.0108104514,
  Hf=127868437.3901352,
  H0=573281.1490184396,
  Tlimit=1000,
  alow={0.07849791190000001, -0.000894748095, 2.500004085, -9.577271229999999e-
009,
        1.218136411e-011, -7.98675252e-015, 2.113769829e-018},
  blow={165508.026, 2.419053631},
  ahigh={-8911.54803, 4.58779009, 2.531500086, -4.9039491e-005, 2.853326582e-008,
        -7.38217591e-012, 7.12072156e-016},
  bhigh={165452.6303, 2.23866978},
  R=769.1142295871198);

constant IdealGases.Common.DataRecord Bminus (
  name="Bminus",
  MM=0.0108115486,
  Hf=50189988.32415183,
  H0=580175.7206178586,
  Tlimit=1000,
  alow={22.01568105, -0.00474046888, 2.500014238, -2.497056599e-008,
        2.556360708e-011, -1.41527478e-014, 3.27177071e-018},
  blow={64517.9188, 4.61636729},
  ahigh={21.18018248, 0.0002070697496, 2.499999729, 1.456693631e-010, -
3.88857176e-014,
        5.09059863e-018, -2.601566168e-022},
  bhigh={64517.8914, 4.61645583},
  R=769.0361767416002);

```

```
constant IdealGases.Common.DataRecord BBr (
  name="BBr",
  MM=0.090715,
  Hf=2656146.81144243,
  H0=99179.38598908672,
  Tlimit=1000,
  allow={37960.935,-497.609602,5.2670082,-3.74284019e-005,-1.13154437e-006,
         1.261945702e-009,-4.28881188e-013},
  blow={30381.25367,-4.35340733},
  ahighe={253685.8027,-640.59095,4.66682347,0.000417548639,-2.963714868e-007,
          7.51911954e-011,-4.83536832e-015},
  bhigh={31890.8434,-0.8255870910000001},
  R=91.65487515846331);

constant IdealGases.Common.DataRecord BBr2 (
  name="BBr2",
  MM=0.170619,
  Hf=573374.6944947515,
  H0=71508.54828594705,
  Tlimit=1000,
  allow={65165.5071,-902.959076,8.063817800000001,0.00112984901,-4.19548239e-006,
         3.96565877e-009,-1.283029835e-012},
  blow={14704.66408,-13.36453164},
  ahighe={-419163.337,415.768847,7.14816983,-0.000568959075,3.19544177e-007,-5.55317699e-011,
          3.19113676e-015},
  bhigh={6118.98205,-5.50275962},
  R=48.73121985241972);

constant IdealGases.Common.DataRecord BBr3 (
  name="BBr3",
  MM=0.250523,
  Hf=-819485.6360493847,
  H0=62680.06131173585,
  Tlimit=1000,
  allow={39680.7369,-633.108716,8.054551569999999,0.009777571299999999,-1.614866025e-005,
         1.216119883e-008,-3.51653961e-012},
  blow={-23667.23308,-11.0609939},
  ahighe={-190104.6795,-150.3627687,10.10999896,-4.32309511e-005,
          9.409696190000001e-009,-1.06521847e-012,4.883048770000001e-017},
  bhigh={-27425.53062,-19.97496003},
  R=33.18845774639455);

constant IdealGases.Common.DataRecord BC (
  name="BC",
  MM=0.0228217,
  Hf=36726532.42308855,
  H0=382754.3522174071,
  Tlimit=1000,
  allow={-39157.58,728.453806,-1.552743361,0.01552898673,-1.976857863e-005,
         1.272066405e-008,-3.22998278e-012},
  blow={96449.13680000001,32.48204466},
  ahighe={-2346280.674,6450.7513,-2.619532384,0.0033391605,-4.50802214e-007,
          1.351919576e-011,8.265104270000001e-016},
  bhigh={57866.8507,50.37884876},
  R=364.323078473558);
```

```

constant IdealGases.Common.DataRecord BC2 (
  name="BC2",
  MM=0.0348324,
  Hf=23003259.17823635,
  H0=335544.6366027033,
  Tlimit=1000,
  allow={-29874.22175, 364.312658, 2.963469415, 0.00656175015, -3.53976724e-006,
         1.927356029e-010, 3.118724866e-013},
  blow={93048.3573, 10.83424193},
  ahighelement1={1525485.889, -5987.862380000001, 14.08384855, -0.0037474998,
                1.081927223e-006, -1.378744645e-010, 6.473822e-015},
  bhighelement1={131120.4013, -63.37215498},
  R=238.6993718491979);

constant IdealGases.Common.DataRecord BCL (
  name="BCL",
  MM=0.046264,
  Hf=3959308.25263704,
  H0=191538.4964551271,
  Tlimit=1000,
  allow={22024.58989, -170.0511155, 3.066075869, 0.00559507018, -8.46087041e-006,
         6.08473341e-009, -1.702889433e-012},
  blow={21974.02537, 6.388978917},
  ahighelement1={-74212.6254, 263.8090127, 3.60022765, 0.001018866266, -
4.666184119999999e-007,
                9.849002029999999e-011, -6.46881817e-015},
  bhighelement1={19127.2648, 5.235317877},
  R=179.7179664533979);

constant IdealGases.Common.DataRecord BCLplus (
  name="BCLplus",
  MM=0.0462634514,
  Hf=26679375.67667076,
  H0=191518.7417253525,
  Tlimit=1000,
  allow={65144.0542, -684.554101, 5.52727682, -0.000445835745, -4.55772884e-007,
         6.37085816e-010, -2.13436612e-013},
  blow={150942.4507, -6.918246393},
  ahighelement1={-216942.3542, 358.997516, 4.03520837, 0.000327445243, -8.63457845e-008,
                1.207061044e-011, -5.262253270000001e-016},
  bhighelement1={144547.7224, 3.488222017},
  R=179.7200975800954);

constant IdealGases.Common.DataRecord BCLOH (
  name="BCLOH",
  MM=0.06327134,
  Hf=-3698441.996012729,
  H0=196479.4644779137,
  Tlimit=1000,
  allow={-28984.27288, 689.404658, -2.288138069, 0.0334942862, -4.72522598e-005,
         3.32698131e-008, -9.224238860000001e-012},
  blow={-32619.811, 39.76886651},
  ahighelement1={702773.844, -2857.391274, 10.9117584, -8.10770934e-005, -2.124440337e-
008,
                5.40991244e-012, -3.46415384e-016},
  bhighelement1={-13072.21888, -35.85168799},
  R=131.4097662543578);

```

```
constant IdealGases.Common.DataRecord BCL_OH_2 (
  name="BCL_OH_2",
  MM=0.0802786799999999,
  Hf=-10032397.8670302,
  H0=168975.1625213569,
  Tlimit=1000,
  allow={80726.23009999999,-859.461982,2.705039614,0.0387879962,-5.72173435e-
005,
        4.14585033e-008,-1.168536754e-011},
  blow={-93799.10739999999,7.134982695},
  ahighe={1423878.191,-5849.70407,17.92165219,-0.0002009909087,-3.39343578e-
008,
        9.838560759999999e-012,-6.47267882e-016},
  bhigh={-64742.6176,-81.00399519},
  R=103.5701135095894);

constant IdealGases.Common.DataRecord BCL2 (
  name="BCL2",
  MM=0.081717,
  Hf=-745028.4518521237,
  H0=140896.2761726446,
  Tlimit=1000,
  allow={35988.7942,-360.28265,4.15768545,0.0115287745,-1.816567395e-005,
        1.341714418e-008,-3.84082286e-012},
  blow={-6765.107050000001,5.182311874},
  ahighe={350140.313,-1662.20359,8.83953472,-0.000977254411,2.467356248e-007,-
2.501151018e-011,
        8.722954559999999e-016},
  bhigh={568.2909440000001,-21.48348344},
  R=101.7471517554487);

constant IdealGases.Common.DataRecord BCL2plus (
  name="BCL2plus",
  MM=0.0817164514,
  Hf=8227417.985994433,
  H0=157236.4166562426,
  Tlimit=1000,
  allow={80659.5624,-1149.602601,10.16765297,-0.00358649491,2.837145924e-006,-
1.225075731e-009,
        2.224794198e-013},
  blow={84786.1741,-29.37256994},
  ahighe={302929.3331,-1305.533159,8.920390060000001,-0.000728991964,
        1.727833266e-007,-1.523567337e-011,4.22691144e-016},
  bhigh={86534.15770000001,-22.67466616},
  R=101.747834830723);

constant IdealGases.Common.DataRecord BCL2OH (
  name="BCL2OH",
  MM=0.0987243399999999,
  Hf=-6127338.445615337,
  H0=142333.3799952474,
  Tlimit=1000,
  allow={20997.5517,-247.0699169,2.578761742,0.03120362393,-4.54531405e-005,
        3.24201177e-008,-9.039822660000001e-012},
  blow={-73090.2862,12.73658611},
  ahighe={587811.379,-2982.969317,14.00662004,-0.0001194189832,-1.269775742e-
008,
        4.42289219e-012,-3.004089365e-016},
```

```

bhigh={-58241.6807,-51.50858228},
R=84.2190689752902);

constant IdealGases.Common.DataRecord BF(
  name="BF",
  MM=0.0298094032,
  Hf=-3587186.341254897,
  H0=291675.4133474233,
  Tlimit=1000,
  allow={-52389.5473,811.8476640000001,-1.141614903,0.01161249417,-
1.175212617e-005,
       6.01923278e-009,-1.238293129e-012},
  blow={-17745.41998,30.05086287},
  ahigh={-374638.978,560.449391,3.60918611,0.000618721693,-1.77893877e-007,
         2.426601527e-011,-9.394651579999999e-016},
  bhigh={-18191.79292,3.71660929},
  R=278.9211157370638);

constant IdealGases.Common.DataRecord BFCL(
  name="BFCL",
  MM=0.0652624032,
  Hf=-4277869.865509336,
  H0=169056.6001100003,
  Tlimit=1000,
  allow={-23635.12735,438.316898,0.391809572,0.01719990692,-2.139592424e-005,
         1.337389186e-008,-3.37116469e-012},
  blow={-36871.6659,26.68102752},
  ahigh={-108630.9561,-523.343848,7.29750474,-3.054410701e-005,-3.154598306e-
008,
         1.052290533e-011,-7.615432260000001e-016},
  bhigh={-33058.9121,-11.86586595},
  R=127.4006409865091);

constant IdealGases.Common.DataRecord BFCL2(
  name="BFCL2",
  MM=0.1007154032,
  Hf=-6384326.325171282,
  H0=131369.3097542005,
  Tlimit=1000,
  allow={4402.55124,-157.1398929,4.01632231,0.01743396078,-2.25278906e-005,
         1.433673443e-008,-3.64875558e-012},
  blow={-78224.5261999999,6.886094746},
  ahigh={-223361.3309,-644.060296,10.47861848,-0.0001911892367,
         4.22488779e-008,-4.84726692e-012,2.247974725e-016},
  bhigh={-77406.2648,-28.14630564},
  R=82.5541251469666);

constant IdealGases.Common.DataRecord BFOH(
  name="BFOH",
  MM=0.0468167432,
  Hf=-9646798.284764072,
  H0=255744.4448634778,
  Tlimit=1000,
  allow={-75639.5367,1354.838128,-5.64821011,0.0387027466,-5.03765918e-005,
         3.33959264e-008,-8.847822949999999e-012},
  blow={-61944.4611,58.0036282},
  ahigh={725131.809,-3215.62085,11.17909777,-0.0001883881929,2.57970808e-009,
         1.052290533e-011,-7.615432260000001e-016});

```

```
2.6651581e-012,-2.186647438e-016},
bhigh={-37168.0808,-39.7846349},
R=177.5961212099008);

constant IdealGases.Common.DataRecord BF_OH_2 (
  name="BF_OH_2",
  MM=0.0638240831999999,
  Hf=-16449739.44568937,
  H0=201886.4878892612,
  Tlimit=1000,
  allow={13818.00676,252.9637004,-3.84455204,0.0525514085,-7.19311773e-005,
         4.94830991e-008,-1.345922496e-011},
  blow={-128312.3258,42.9365937},
  ahigh={1422037.64,-6257.297570000001,18.22460686,-0.000322139645,-
7.1267399e-009,
         6.75869977e-012,-5.04255454e-016},
  bhigh={-91862.5419,-85.27895049999999},
  R=130.2717028295677);

constant IdealGases.Common.DataRecord BF2 (
  name="BF2",
  MM=0.0488078063999999,
  Hf=-10232521.00098479,
  H0=217428.3743266118,
  Tlimit=1000,
  allow={-67876.51760000001,1085.903536,-3.023320961,0.02326503687,-
2.641444147e-005,
         1.515620683e-008,-3.51855918e-012},
  blow={-66409.18250000001,44.31968310000001},
  ahigh={-115309.1296,-810.9800119999999,7.60270923,-0.0002409209242,
         5.32847186e-008,-6.11879794e-012,2.839984178e-016},
  bhigh={-57962.2217,-16.55644047},
  R=170.3512739716162);

constant IdealGases.Common.DataRecord BF2plus (
  name="BF2plus",
  MM=0.0488072578,
  Hf=6609394.064339341,
  H0=217436.5755906082,
  Tlimit=1000,
  allow={-29383.56477,295.5698598,1.894274526,0.01280508319,-1.388757254e-005,
         7.62436675e-009,-1.706339371e-012},
  blow={35989.9244,13.84803311},
  ahigh={-176940.9966,-534.368518,7.72805291,-1.958622052e-006,-2.419680893e-
008,
         5.73253381e-012,-3.21635423e-016},
  bhigh={39087.4505,-19.36560315},
  R=170.3531887423513);

constant IdealGases.Common.DataRecord BF2minus (
  name="BF2minus",
  MM=0.048808355,
  Hf=-15034371.94308229,
  H0=213766.0857449509,
  Tlimit=1000,
  allow={21097.65318,143.95646,0.2604646296,0.01817373751,-2.241565287e-005,
         1.369712886e-008,-3.35595268e-012},
```

```

blow={-89718.0178,23.51692604},
ahigh={-170992.0627,-615.621979,7.45887137,-0.0001836703248,
      4.06436395999999e-008,-4.66770095e-012,2.16626484e-016},
bhigh={-87414.0453,-16.20604664},
R=170.3493592439245);

constant IdealGases.Common.DataRecord BF2CL(
  name="BF2CL",
  MM=0.08426080640000001,
  Hf=-10538707.59062662,
  H0=146256.5281122208,
  Tlimit=1000,
  allow={4500.10125,-119.6284966,3.19863514,0.01743426933,-1.994070287e-005,
         1.12513643e-008,-2.544020371e-012},
  blow={-107677.9851,10.08195015},
  ahigh={-206681.7398,-1024.447292,10.75843981,-0.0003023235319,
         6.67252147e-008,-7.6499419e-012,3.5462245e-016},
  bhigh={-104693.6725,-32.29995036},
  R=98.67543826402306);

constant IdealGases.Common.DataRecord BF2OH(
  name="BF2OH",
  MM=0.0658151464,
  Hf=-16595227.41713448,
  H0=188343.9858153989,
  Tlimit=1000,
  allow={-7522.52165,436.712169,-3.128973474,0.04323957680000001,-5.78549404e-
005,
         3.88585813e-008,-1.038574523e-011},
  blow={-134425.9458,41.3917055},
  ahigh={576075.9250000001,-3611.90019,14.4689764,-0.0003028293733,
         2.764337508e-008,-1.90549317e-013,-8.69497194e-017},
  bhigh={-113358.9041,-58.9698151},
  R=126.3306769762044);

constant IdealGases.Common.DataRecord BF3(
  name="BF3",
  MM=0.0678062096,
  Hf=-16753627.82703017,
  H0=171831.2536378674,
  Tlimit=1000,
  allow={3465.584,21.33198651,1.641245191,0.01993755064,-2.15011993e-005,
         1.145669081e-008,-2.442285789e-012},
  blow={-137945.5591,16.25533544},
  ahigh={-181976.7014,-1405.347931,11.03412258,-0.000410459105,
         9.031277570000001e-008,-1.03305736e-011,4.780551830000001e-016},
  bhigh={-132313.6863,-37.3838608},
  R=122.6210998822739);

constant IdealGases.Common.DataRecord BF4minus(
  name="BF4minus",
  MM=0.0868051614,
  Hf=-20289876.42663378,
  H0=158926.5289932403,
  Tlimit=1000,
  allow={206848.52,-2463.190805,12.06414196,0.01102766923,-1.75172076e-005,
         1.201346256e-008,-3.150588626e-012},
  blow={-107677.9851,10.08195015},
  ahigh={-206681.7398,-1024.447292,10.75843981,-0.0003023235319,
         6.67252147e-008,-7.6499419e-012,3.5462245e-016},
  bhigh={-104693.6725,-32.29995036},
  R=98.67543826402306);

```

```
blow={-201056.9206,-46.1082694},  
ahigh={-437324.171,-775.928941,13.58222575,-0.0002341024178,5.19691594e-008,  
-5.982245780000001e-012,2.781163206e-016},  
bhigh={-212725.1877,-49.533697},  
R=95.783152359878);  
  
constant IdealGases.Common.DataRecord BH (   
  name="BH",  
  MM=0.01181894,  
  Hf=37966789.23829041,  
  H0=730954.2141681063,  
  Tlimit=1000,  
  allow={20630.8255,-368.250252,6.07133787,-0.00872832107,1.458566459e-005,-  
1.036840401e-008,  
  2.779462579e-012},  
  blow={54604.5992,-13.00392582},  
  ahigh={-1098531.663,-174.5890126,8.442426810000001,-0.00544019667,  
  2.718307052e-006,-4.83981221e-010,2.868523222e-014},  
  bhigh={50167.3567,-29.71030686},  
  R=703.4871147497153);  
  
constant IdealGases.Common.DataRecord BHCL (   
  name="BHCL",  
  MM=0.04727194,  
  Hf=2991590.931110507,  
  H0=217989.9322938725,  
  Tlimit=1000,  
  allow={47632.7491,-442.001652,3.97564588,0.00854169875999999,-1.377785182e-  
005,  
  1.131521587e-008,-3.56965918e-012},  
  blow={18222.78428,3.028452399},  
  ahigh={349999.172,-2238.404757,8.71110605,-0.001088882906,4.18930109e-007,-  
6.56871126e-011,  
  3.6200676e-015},  
  bhigh={28412.40364,-25.89566225},  
  R=175.8859907166916);  
  
constant IdealGases.Common.DataRecord BHCL2 (   
  name="BHCL2",  
  MM=0.08272494,  
  Hf=-3044831.81251023,  
  H0=141984.170674527,  
  Tlimit=1000,  
  allow={44199.1481,-238.4739577,1.356682235,0.02517866897,-3.6329104e-005,  
  2.644335581e-008,-7.62504411e-012},  
  blow={-30038.77914,17.87024616},  
  ahigh={411099.606,-2858.027441,11.79547506,-0.000632281797,1.269551549e-007,  
  -1.352862344e-011,5.92165351e-016},  
  bhigh={-16172.8876,-42.17900972},  
  R=100.5074406823384);  
  
constant IdealGases.Common.DataRecord BHF (   
  name="BHF",  
  MM=0.0308173432,  
  Hf=-2466544.325599099,  
  H0=325857.3243912862,  
  Tlimit=1000,
```

```

alow={-49083.6226, 961.076021000001, -2.919850316, 0.02149156633, -
2.595677863e-005,
    1.693492714e-008, -4.55364884e-012},
blow={-14669.25195, 41.6070931},
ahigh={1184821.3, -4677.43518, 10.67328148, -0.001606366598, 3.7045347e-007, -
3.79788275e-011,
    1.432441145e-015},
bhigh={18026.40937, -42.85931169999999},
R=269.798468545465};

constant IdealGases.Common.DataRecord BHFCL(
  name="BHFCL",
  MM=0.0662703432,
  Hf=-7288880.616510825,
  H0=172242.0384266246,
  Tlimit=1000,
  alow={576.116972, 304.215625, -0.9808900360000001, 0.02803843648, -3.67673402e-
005,
    2.499824138e-008, -6.87342164e-012},
  blow={-60502.0739, 31.32842092},
  ahigh={439077.892, -3137.181709, 12.00286161, -0.000715366934, 1.453878608e-007,
    -1.56519707e-011, 6.910075059999999e-016},
  bhigh={-42323.6834, -44.66542881},
  R=125.4629386014708);

constant IdealGases.Common.DataRecord BHF2(
  name="BHF2",
  MM=0.0498157464,
  Hf=-14846988.44138969,
  H0=213975.4549577521,
  Tlimit=1000,
  alow={-83535.40280000001, 1616.659766, -8.161405670000001, 0.0415378679, -
4.92106328e-005,
    3.054165306e-008, -7.78681398e-012},
  blow={-97480.65459999999, 70.3863075},
  ahigh={466087.6960000001, -3728.10049, 12.44152658, -0.000890684319,
    1.841732568e-007, -2.010774349e-011, 8.979196590000001e-016},
  bhigh={-69781.4293, -51.0321804},
  R=166.904495081499);

constant IdealGases.Common.DataRecord BH2(
  name="BH2",
  MM=0.01282688,
  Hf=25642171.6738599,
  H0=781449.5029188704,
  Tlimit=1000,
  alow={28125.57296, -300.0083489, 4.824221, -0.0002186429819, 1.485457398e-006,
    3.89373968e-010, -6.331629389999999e-013},
  blow={39919.8842, -5.04268285},
  ahigh={1360117.365, -4917.70449, 9.75897103, -0.000741587064, 1.269760019e-007,
    -1.187742442e-011, 4.68255251e-016},
  bhigh={68902.92660000001, -41.9049012},
  R=648.2068905298874);

constant IdealGases.Common.DataRecord BH2CL(
  name="BH2CL",
  MM=0.04827988,

```

```
Hf=-1674522.679012458,
H0=215967.8524470235,
Tlimit=1000,
alow={16466.38117,127.7320942,0.1300268921,0.01938143193,-2.316389616e-005,
      1.607896164e-008,-4.66833765e-012},
blow={-11121.12215,23.08500112},
ahigh={1427215.437,-6265.56434,13.82183005,-0.001313843556,2.587131606e-007,
      -2.71384527e-011,1.172827257e-015},
bhigh={26393.87516,-63.99725308},
R=172.2140154449431);

constant IdealGases.Common.DataRecord BH2F(
  name="BH2F",
  MM=0.0318252832,
  Hf=-10179225.42791387,
  H0=318549.3098769974,
  Tlimit=1000,
  alow={-88379.2044,1705.285929,-8.061246130000001,0.0364318868,-4.15248255e-
005,
        2.620135756e-008,-6.930886410000001e-012},
  blow={-47872.7341,68.6208767},
  ahigh={1435082.111,-6625.32017,14.09165008,-0.00142246906,2.828724718e-007,
        -2.992498948e-011,1.302618926e-015},
  bhigh={-793.719267,-68.013564},
  R=261.2536689068646);

constant IdealGases.Common.DataRecord BH3 (
  name="BH3",
  MM=0.01383482,
  Hf=7571229.622069531,
  H0=727129.0121591751,
  Tlimit=1000,
  alow={-66196.3507,1262.658391,-4.6543559,0.02461795131,-2.501537437e-005,
        1.562330756e-008,-4.32394948e-012},
  blow={5667.58798,46.66504620000001},
  ahigh={1855778.95,-8002.492370000001,15.05692199,-0.001790456689,
        3.6125111e-007,-3.86603591e-011,1.698508879e-015},
  bhigh={59675.3707,-79.94046159999999},
  R=600.9815812565686);

constant IdealGases.Common.DataRecord BH3NH3 (
  name="BH3NH3",
  MM=0.03086534,
  Hf=-3725862.083489118,
  H0=410866.3633706935,
  Tlimit=1000,
  alow={-181106.3598,3010.341543,-15.24271135,0.0621563775,-5.95988185e-005,
        3.26742936e-008,-7.75124309e-012},
  blow={-29342.78877,108.7399608},
  ahigh={4354925.96,-18242.35232,31.6299983,-0.00320824361,5.88507545e-007,-
5.80853553e-011,
        2.3825786e-015},
  bhigh={94381.18030000001,-189.7865799},
  R=269.3789214698429);

constant IdealGases.Common.DataRecord BH4 (
  name="BH4",
```

```

MM=0.01484276,
Hf=17194273.6391345,
H0=725665.3749033199,
Tlimit=1000,
alow={27342.74578,-84.73619380000001,1.551871695,0.01439765491,-6.11115236e-
006,
      -1.324417559e-010,5.13300815e-013},
blow={30220.48074,12.50358328},
ahigh={1354714.146,-7870.76275,18.2617061,-0.001948275329,4.07270689e-007,-
4.4827109e-011,
      2.014115885e-015},
bhigh={74702.49489999999,-96.8608689},
R=560.1702109311207);

constant IdealGases.Common.DataRecord BI(
  name="BI",
  MM=0.13771547,
  Hf=2367108.807746871,
  H0=66384.71335137585,
  Tlimit=1000,
  alow={35174.4953,-544.5819300000001,6.02751101,-0.002387237926,
        2.244796277e-006,-1.110001347e-009,2.280204403e-013},
  blow={40719.1013,-7.28804984},
  ahigh={2132872.626,-6283.684960000001,11.23964343,-0.00325885675,
        7.06428215e-007,-4.23006893e-011,-4.20807847e-016},
  bhigh={77976.3481,-46.6404261},
  R=60.37427748676311);

constant IdealGases.Common.DataRecord BI2(
  name="BI2",
  MM=0.26461994,
  Hf=899767.1301716719,
  H0=47949.61407670186,
  Tlimit=1000,
  alow={66409.75900000001,-1044.411994,9.73419724,-0.00395255754,
        3.16556778e-006,-1.273764644e-009,1.879968132e-013},
  blow={32057.5089,-20.12407058},
  ahigh={-392402.223,442.290235,7.12962358,-0.000561963406,3.18076052e-007,-
5.537079720000001e-011,
        3.1839597e-015},
  bhigh={22919.72638,-3.20179324},
  R=31.42042886110548);

constant IdealGases.Common.DataRecord BI3(
  name="BI3",
  MM=0.39152441,
  Hf=54658.1501776607,
  H0=43249.08375444586,
  Tlimit=1000,
  alow={62431.1489,-1002.408628,11.06650075,0.001609588058,-5.14725025e-006,
        4.73251402e-009,-1.512999796e-012},
  blow={5160.34815,-24.20687643},
  ahigh={-150617.6506,-81.6721060999999,10.05937867,-2.319396215e-005,
        5.01996673e-009,-5.65434963e-013,2.580651348e-017},
  bhigh={-426.168791,-16.17962779},
  R=21.23615230018481);

```

```
constant IdealGases.Common.DataRecord BN (
  name="BN",
  MM=0.0248177,
  Hf=23157923.90108673,
  H0=361077.2956398055,
  Tlimit=1000,
  allow={-45697.0758, 670.4286070000001, 0.0361508908, 0.007339487509999999, -4.96903349e-006,
  1.22903138e-009, 6.34028152e-014},
  blow={64854.6508, 25.39869896},
  ahigh={-227693.2705, -102.5649298, 4.41458681, 0.0002561670989, -1.612994942e-008,
  -6.52605292999999e-013, 5.74946612e-017},
  bhigh={67844.6513, -0.6778568656},
  R=335.0218593987356);

constant IdealGases.Common.DataRecord BO (
  name="BO",
  MM=0.0268104,
  Hf=761137.6182376987,
  H0=323535.0461015128,
  Tlimit=1000,
  allow={-11662.16822, 92.17579389999999, 3.65549849, -0.00311854292,
  9.00832983e-006, -8.017789990000001e-009, 2.472952292e-012},
  blow={873.8284780000001, 4.48284739},
  ahigh={17886.00589, -630.901963, 4.5745284, 0.0001988001643, -9.70296348e-008,
  1.870854291e-011, -1.030218131e-015},
  bhigh={4841.311089999999, -3.39889058},
  R=310.1211470175753);

constant IdealGases.Common.DataRecord BOminus (
  name="BOminus",
  MM=0.0268109486,
  Hf=-10361105.8356958,
  H0=323525.4421397086,
  Tlimit=1000,
  allow={-82420.23209999999, 920.363689999999, -0.2302659981,
  0.00622990411999999, -3.157624391e-006, 1.184578107e-010,
  2.819228693e-013},
  blow={-39111.4122, 25.99226838},
  ahigh={212280.4369, -1262.124688, 5.38316731, -0.00031810491, 7.28898825e-008,
  8.14170891999999e-012,
  3.71536439e-016},
  bhigh={-27068.92261, -9.77375367},
  R=310.1148013837899);

constant IdealGases.Common.DataRecord BOCL (
  name="BOCL",
  MM=0.0622634,
  Hf=-5115954.075749156,
  H0=170371.5666025305,
  Tlimit=1000,
  allow={70523.8064, -1315.301429, 11.30087727, -0.0120711205, 1.880516131e-005,
  1.373823975e-008,
  3.85489016e-012},
  blow={-33553.991, -36.98462517},
  ahigh={155714.8487, -1446.254968, 8.50944805, -0.000385900738, 8.26226882e-008,
  -9.26102745999999e-012, 4.21999801e-016},
```

```

bhight={-32035.8904,-23.72713292},
R=133.5370699319344);

constant IdealGases.Common.DataRecord BOCL2 (
  name="BOCL2",
  MM=0.0977164,
  Hf=-3700151.786189422,
  H0=134774.2037160599,
  Tlimit=1000,
  allow={7322.66736,-202.7726965,4.16192409,0.01733309456,-2.263908718e-005,
    1.45227519e-008,-3.71959312e-012},
  blow={-44144.4188,6.52042411},
  ahight={-229752.7041,-619.456793,10.46075518,-0.0001841620739,
    4.07121543e-008,-4.67224655e-012,2.167237536e-016},
  bhight={-43715.3842,-27.43570215},
  R=85.08778465027366);

constant IdealGases.Common.DataRecord BOF (
  name="BOF",
  MM=0.0458088032,
  Hf=-12944636.61080759,
  H0=218043.1336830909,
  Tlimit=1000,
  allow={72268.02649999999,-1131.38663,8.81605452,-0.00462769457,
    7.80129958e-006,-5.68120583999999e-009,1.534732094e-012},
  blow={-67111.1600999999,-25.47957229},
  ahight={193702.0297,-1739.112297,8.695650860000001,-0.000451889557,
    9.59101908e-008,-1.067806769e-011,4.840007660000001e-016},
  bhight={-63301.5239,-27.02458617},
  R=181.5038031816557);

constant IdealGases.Common.DataRecord BOF2 (
  name="BOF2",
  MM=0.0648072064,
  Hf=-12849924.35656045,
  H0=179169.2412774639,
  Tlimit=1000,
  allow={7227.79413,21.48783624,1.157840301,0.02250585296,-2.610731536e-005,
    1.508824929e-008,-3.51836485e-012},
  blow={-101399.5733,20.10127136},
  ahight={-220502.8861,-1254.23869,10.9255138,-0.0003680144,8.10716494e-008,-
    9.28127506e-012,
    4.2975493e-016},
  bhight={-96804.4757999999,-34.8285333},
  R=128.2954853613317);

constant IdealGases.Common.DataRecord BOH (
  name="BOH",
  MM=0.02781834,
  Hf=-242882.1417812853,
  H0=360608.2893515573,
  Tlimit=1000,
  allow={-75214.1027,1391.444703,-5.63033018,0.02966358811,-3.83083518e-005,
    2.550606424e-008,-6.79585721e-012},
  blow={-8341.300280000001,55.1769003999999},
  ahight={844749.8570000001,-3016.982887,8.0838647,-0.0001609855388,-
    2.191830959e-009,
    2.191830959e-009},
  R=133.5370699319344);

```

---

**950 Modelica.Media.IdealGases.Common.SingleGasesData**

---

```
 3.127133119e-012,-2.376665451e-016},
bhigh={16476.98495,-26.03227947},
R=298.8845488264217);

constant IdealGases.Common.DataRecord BO2 (
  name="BO2",
  MM=0.0428098,
  Hf=-7220822.031404024,
  H0=251628.2953903078,
  Tlimit=1000,
  allow={-41410.9064,719.885461,-1.477629435,0.02277415194,-2.786326934e-005,
    1.718462069e-008,-4.27899144e-012},
  blow={-41776.5769,32.5845801},
  ahigh={-38344.5234,-956.326114,8.20096277999999,-0.0002062130802,
    9.87228899e-009,8.15836676e-012,-7.52751966e-016},
  bhigh={-34235.6402,-22.24772278},
  R=194.2188938046896);

constant IdealGases.Common.DataRecord BO2minus (
  name="BO2minus",
  MM=0.0428103486,
  Hf=-16689749.42661411,
  H0=224187.4059395069,
  Tlimit=1000,
  allow={53242.7969,-732.631303,5.90247,0.0024374652,-4.87880395e-007,-
    8.47572476e-010,
    4.08078118e-013},
  blow={-83442.9368,-10.53931911},
  ahigh={119922.6372,-1715.766982,8.70103821999999,-0.000460124212,
    9.867011870000001e-008,-1.107288516e-011,5.05018324e-016},
  bhigh={-78257.89689999999,-28.39356505},
  R=194.2164049558802);

constant IdealGases.Common.DataRecord B_OH_2 (
  name="B_OH_2",
  MM=0.04482568,
  Hf=-9486607.721288333,
  H0=267071.4197754501,
  Tlimit=1000,
  allow={1491.659222,364.818484,-2.906006389,0.0413683446,-5.78258422e-005,
    4.069783560000001e-008,-1.126499791e-011},
  blow={-53754.8366,38.8853518},
  ahigh={1557023.406,-5784.71076,14.87391235,-0.0001821804765,-3.80354164e-
    008,
    1.030332746e-011,-6.68588983e-016},
  bhigh={-18068.10301,-65.902689},
  R=185.4845704515805);

constant IdealGases.Common.DataRecord BS (
  name="BS",
  MM=0.042876,
  Hf=6379312.552476911,
  H0=203472.9219143577,
  Tlimit=1000,
  allow={-38394.901,698.941749,-1.21398046,0.01396391264,-1.689881284e-005,
    1.038889985e-008,-2.582063989e-012},
  blow={28656.85472,31.54819945},
```

```

ahigh={1358760.165,-4364.03596,9.034307589999999,-0.002114548699,
        4.18927531e-007,-1.354787322e-011,-1.360684605e-015},
bhigh={59130.6545,-33.36696868},
R=193.9190222968561;

constant IdealGases.Common.DataRecord BS2 (
  name="BS2",
  MM=0.07494100000000001,
  Hf=852236.559426749,
  H0=180936.4700230848,
  Tlimit=1000,
  allow={18202.64374,-515.067189,8.547515000000001,-0.001626155268,
         1.866387194e-006,-1.277165468e-009,3.68198069e-013},
  blow={8186.839750000001,-17.78384854},
  ahigh={512040.251,-1938.581076,9.814873459999999,-0.001339731606,
         3.75430116e-007,-4.40038378e-011,1.887662968e-015},
  bhigh={17365.35,-27.57546712},
  R=110.9469048985202);

constant IdealGases.Common.DataRecord B2 (
  name="B2",
  MM=0.021622,
  Hf=39652691.98039035,
  H0=407229.0259920451,
  Tlimit=1000,
  allow={-151641.8096,2630.168946,-13.81358321,0.05216225190000001,-
6.93049474e-005,
         4.46941039e-008,-1.128496456e-011},
  blow={89952.5105,98.13118490000001},
  ahigh={1094594.495,-2602.735739,6.90945621,-0.0006405949889999999,
         1.951732355e-007,-2.555902119e-011,1.053557323e-015},
  bhigh={118780.7611,-19.49045025},
  R=384.5376005919897);

constant IdealGases.Common.DataRecord B2C (
  name="B2C",
  MM=0.0336327,
  Hf=23799237.46829722,
  H0=348562.9759133224,
  Tlimit=1000,
  allow={-41665.1987,579.777952,1.411017091,0.01174705713,-1.066474425e-005,
         4.71723262e-009,-7.97001326e-013},
  blow={91968.742,17.69254387},
  ahigh={1159241.019,-4567.79881,12.28672097,-0.002500830701,6.40997985e-007,
         -6.93293570000001e-011,2.672933741e-015},
  bhigh={122244.2137,-51.99066186},
  R=247.2139316795856);

constant IdealGases.Common.DataRecord B2CL4 (
  name="B2CL4",
  MM=0.163434,
  Hf=-2998152.159281422,
  H0=132090.8256543926,
  Tlimit=1000,
  allow={42855.5513,-728.074881,9.315763670000001,0.02331629623,-3.51124761e-
005,
         2.464057658e-008,-6.73547996e-012},

```

---

**952 Modelica.Media.IdealGases.Common.SingleGasesData**

---

```
blow={-58190.2324,-16.22083878},  
ahigh={-642798.703,507.856299,14.86725022,2.026983515e-006,6.27271881e-009,  
-1.218822573e-012,7.24120840999999e-017},  
bhigh={-68200.66,-41.39110669},  
R=50.87357587772434);  
  
constant IdealGases.Common.DataRecord B2F4 (  
  name="B2F4",  
  MM=0.0976156127999999,  
  Hf=-14731250.0403624,  
  H0=181126.6916515224,  
  Tlimit=1000,  
  allow={-59716.9564,818.9425440000001,-0.643767129,0.0385896607,-4.31305319e-  
005,  
        2.404463223e-008,-5.39355935e-012},  
  blow={-179004.1225,35.5217997},  
  ahigh={38558.6589,-2928.842655,17.92063736,-0.000952399827,1.614708845e-007,  
        -1.428613504e-011,5.21873967e-016},  
  bhigh={-161305.1924,-71.8480325},  
  R=85.17563698580808);  
  
constant IdealGases.Common.DataRecord B2H (  
  name="B2H",  
  MM=0.02262994,  
  Hf=35186239.86630102,  
  H0=447000.6106953885,  
  Tlimit=1000,  
  allow={87755.7974,-1343.931564,9.83209915,-0.00560713775,  
        5.73065104999999e-006,-2.266436197e-009,1.97235208e-013},  
  blow={100990.8698,-32.8246148},  
  ahigh={617554.5750000001,-2627.897466,9.08490507,-0.000538941833,  
        1.0507147e-007,-1.092459556e-011,4.68485466e-016},  
  bhigh={109953.7555,-31.6402991},  
  R=367.4102538495462);  
  
constant IdealGases.Common.DataRecord B2H2 (  
  name="B2H2",  
  MM=0.02363788,  
  Hf=19235140.33407395,  
  H0=447300.6462508482,  
  Tlimit=1000,  
  allow={142161.7322,-2138.011262,13.17732399,-0.00894265807,1.127804935e-005,  
        -5.7878862e-009,1.000813871e-012},  
  blow={63723.19250000001,-53.4698274},  
  ahigh={1240614.592,-5368.96502,13.79292454,-0.001136749748,2.245617156e-007,  
        -2.361530467e-011,1.022604572e-015},  
  bhigh={85065.1149,-64.354613},  
  R=351.7435573748576);  
  
constant IdealGases.Common.DataRecord B2H3 (  
  name="B2H3",  
  MM=0.02464582,  
  Hf=14244716.58885767,  
  H0=484407.5384791417,  
  Tlimit=1000,  
  allow={94251.4716,-1075.866284,6.38399939,0.01167858327,-1.429661252e-005,  
        1.128776882e-008,-3.68132506e-012},
```

```

blow={46353.3508,-14.00002845},
ahigh={1771506.902,-7872.26093,17.89447015,-0.001709304785,3.40957543e-007,
      -3.61454368e-011,1.575671409e-015},
bhigh={87377.1189,-90.5462946},
R=337.3583025437985);

constant IdealGases.Common.DataRecord B2H3_db (
  name="B2H3_db",
  MM=0.02464582,
  Hf=14339479.35187387,
  H0=483296.0315380052,
  Tlimit=1000,
  allow={60640.1484,-752.446633,5.58448384,0.01080468494,-8.64903281e-006,
         4.7483331e-009,-1.309096028e-012},
  blow={44917.923,-8.87368682},
  ahigh={1497731.017,-7219.31414,17.50463766,-0.0015796988,3.16410347e-007,-
3.36722515e-011,
        1.472919654e-015},
  bhigh={83243.4967,-87.79549259999999},
  R=337.3583025437985);

constant IdealGases.Common.DataRecord B2H4 (
  name="B2H4",
  MM=0.02565376,
  Hf=8231232.341769785,
  H0=481015.6094077438,
  Tlimit=1000,
  allow={40001.323,-307.1548876,1.763071673,0.02646361882,-3.066168202e-005,
         2.163781936e-008,-6.49183541e-012},
  blow={25810.33714,10.1526713},
  ahigh={2292078.842,-10598.32517,22.68612155,-0.002363586598,4.76217307e-007,
      -5.0901004e-011,2.23391159e-015},
  bhigh={86385.91979999999,-124.6794566},
  R=324.1034452649437);

constant IdealGases.Common.DataRecord B2H4_db (
  name="B2H4_db",
  MM=0.02565376,
  Hf=8183301.31723381,
  H0=449394.0849216645,
  Tlimit=1000,
  allow={-17814.96923,571.6048040000001,-2.763085348,0.03125348696,-
2.709524346e-005,
        1.341728634e-008,-3.004352798e-012},
  blow={21581.38307,36.7587294},
  ahigh={1886854.256,-10100.45614,22.47110812,-0.002319968667,4.73187996e-007,
      -5.11067303e-011,2.262693782e-015},
  bhigh={82334.87,-123.7455797},
  R=324.1034452649437);

constant IdealGases.Common.DataRecord B2H5 (
  name="B2H5",
  MM=0.0266617,
  Hf=9556175.67521951,
  H0=458364.9579734226,
  Tlimit=1000,
  allow={38309.6545,-40.6614805,-1.422523787,0.0367459527,-3.85198971e-005,

```

```
 2.44449444e-008,-6.82172294e-012},
blow={30089.56452,28.02425895},
ahigh={2696691.012,-13200.19835,27.35260413,-0.002961879012,5.98499943e-007,
-6.41374855e-011,2.821187823e-015},
bhigh={106452.4901,-155.9654379},
R=311.8507822081863);

constant IdealGases.Common.DataRecord B2H5_db (
  name="B2H5_db",
  MM=0.0266617,
  Hf=10320070.21307719,
  H0=437613.3554874596,
  Tlimit=1000,
  allow={14244.46027,420.45582,-3.70045125,0.036720531,-3.19350373e-005,
  1.695013434e-008,-4.25703814e-012},
  blow={30466.84683,42.45495820000001},
  ahigh={2571729.448,-13568.29858,27.81376038,-0.00319236503,6.56116526e-007,
  -7.12733394e-011,3.16949265e-015},
  bhigh={110637.0527,-159.7214457},
  R=311.8507822081863);

constant IdealGases.Common.DataRecord B2H6 (
  name="B2H6",
  MM=0.02766964,
  Hf=1322749.410545276,
  H0=431242.0761527797,
  Tlimit=1000,
  allow={-10528.44558,1041.795556,-8.960518860000001,0.0549248088,-5.30519705e-
005,
  3.011904756e-008,-7.688768300000001e-012},
  blow={-925.9374349999999,68.12592429999999},
  ahigh={2835765.414,-15676.00163,32.2612233,-0.00373860973,7.71880646e-007,-
8.41445454e-011,
  3.75222338e-015},
  bhigh={93583.7855,-192.0223395},
  R=300.4907906282843);

constant IdealGases.Common.DataRecord B2O (
  name="B2O",
  MM=0.0376214,
  Hf=5124690.149755193,
  H0=313199.8809188387,
  Tlimit=1000,
  allow={-56995.4676,1018.000465,-2.636481947,0.02746009279,-3.63243272e-005,
  2.406890217e-008,-6.38412972e-012},
  blow={17038.74394,38.5534611},
  ahigh={-162872.322,-387.628981,7.78772352,-0.0001146822108,2.528097612e-008,
  -2.893783941e-012,1.339210475e-016},
  bhigh={22630.35354,-19.08513011},
  R=221.0037903958917);

constant IdealGases.Common.DataRecord B2O2 (
  name="B2O2",
  MM=0.0536208,
  Hf=-8536080.886521647,
  H0=249839.9501685913,
  Tlimit=1000,
```

```

alow={81743.9169,-1732.702797,16.05560926,-0.02160057288,3.56685457e-005,-
2.660198794e-008,
    7.531833240000001e-012},
blow={-48996.32900000001,-61.72702390000001},
ahigh={460578.966,-2990.079203,12.56764079,-0.000785097495,1.672537624e-007,
    -1.86769478e-011,8.486190669999999e-016},
bhigh={-40157.4815,-48.7441371},
R=155.0605735087876);

constant IdealGases.Common.DataRecord B203 (
  name="B203",
  MM=0.0696201999999999,
  Hf=-11999136.32824956,
  H0=207105.5383351384,
  Tlimit=1000,
  alow={73796.11910000001,-1263.620592,10.72681512,0.000384138372,
      5.97605838e-006,-6.55289135e-009,2.123951064e-012},
  blow={-96281.83140000001,-30.88078011},
  ahigh={390503.53,-3691.34821,15.55502598,-0.0009707645510000001,
      2.068887872e-007,-2.310858356e-011,1.050136734e-015},
  bhigh={-82630.5441,-63.9086344},
  R=119.4261435617824);

constant IdealGases.Common.DataRecord B2_OH_4 (
  name="B2_OH_4",
  MM=0.08965136,
  Hf=-13998533.10646933,
  H0=216874.2225438633,
  Tlimit=1000,
  alow={-17287.17249,849.591348,-8.715959570000001,0.0936236261,-
0.0001278784165,
      8.81367116e-008,-2.400605571e-011},
  blow={-156433.7133,71.2016931},
  ahigh={2983925.656,-12074.69577,31.6261057,-0.000516368244,-4.23209397e-008,
      1.67197728e-011,-1.156366726e-015},
  bhigh={-82809.7696999999,-165.3010498},
  R=92.74228522579023);

constant IdealGases.Common.DataRecord B2S (
  name="B2S",
  MM=0.053687,
  Hf=11590528.52645892,
  H0=245547.4323393,
  Tlimit=1000,
  alow={66386.0929999999,-920.1185690000001,8.699831339999999,0.000764558901,
      -3.70902603e-006,3.64237398e-009,-1.197766071e-012},
  blow={77703.758,-22.51712444},
  ahigh={-121150.1483,-64.73398450000001,7.54682561,-1.82054145e-005,
      3.92395117e-009,-4.40386238e-013,2.003651718e-017},
  bhigh={72585.3187,-13.94971229},
  R=154.8693724737832);

constant IdealGases.Common.DataRecord B2S2 (
  name="B2S2",
  MM=0.085752,
  Hf=1612989.924433249,
  H0=169898.2181173617,

```

---

**956 Modelica.Media.IdealGases.Common.SingleGasesData**

---

```
Tlimit=1000,
alow={-70615.4997, 718.294478, 1.919797788, 0.02036407461, -2.309813846e-005,
      1.311616725e-008, -2.996394781e-012},
blow={11008.29019, 18.55402586},
ahigh={-164153.5014, -774.286564, 11.07731737, -0.0002313278554,
      5.12561429e-008, -5.89408123e-012, 2.738673834e-016},
bhigh={17330.03515, -33.59766998},
R=96.95951114842804);

constant IdealGases.Common.DataRecord B2S3(
  name="B2S3",
  MM=0.117817,
  Hf=150691.8101801947,
  H0=169229.0331615981,
  Tlimit=1000,
  alow={-34312.685, 558.761537, 3.41407741, 0.02398333891, -2.841617637e-005,
        1.690005199e-008, -4.060070709999999e-012},
  blow={-3027.690575, 18.46767864},
  ahigh={-190585.5086, -934.8342479999999, 13.69333613, -0.000276664725,
        6.1101558e-008, -7.00804857e-012, 3.24952462e-016},
  bhigh={2906.43376, -39.46870479},
  R=70.57107208637125);

constant IdealGases.Common.DataRecord B3H7_C2v(
  name="B3H7_C2v",
  MM=0.03948858,
  Hf=4457469.450661431,
  H0=375258.8469881672,
  Tlimit=1000,
  alow={108600.9469, -792.15368, -0.9577368039999999, 0.0565611929, -6.4050345e-
005,
        4.28132296e-008, -1.227580455e-011},
  blow={24306.38819, 21.02582666},
  ahigh={3598592.14, -18637.01968, 39.9399883, -0.00427541016,
        8.705358820000001e-007, -9.38510502e-011, 4.14786531e-015},
  bhigh={127215.4092, -237.9769856},
  R=210.5538360710869);

constant IdealGases.Common.DataRecord B3H7_Cs(
  name="B3H7_Cs",
  MM=0.03948858,
  Hf=4034541.682683955,
  H0=356642.9078989419,
  Tlimit=1000,
  alow={52981.69330000001, 407.174537, -9.63573053, 0.07944312000000001, -
9.30645681999999e-005,
        6.105147350000001e-008, -1.685927691e-011},
  blow={17070.69083, 68.73966560000001},
  ahigh={3281576.45, -18110.96595, 39.6383246, -0.00417842411, 8.52682906e-007, -
9.20960024e-011,
        4.076496770000001e-015},
  bhigh={121362.2564, -235.7090217},
  R=210.5538360710869);

constant IdealGases.Common.DataRecord B3H9(
  name="B3H9",
  MM=0.04150446,
```

```

Hf=3346840.315474529,
H0=417543.9217857551,
Tlimit=1000,
alow={84832.14900000001,-520.616304,-0.466570121,0.0534698386,-4.15579682e-
005,
      1.944135439e-008,-4.54122735e-012},
blow={18051.011,21.10804198},
ahigh={3863999.5,-22294.33407,48.7504615,-0.00541844926,1.125767651e-006,-
1.233226406e-010,
      5.52034818e-015},
bhigh={142218.7123,-295.8240102},
R=200.3271937521895);

constant IdealGases.Common.DataRecord B3N3H6 (
  name="B3N3H6",
  MM=0.0805007399999999,
  Hf=-6360189.980862289,
  H0=201505.5762220323,
  Tlimit=1000,
  alow={-155262.1082,4010.21127,-33.824913,0.1624575582,-0.0002065528647,
        1.368862991e-007,-3.6734709e-011},
  blow={-80512.534,199.5000751},
  ahigh={3739245.17,-19412.38951,44.4507003,-0.00321030805,5.71746542e-007,-
5.48669214e-011,
        2.191943197e-015},
  bhigh={47584.2616,-264.1156625},
  R=103.2844170128126);

constant IdealGases.Common.DataRecord B3O3CL3 (
  name="B3O3CL3",
  MM=0.1867902,
  Hf=-8758393.325774049,
  H0=130905.0528346776,
  Tlimit=1000,
  alow={-42844.52800000001,1005.623966,-4.55651963,0.0811796713999999,-
0.0001038385803,
        6.62142233e-008,-1.695748504e-011},
  blow={-204090.7277,54.66113803},
  ahigh={-741405.981,-2527.447455,26.87263667,-0.000746338936,1.64633411e-007,
        -1.886231553e-011,8.737926310000001e-016},
  bhigh={-192350.9858,-118.5636255},
  R=44.51235664397812);

constant IdealGases.Common.DataRecord B3O3FCL2 (
  name="B3O3FCL2",
  MM=0.1703356032,
  Hf=-11059388.90995162,
  H0=137658.4728001245,
  Tlimit=1000,
  alow={-37191.2007,867.3745889999999,-4.09577846,0.0781556397,-
9.76703516999999e-005,
        6.09761009e-008,-1.532880319e-011},
  blow={-233138.3552,51.98088158},
  ahigh={-716236.083,-2851.210308,27.10953133,-0.000840091964,
        1.852318684e-007,-2.121694694e-011,9.827291560000002e-016},
  bhigh={-220272.9252,-121.0679365},
  R=48.81229668842362);

```

```
constant IdealGases.Common.DataRecord B3O3F2CL (
  name="B3O3F2CL",
  MM=0.1538810064,
  Hf=-13860169.47053188,
  H0=144663.6691609264,
  Tlimit=1000,
  allow={-52770.1554,1104.904696,-5.85418105,0.07953943099999999,-9.59266077e-005,
         5.79985631e-008,-1.416698348e-011},
  blow={-264040.0115,60.71247222},
  ahighelement={-681078.749,-3370.00902,27.48953579,-0.000990544256,2.18290844e-007,
               -2.49957348e-011,1.157532106e-015},
  bhigh={-247211.4785,-126.1501548},
  R=54.03182754333741);

constant IdealGases.Common.DataRecord B3O3F3 (
  name="B3O3F3",
  MM=0.1374264096,
  Hf=-17337999.9152652,
  H0=154269.7510741051,
  Tlimit=1000,
  allow={-72515.4978,1400.996228,-7.9067162,0.0821716185,-9.63315266e-005,
         5.66819426e-008,-1.349024305e-011},
  blow={-295346.1728,70.0153683000001},
  ahighelement={-662982.291,-3784.94687,27.79526578,-0.001112096232,2.450806704e-007,
               -2.80648498e-011,1.299750466e-015},
  bhigh={-274886.5475,-131.5605015},
  R=60.50126772721858);

constant IdealGases.Common.DataRecord B4H4 (
  name="B4H4",
  MM=0.04727576,
  Hf=6899738.893674052,
  H0=315946.3115981636,
  Tlimit=1000,
  allow={217179.1579,-2598.282557,9.84544061000001,0.0302630914,-
        4.128334159999999e-005,
         3.089368344e-008,-9.35338426e-012},
  blow={50791.4348,-37.7075721},
  ahighelement={1937200.886,-10781.85932,28.79625241,-0.002401547707,4.83775678e-007,
               -5.17049023e-011,2.269138182e-015},
  bhigh={98365.95389999999,-156.1814117},
  R=175.8717786874288);

constant IdealGases.Common.DataRecord B4H10 (
  name="B4H10",
  MM=0.0533234,
  Hf=1239605.876594516,
  H0=290408.6011019553,
  Tlimit=1000,
  allow={-32966.0482000001,2440.532922,-26.85332563,0.1406499347,-
        0.0001586710574,
         9.87459413e-008,-2.601426018e-011},
  blow={-3091.924018,158.9905532},
  ahighelement={4030474.13,-25606.29762,56.8723366,-0.00617856786,1.280621032e-006,-
               1.400254585e-010,
               6.25886656000001e-015},
  bhigh={150600.9087,-352.07142},
```

```

R=155.9253911040931);

constant IdealGases.Common.DataRecord B4H12 (
  name="B4H12",
  MM=0.05533928000000001,
  Hf=3401491.924000457,
  H0=393006.8117980573,
  Tlimit=1000,
  allow={71521.70299999999,-159.0351575,-4.71977166,0.088112289,-8.60494688e-
005,
         5.12353551e-008,-1.37986525e-011},
  blow={21941.86547,41.878974},
  ahigh={6158360.8,-32020.8673,66.60443599999999,-0.00740636928,
         1.512993908e-006,-1.635619071e-010,7.24538514e-015},
  bhigh={205458.6275,-414.210686},
  R=150.2453953141421);

constant IdealGases.Common.DataRecord B5H9 (
  name="B5H9",
  MM=0.06312646,
  Hf=1159893.965224725,
  H0=254293.9521715617,
  Tlimit=1000,
  allow={89984.7372,104.2533357,-11.95549834,0.1007310686,-9.866680210000001e-
005,
         5.29553518e-008,-1.232119583e-011},
  blow={8374.341780000001,76.6309048},
  ahigh={3169017.62,-22825.03203,55.1200898,-0.005563411420000001,
         1.157837073e-006,-1.270332301e-010,5.69426203e-015},
  bhigh={133445.2571,-337.106046},
  R=131.7113616065276);

constant IdealGases.Common.DataRecord Ba (
  name="Ba",
  MM=0.137327,
  Hf=1347149.504467439,
  H0=45128.98410363585,
  Tlimit=1000,
  allow={2222.563526,-34.0797785,2.706751118,-0.0006382894490000001,
         1.063003846e-006,-9.10262427e-010,3.148062219e-013},
  blow={21665.49702,5.10254588},
  ahigh={-19265792.28,60065.0104,-66.3396413,0.0350756593,-7.80760183e-006,
         8.0851268e-010,-3.199486918e-014},
  bhigh={-358966.372,500.75834},
  R=60.54506397139674);

constant IdealGases.Common.DataRecord Baplus (
  name="Baplus",
  MM=0.1373264514,
  Hf=5054011.961456684,
  H0=45129.16438762649,
  Tlimit=1000,
  allow={-54231.6755,674.468078,-0.8940272500000001,0.008796642950000001,-
1.222812901e-005,
         8.38795387000001e-009,-2.037964358e-012},
  blow={79417.7464,26.07063698},
  ahigh={8794971.85,-19518.17883,14.85542861,-0.00094042335,-7.03125779e-007,
         1.222812901e-005,
         8.38795387000001e-009,-2.037964358e-012});

```

```
    1.667412753e-010,-1.07011731e-014},  
bhigh={214680.0732,-92.28264190000002},  
R=60.54530584047409);  
  
constant IdealGases.Common.DataRecord BaBr(  
  name="BaBr",  
  MM=0.217231,  
  Hf=-346749.2070652899,  
  H0=46955.19055751711,  
  Tlimit=1000,  
  allow={-691.2557439999999,-62.4550016,4.79667765,-0.000683800494,  
         1.04394768e-006,-7.4412243e-010,2.172187003e-013},  
  blow={-10113.49165,5.19645487},  
  ahigh={-1056014.042,1034.740207,6.73572938,-0.0037840263,1.952196896e-006,-  
3.34783188e-010,  
         1.893189803e-014},  
  bhigh={-19502.33645,-5.06927964},  
  R=38.27479503385797);  
  
constant IdealGases.Common.DataRecord BaBr2(  
  name="BaBr2",  
  MM=0.297135,  
  Hf=-1388307.473034143,  
  H0=52116.91318760833,  
  Tlimit=1000,  
  allow={-5143.37406,-38.7051846,7.16244697,-0.000369671531,4.68971527e-007,-  
3.107902475e-010,  
         8.36358205e-014},  
  blow={-51533.1826,0.974140700000001},  
  ahigh={-8928.25121,-0.753655889,7.00065821,-2.94982171e-007,  
         7.092392109999999e-011,-8.67003809999999e-015,4.22174231e-019},  
  bhigh={-51726.6949,1.910185706},  
  R=27.98213606609791);  
  
constant IdealGases.Common.DataRecord BaCL(  
  name="BaCL",  
  MM=0.17278,  
  Hf=-788811.6738048386,  
  H0=57165.86989234866,  
  Tlimit=1000,  
  allow={-2661.946867,-77.2499634,4.74610605,-0.000352354888,3.99919835e-007,-  
1.936413749e-010,  
         4.124048529999999e-014},  
  blow={-17363.28141,3.904455084},  
  ahigh={-1075410.089,1209.303069,6.2725586,-0.00328811031,1.717751087e-006,-  
2.910607065e-010,  
         1.632658438e-014},  
  bhigh={-27791.73737,-3.574185216},  
  R=48.12172705174211);  
  
constant IdealGases.Common.DataRecord BaClplus(  
  name="BaClplus",  
  MM=0.1727794514,  
  Hf=2018166.090785492,  
  H0=55957.05925479053,  
  Tlimit=1000,  
  allow={-895.155669,-133.4692684,4.93541612,-0.000757405451,8.355724e-007,-
```

```

4.68525851e-010,
  1.095343103e-013},
blow={41251.5599,1.896278378},
ahigh={-18291.98157,-2.931616688,4.50228071,3.67178849e-005,
  2.396147871e-009,5.4794586e-014,1.174342316e-018},
bhigh={40551.00150000001,4.503765564},
R=48.12187984525572);

constant IdealGases.Common.DataRecord BaCL2 (
  name="BaCL2",
  MM=0.208233,
  Hf=-2397801.438772913,
  H0=70109.50713863796,
  Tlimit=1000,
  alow={-1157.501999,-205.4518828,7.83751405,-0.001867700704,2.335378734e-006,
    -1.531373341e-009,4.08852509e-013},
  blow={-61156.6528,-5.978149218},
  ahigh={-22165.98768,-4.00248381,7.00342674,-1.515820808e-006,
    3.61184921e-010,-4.38675906e-014,2.125803293e-018},
  bhigh={-62190.9527,-1.132439694},
  R=39.92869525963705);

constant IdealGases.Common.DataRecord BaF (
  name="BaF",
  MM=0.1563254032,
  Hf=-2040575.501295109,
  H0=59760.67746358449,
  Tlimit=1000,
  alow={25303.17292,-471.69951,6.18503622,-0.0033666788,
    4.011834559999999e-006,-2.506578601e-009,6.47725816e-013},
  blow={-37318.7214,-6.22071538},
  ahigh={1868380.434,-7071.099600000001,14.66959284,-0.0069959502,
    2.366313845e-006,-3.2877343e-010,1.647896238e-014},
  bhigh={3571.78068,-66.6239092},
  R=53.18695381429856);

constant IdealGases.Common.DataRecord BaFplus (
  name="BaFplus",
  MM=0.1563248546,
  Hf=857593.7226555399,
  H0=58862.77024562158,
  Tlimit=1000,
  alow={28407.19922,-486.357182,5.94605839,-0.002429844077,2.445346729e-006,-
1.316757471e-009,
  2.97488632e-013},
  blow={17306.39076,-5.95120693},
  ahigh={-40544.2367,-17.01997704,4.51305534,2.632753508e-005,
    2.612088236e-009,-9.502197929999999e-014,6.515176160000001e-018},
  bhigh={14744.8657,2.783167711},
  R=53.18714046640156);

constant IdealGases.Common.DataRecord BaF2 (
  name="BaF2",
  MM=0.1753238064,
  Hf=-4631450.227286418,
  H0=76641.14346994921,
  Tlimit=1000,

```

```
alow={34883.9552,-764.845956,9.79775038,-0.00575645751,6.7756038e-006,-
4.24361475e-009,
    1.093687692e-012},
blow={-95903.5644999999,-20.82035292},
ahigh={-55086.3366,-16.91114228,7.01356488,-5.73190977e-006,
    1.321203959e-009,-1.565528167e-013,7.44498301999999e-018},
bhigh={-99835.29700000001,-4.38958149},
R=47.42351977592018);

constant IdealGases.Common.DataRecord BaH(
  name="BaH",
  MM=0.13833494,
  Hf=1514694.010059932,
  H0=63115.68863224288,
  Tlimit=1000,
  alow={-37652.6888,698.668012,-1.313524351,0.01460648882,-1.802974829e-005,
    1.133031115e-008,-2.865589059e-012},
  blow={20974.78889,32.308026},
  ahigh={-6755466.57,17307.00212,-11.36812458,0.00561774474,-3.18078885e-007,
    -1.052342502e-010,1.111101879e-014},
  bhigh={-89540.70379999999,118.2377538},
  R=60.10391879303957);

constant IdealGases.Common.DataRecord BaI(
  name="BaI",
  MM=0.26423147,
  Hf=-38745.10102827645,
  H0=39348.54920952451,
  Tlimit=1000,
  alow={-4186.28554,10.33094613,4.42890023,0.0003026019976,-2.591429408e-007,
    1.7416054e-010,-3.370062760000001e-014},
  blow={-2636.169733,8.201924679999999},
  ahigh={-3533859.66,7394.258,1.285151488,-0.002303429302,2.214839289e-006,-
4.68109681e-010,
    3.011371472e-014},
  bhigh={-53402.2959,37.1044387},
  R=31.46662280613282);

constant IdealGases.Common.DataRecord BaI2(
  name="BaI2",
  MM=0.39113594,
  Hf=-737441.7318950542,
  H0=40370.70845496836,
  Tlimit=1000,
  alow={-4241.592729999999,-18.62214382,7.07855675,-0.0001793922821,
    2.281437677e-007,-1.514647881e-010,4.08149836e-014},
  blow={-36703.5149,3.16313897},
  ahigh={-6048.2248,-0.362272945,7.00031745,-1.425793502e-007,3.43334942e-011,
    -4.20175497999999e-015,2.047703474e-019},
  bhigh={-36796.5075,3.61547532},
  R=21.25724370918203);

constant IdealGases.Common.DataRecord BaO(
  name="BaO",
  MM=0.1533264,
  Hf=-769263.4275636812,
  H0=58790.31921443405,
```

```

Tlimit=1000,
alow={37643.985,-507.15553,5.39284728,-0.000411932469,-6.52789662e-007,
      9.43058841999999e-010,-3.43660303e-013},
blow={-12755.52614,-3.75221376},
ahigh={13184816.98,-38542.5325,46.8674161,-0.02188646633,
       5.335677450000001e-006,-5.1523043e-010,1.4330834e-014},
bhigh={230690.2396,-302.8332772},
R=54.22726940696449;

constant IdealGases.Common.DataRecord BaOplus(
  name="BaOplus",
  MM=0.1533258514,
  Hf=3337367.321477009,
  H0=61762.48762705387,
  Tlimit=1000,
  alow={356641.097,-3779.24671,16.73544137,-0.0130661161,2.686157445e-006,
        4.044077950000001e-009,-2.016508813e-012},
  blow={79832.6849000001,-72.8511855},
  ahigh={-331767.923,1594.253495,3.22724781,0.00057226415,-1.147415658e-007,
         1.382873231e-011,-6.38443703e-016},
  bhigh={50252.4761,14.09092365},
  R=54.22746343217109);

constant IdealGases.Common.DataRecord BaOH(
  name="BaOH",
  MM=0.15433434,
  Hf=-1453056.481143471,
  H0=72728.99213486773,
  Tlimit=1000,
  alow={37762.3151,-889.139022,9.531718,-0.0054588259,4.94178301e-006,-
1.809327537e-009,
        2.050048305e-013},
  blow={-24418.52965,-24.89410158},
  ahigh={2637336.694,-8365.410400000001,15.99803975,-0.005098446049999999,
         1.594657431e-006,-2.180478542e-010,1.084610688e-014},
  bhigh={23935.50647,-74.9070579},
  R=53.87311728549849);

constant IdealGases.Common.DataRecord BaOHplus(
  name="BaOHplus",
  MM=0.1543337914,
  Hf=1308976.544717996,
  H0=73237.2340332462,
  Tlimit=1000,
  alow={27976.32165,-768.07694,9.05297844,-0.0043685779,3.52349726e-006,-
8.38636087e-010,
        -6.47545747e-014},
  blow={26232.80802,-22.64561641},
  ahigh={876673.843,-2335.8196,7.97316606,0.0001038679898,-6.31948578e-008,
         1.028729445e-011,-5.74180857e-016},
  bhigh={37725.1747,-19.30805001},
  R=53.87330878466322);

constant IdealGases.Common.DataRecord Ba_OH_2(
  name="Ba_OH_2",
  MM=0.17134168,
  Hf=-3540682.407222807,

```

```
H0=101485.6688693609,
Tlimit=1000,
alow={57028.149,-1577.384797,16.59935325,-0.01039598275,
      9.664889600000001e-006,-3.67362144e-009,4.62578499e-013},
blow={-68351.58899999999,-58.51911699999999},
ahigh={1762908.112,-4676.205419999999,13.95135494,0.0002051842928,-
1.25726553e-007,
      2.048917625e-011,-1.144045777e-015},
bhigh={-45459.6441,-49.2945719},
R=48.52568271771352);

constant IdealGases.Common.DataRecord BaS (
  name="BaS",
  MM=0.169392,
  Hf=229475.5065174271,
  H0=56414.20492112969,
  Tlimit=1000,
  alow={13770.75147,-329.350056,5.81154584,-0.00284897376,3.57633602e-006,-
2.325649347e-009,
      6.17280531e-013},
  blow={4964.43245,-3.497962615},
  ahigh={6104262.07,-22276.17479,35.9751351,-0.0215779374,7.31987235e-006,-
1.089038636e-009,
      5.88410195e-014},
  bhigh={140413.8006,-214.1776331},
  R=49.08420704637764);

constant IdealGases.Common.DataRecord Ba2 (
  name="Ba2",
  MM=0.274654,
  Hf=1296046.432966569,
  H0=41357.9885965615,
  Tlimit=1000,
  alow={-12941.77083,-727.633008,14.64866569,-0.0396633165,5.89871106e-005,-
4.23530056e-008,
      1.189594362e-011},
  blow={43867.1223,-41.4567367},
  ahigh={216011.4547,195.0676877,2.253699771,0.0001507750613,-4.742515e-008,
      7.04895196e-012,-3.54744017e-016},
  bhigh={41677.6297,23.75906459},
  R=30.27253198569837);

constant IdealGases.Common.DataRecord Be (
  name="Be",
  MM=0.00901218199999999,
  Hf=35951337.86690061,
  H0=687672.3084376237,
  Tlimit=1000,
  alow={-0.000411290152,5.36496736e-006,2.49999972,7.56920369e-011,-
1.097852652e-013,
      8.00211024e-017,-2.303022777e-020},
  blow={38222.6459,2.146172983},
  ahigh={-692628.584,2466.773005,-0.9776613340000001,0.002458939515,-
9.04795041999999e-007,
      1.587880407e-010,-9.415600603e-015},
  bhigh={23002.12917,26.23234754},
  R=922.581456965694);
```

```

constant IdealGases.Common.DataRecord Beplus(
  name="Beplus",
  MM=0.0090116334,
  Hf=136457096.4460228,
  H0=687714.1717726777,
  Tlimit=1000,
  allow={0,0,2.5,0,0,0,0},
  blow={147152.8569,2.839228698},
  ahigh={-94781.35979999999,276.8325398,2.191388413,0.0001648824289,-
4.28016682e-008,
        4.54235047e-012,-6.270417825e-017},
  bhigh={145385.0986,5.05550384},
  R=922.6376208335329);

constant IdealGases.Common.DataRecord Beplusplus(
  name="Beplusplus",
  MM=0.009011084800000001,
  Hf=332146638.7709501,
  H0=687756.0402050593,
  Tlimit=1000,
  allow={0,0,2.5,0,0,0,0},
  blow={359227.916,2.145990203},
  ahigh={0,0,2.5,0,0,0,0},
  bhigh={359227.916,2.145990203},
  R=922.6937915399486);

constant IdealGases.Common.DataRecord BeBr(
  name="BeBr",
  MM=0.08891618200000001,
  Hf=1489563.598221075,
  H0=100893.9857539092,
  Tlimit=1000,
  allow={37462.8806,-472.233907,5.03940904,0.0005798853219999999,-1.937634343e-
006,
        1.793372625e-009,-5.67229558e-013},
  blow={17231.35679,-2.573912886},
  ahigh={821723.8139999999,-2850.509348,8.017266019999999,-0.002105663212,
        6.84913634e-007,-9.98912932e-011,5.192456840000001e-015},
  bhigh={32293.7287,-23.26810811},
  R=93.50909826515043);

constant IdealGases.Common.DataRecord BeBr2(
  name="BeBr2",
  MM=0.168820182,
  Hf=-1386460.411469051,
  H0=76042.45445014389,
  Tlimit=1000,
  allow={-21186.98678,209.0382611,3.73117131,0.01115501549,-1.542821761e-005,
        1.052354444e-008,-2.850006057e-012},
  blow={-30904.56277,9.458917720000001},
  ahigh={-104133.1811,-154.3443092,7.61446541,-4.55670063e-005,
        1.003180548e-008,-1.146879501e-012,5.30179538e-017},
  bhigh={-29845.27608,-11.4936098},
  R=49.25046224627338);

constant IdealGases.Common.DataRecord BeCL(
  name="BeCL",

```

```
MM=0.044465182,
Hf=1274999.998875525,
H0=199281.9235508808,
Tlimit=1000,
alow={20161.38947,-141.3988978,2.898223773,0.00606992551,-
9.162038470000001e-006,
       6.61262066e-009,-1.859126853e-012},
blow={6626.691440000001,7.90018105},
ahigh={280884.5146,-1103.100122,5.67118983,-0.0005443000450000001,
       1.459036606e-007,-1.230971735e-011,9.90080476e-017},
bhigh={12246.49804,-8.38237137},
R=186.9883721604918);

constant IdealGases.Common.DataRecord BeCL2 (
  name="BeCL2",
  MM=0.079918182,
  Hf=-4523865.920773824,
  H0=151165.3505831752,
  Tlimit=1000,
  alow={-22299.57657,152.8781667,3.71499471,0.0106777655,-1.397766942e-005,
         9.066693559999998e-009,-2.350292311e-012},
  blow={-45904.2823,6.682555572},
  ahigh={-115408.746,-248.4300053,7.68567214,-7.44496037e-005,
         1.649479787e-008,-1.895997075e-012,8.804973950000001e-017},
  bhigh={-44687.4728,-15.00337407},
  R=104.0373015492269);

constant IdealGases.Common.DataRecord BeF (
  name="BeF",
  MM=0.0280105852,
  Hf=-6091439.853245195,
  H0=310993.3240523657,
  Tlimit=1000,
  alow={-46644.9723,788.717885,-1.463363201,0.01381790494,-1.591299964e-005,
         9.35293152e-009,-2.229211181e-012},
  blow={-25226.27476,31.976343},
  ahigh={-185794.1543,47.76082770000001,4.26998602,0.0002310668473,-
6.657214920000001e-008,
       1.152149888e-011,-6.33558644e-016},
  bhigh={-22579.5096,-0.218478727},
  R=296.8332128955307);

constant IdealGases.Common.DataRecord BeF2 (
  name="BeF2",
  MM=0.0470089884,
  Hf=-16945445.18213883,
  H0=231440.0792338684,
  Tlimit=1000,
  alow={7490.27024,-287.8053102,5.43471282,0.00390323346,-2.214865601e-006,-
1.965806119e-010,
       4.04236227e-013},
  blow={-95916.15640000001,-5.61661692},
  ahigh={-64820.1807,-768.455606,8.062860840000001,-0.0002227502549,
       4.89159866e-008,-5.58788672e-012,2.583409761e-016},
  bhigh={-93954.9549,-21.25660266},
  R=176.8698345357289);
```

```

constant IdealGases.Common.DataRecord BeH (
  name="BeH",
  MM=0.010020122,
  Hf=34156480.72947615,
  H0=863073.7230544697,
  Tlimit=1000,
  allow={-1615.149125,-59.3528608,4.5072691,-0.00526418961,1.145319967e-005,-
9.247883590000001e-009,
  2.700364753e-012},
  blow={40301.93040000001,-3.48511705},
  ahigh={-2424081.636,6597.39846,-3.62631648,0.00457963435,-1.163329056e-006,
  1.297785691e-010,-5.29188581e-015},
  bhigh={-2489.846221,52.1601869},
  R=829.7775216708938);

constant IdealGases.Common.DataRecord BeHplus (
  name="BeHplus",
  MM=0.0100195734,
  Hf=117591728.1069072,
  H0=862706.4900787092,
  Tlimit=1000,
  allow={-32206.1102,252.0229596,3.20771957,-0.00219128729,6.90253449e-006,-
5.76952263e-009,
  1.656414969e-012},
  blow={139253.066,3.30704477},
  ahigh={462234.1589999999,-1760.169629,5.37990555,2.742462444e-005,-
1.16720817e-007,
  3.55717791e-011,-2.569695765e-015},
  bhigh={151585.9734,-13.69222641},
  R=829.822954338555);

constant IdealGases.Common.DataRecord BeH2 (
  name="BeH2",
  MM=0.011028062,
  Hf=14608088.25703011,
  H0=835967.6432722267,
  Tlimit=1000,
  allow={90653.14900000001,-1320.65722,9.90130806,-0.01255772231,
  2.205704068e-005,-1.633609388e-008,4.50510324e-012},
  blow={24645.61869,-36.3582675},
  ahigh={607426.201,-3407.53425,9.79237036,-0.000825461482,1.915656246e-007,-
1.962774813e-011,
  8.929908660000001e-016},
  bhigh={37931.052,-42.5465358},
  R=753.9377272271411);

constant IdealGases.Common.DataRecord BeI (
  name="BeI",
  MM=0.135916652,
  Hf=1526334.749622879,
  H0=66872.66693414432,
  Tlimit=1000,
  allow={35300.5481,-519.095026,5.71450524,-0.00140907364,7.66477212e-007,-
2.601945105e-011,
  -8.23096063999999e-014},
  blow={26379.03657,-5.14099784},
  ahigh={-532820.662,1139.497861,3.62917166,0.0001704033769,1.067048688e-007,
  -3.58945788e-011,2.72276956e-015},

```

## 968 Modelica.Media.IdealGases.Common.SingleGasesData

---

```
bhigh={15854.74307, 9.370951270000001},  
R=61.17331377468009);  
  
constant IdealGases.Common.DataRecord BeI2(  
  name="BeI2",  
  MM=0.262821122,  
  Hf=-246401.2424389543,  
  H0=51139.53131970877,  
  Tlimit=1000,  
  allow={359.272618,-48.6605901,5.24157118,0.00783030505999999,-1.169525374e-  
005,  
     8.38958391000001e-009,-2.356636003e-012},  
  blow={-9333.22092,2.896318281},  
  ahigh={-93923.72840000001,-89.5921438,7.56555479,-2.577727326e-005,  
     5.61426533e-009,-6.35966385e-013,2.917080336e-017},  
  bhigh={-9814.768620000001,-9.03204281},  
  R=31.63547867359002);  
  
constant IdealGases.Common.DataRecord BeN(  
  name="BeN",  
  MM=0.023018882,  
  Hf=18549988.65713808,  
  H0=379046.2542881101,  
  Tlimit=1000,  
  allow={-42477.8363,759.0188900000001,-1.561469235,0.01496945721,-1.83560888e-  
005,  
     1.146861461e-008,-2.897470581e-012},  
  blow={46830.0332,32.56973759},  
  ahigh={-59096.9616,-307.8649431,4.72795915,-3.92815883e-005,  
     2.006201069e-008,-2.300018068e-012,1.066101831e-016},  
  bhigh={51560.9571,-3.031380625},  
  R=361.2022512648529);  
  
constant IdealGases.Common.DataRecord BeO(  
  name="BeO",  
  MM=0.025011582,  
  Hf=5155223.847895747,  
  H0=347363.2335611557,  
  Tlimit=1000,  
  allow={-48694.5831999999,721.347362,-0.378415732,0.00879220384999999,-  
7.07015126e-006,  
     2.25017562e-009,-2.799117872e-014},  
  blow={11014.66639,25.74081184},  
  ahigh={-35034724.5,105585.9473,-116.5011722,0.0647736948,-1.649656058e-005,  
     2.0055584e-009,-9.426891790000001e-014},  
  bhigh={-657001.8000000001,864.0898030000001},  
  R=332.4248742042786);  
  
constant IdealGases.Common.DataRecord BeOH(  
  name="BeOH",  
  MM=0.026019522,  
  Hf=-3832419.404168916,  
  H0=415013.38879323,  
  Tlimit=1000,  
  allow={-38530.2452,449.805928999999,1.532096459,0.01299912603,-1.708979996e-  
005,  
     1.169968182e-008,-3.16318371e-012},
```

```

blow={-15590.46567,15.4554293},
ahigh={855033.031,-2646.537838,8.204784630000001,1.110814513e-005,-
4.265364e-008,
    7.926511920000001e-012,-4.64541952e-016},
bhigh={3121.216271,-25.71641308},
R=319.5474536388486);

constant IdealGases.Common.DataRecord BeOHplus(
  name="BeOHplus",
  MM=0.0260189734,
  Hf=29208835.73369579,
  H0=367074.5902680389,
  Tlimit=1000,
  allow={71061.19160000001,-806.8981600000001,5.60310781,0.004932070859999999,
      -7.72284715999999e-006,5.91088980999999e-009,-1.695201901e-012},
  blow={94407.8242,-10.88099852},
  ahigh={813487.439,-2814.653683,8.381145009999999,-6.94207328e-005,-
2.369535806e-008,
    5.67810638e-012,-3.58325356e-016},
  bhigh={107312.9932,-29.01093335},
  R=319.5541911734304);

constant IdealGases.Common.DataRecord Be_OH_2(
  name="Be_OH_2",
  MM=0.043026862,
  Hf=-14848036.46614991,
  H0=353868.6600012801,
  Tlimit=1000,
  allow={7863.08983,-739.7142249999999,10.43303334,0.00451424953,-6.95060153e-
006,
      5.77703004e-009,-1.727199382e-012},
  blow={-75856.7696,-33.0146661},
  ahigh={1722889.403,-5242.47719,14.86877785,3.93149836e-005,-8.92027631e-008,
      1.630912925e-011,-9.50537113999999e-016},
  bhigh={-46428.2384,-64.7680402},
  R=193.2390979383995);

constant IdealGases.Common.DataRecord BeS(
  name="BeS",
  MM=0.041077182,
  Hf=6015390.880513663,
  H0=213746.5515526357,
  Tlimit=1000,
  allow={-7390.57908,282.8134792,0.65520194,0.01076012495,-1.398380157e-005,
      8.81841889e-009,-2.110588578e-012},
  blow={27515.93294,19.80493561},
  ahigh={-16557083.61,54314.6612,-63.5185925,0.0396409199,-1.078978696e-005,
      1.394030343e-009,-6.93695504e-014},
  bhigh={-312155.9628,480.3454935},
  R=202.4109638290182);

constant IdealGases.Common.DataRecord Be2(
  name="Be2",
  MM=0.018024364,
  Hf=35371196.34290564,
  H0=545823.3089389452,
  Tlimit=1000,

```

---

**970 Modelica.Media.IdealGases.Common.SingleGasesData**

---

```
alow={-158087.381,2035.567774,-3.69158063,0.01028818435,-9.60385451e-006,
      4.707519419999999e-009,-9.374688790000001e-013},
blow={65269.68369999999,48.7932499},
ahigh={101003.4484,141.4207488,2.292070182,0.0001543331824,-5.801043e-008,
       1.003129697e-011,-5.68281834e-016},
bhigh={75501.3973,12.38758616},
R=461.290728482847);

constant IdealGases.Common.DataRecord Be2CL4 (
  name="Be2CL4",
  MM=0.159836364,
  Hf=-5127775.385330963,
  H0=141653.5225989,
  Tlimit=1000,
  alow={121615.2589,-2000.824823,17.23384275,0.00581044828,-1.362970089e-005,
         1.156602362e-008,-3.5522558e-012},
  blow={-92064.75470000001,-61.37959773999999},
  ahigh={-335492.008,-231.6199283,16.17047912,-6.73440595e-005,
         1.472073032e-008,-1.672352806e-012,7.68877397e-017},
  bhigh={-103107.5197,-50.47009374},
  R=52.01865077461346);

constant IdealGases.Common.DataRecord Be2F4 (
  name="Be2F4",
  MM=0.0940179768,
  Hf=-18418816.28322808,
  H0=210965.0587588479,
  Tlimit=1000,
  alow={-6327.70187,-69.0253372,5.47093723,0.03097341347,-4.09431006e-005,
         2.668113685e-008,-6.942909560000001e-012},
  blow={-210597.9269,-0.2073546195},
  ahigh={-368183.495,-943.919148,16.70099855,-0.0002797974015,
         6.178061719999999e-008,-7.08308019e-012,3.28279104e-016},
  bhigh={-208904.5725,-60.9266576},
  R=88.43491726786446);

constant IdealGases.Common.DataRecord Be2O (
  name="Be2O",
  MM=0.034023764,
  Hf=-1088468.724389224,
  H0=328899.1188629218,
  Tlimit=1000,
  alow={-426.367357,-115.9556261,4.08267777,0.01015885064,-1.326015538e-005,
         8.48895536e-009,-2.168039077e-012},
  blow={-5362.25812,1.253737886},
  ahigh={-141264.6378,-349.864459,7.76093268,-0.0001044953518,
         2.313342334e-008,-2.657722263e-012,1.233820439e-016},
  bhigh={-5172.38549,-18.54887619},
  R=244.3724921205073);

constant IdealGases.Common.DataRecord Be2OF2 (
  name="Be2OF2",
  MM=0.0720205704,
  Hf=-16725415.99309522,
  H0=230279.6813172699,
  Tlimit=1000,
  alow={-27834.91921,178.2881461,4.09425136,0.02270142496,-2.672688189e-005,
```

```

  1.572575618e-008,-3.73607058e-012},
blow={-148008.1903,7.35349477},
ahigh={-228182.6506,-1130.355396,13.83400563,-0.000331608064,
  7.30492101e-008,-8.36270438e-012,3.87218498e-016},
bhigh={-143139.0587,-47.3915023},
R=115.4457949141708);

constant IdealGases.Common.DataRecord Be202 (
  name="Be202",
  MM=0.050023164,
  Hf=-8228893.898034918,
  H0=218477.4237791116,
  Tlimit=1000,
  allow={-30317.1953,1163.353177,-8.73315595,0.0537200293,-7.27943919e-005,
    4.89492825e-008,-1.312147056e-011},
  blow={-55469.3688,69.53282640000001},
  ahigh={-396057.6640000001,-907.4948830000001,10.67160422,-0.0002670905218,
    5.87762999e-008,-6.71866409e-012,3.105949075e-016},
  bhigh={-48611.1433,-36.1654689},
  R=166.2124371021393);

constant IdealGases.Common.DataRecord Be303 (
  name="Be303",
  MM=0.075034746,
  Hf=-13643287.71100258,
  H0=201583.0506043161,
  Tlimit=1000,
  allow={-27380.72346,874.38326,-6.59960612,0.0653302114,-8.762305740000001e-
005,
    5.81724479e-008,-1.5406005e-011},
  blow={-128467.9012,58.3720312},
  ahigh={-589080.936,-1392.823588,17.03450699,-0.000412733074,9.10752915e-008,
    -1.043465231e-011,4.83305377e-016},
  bhigh={-121904.5534,-70.0061451},
  R=110.8082914014262);

constant IdealGases.Common.DataRecord Be404 (
  name="Be404",
  MM=0.100046328,
  Hf=-16485308.50627521,
  H0=158265.6686810135,
  Tlimit=1000,
  allow={-68932.5564,1687.565069,-14.79616594,0.1007319774,-0.0001291057622,
    8.243136760000001e-008,-2.111283987e-011},
  blow={-207287.7151,100.2507262},
  ahigh={-884133.8459999999,-2726.394165,24.02941774,-0.0008114675460000001,
    1.794255972e-007,-2.059359498e-011,9.552905330000001e-016},
  bhigh={-192371.0587,-113.662075},
  R=83.10621855106966);

constant IdealGases.Common.DataRecord Br (
  name="Br",
  MM=0.079904,
  Hf=1400055.066079295,
  H0=77560.92310772926,
  Tlimit=1000,
  allow={-3700.29391,61.4521542,2.092120721,0.00137681887,-2.445566658e-006,
    1.572575618e-008,-3.73607058e-012});

```

```
    2.050975161e-009,-5.144249091e-013},
    blow={12425.08647,8.996166199999999},
    ahigh={-4789717.399999999,16920.51999,-20.24085357,0.01395620355,-
3.65623056e-006,
        4.489781e-010,-2.122507526e-014},
    bhigh={-92070.54960000001,166.1695929},
    R=104.0557669203044);

constant IdealGases.Common.DataRecord Brplus(
  name="Brplus",
  MM=0.0799034513999999,
  Hf=15743087.58832913,
  H0=77561.53071505495,
  Tlimit=1000,
  allow={39811.8742,-475.158147,4.73457898,-0.00516932479,5.90585752e-006,-
2.885129283e-009,
        5.054894290000001e-013},
  blow={152905.2046,-5.76928322},
  ahigh={1741149.829,-5455.46567,8.470663350000001,-0.002683465574,
        6.59045456e-007,-7.9537703e-011,3.735898818e-015},
  bhigh={185178.1643,-36.3790903},
  R=104.0564813449348);

constant IdealGases.Common.DataRecord Brminus(
  name="Brminus",
  MM=0.0799045486,
  Hf=-2740776.036371977,
  H0=77560.39059833948,
  Tlimit=1000,
  allow={0,0,2.5,0,0,0,0},
  blow={-27084.92743,5.41955617},
  ahigh={0,0,2.5,0,0,0,0},
  bhigh={-27084.92743,5.41955617},
  R=104.055052505484);

constant IdealGases.Common.DataRecord BrCL(
  name="BrCL",
  MM=0.115357,
  Hf=128202.9959170228,
  H0=81547.82978059414,
  Tlimit=1000,
  allow={16532.91583,-357.876928,5.70062216,-0.002201257188,2.415146969e-006,-
1.371032976e-009,
        3.22780342e-013},
  blow={2252.619279,-4.155659669},
  ahigh={-128486.3067,-1138.379626,7.9640408,-0.00343805667,1.524999514e-006,
        -2.761469114e-010,1.694450407e-014},
  bhigh={5938.26426000001,-19.25044389},
  R=72.07600752446753);

constant IdealGases.Common.DataRecord BrF(
  name="BrF",
  MM=0.0989024031999999,
  Hf=-595045.6520352784,
  H0=91212.22243465163,
  Tlimit=1000,
  allow={38361.4319,-518.461197,5.45832705,-0.00056568431,-4.12131197e-007,
```

```

    7.85991348e-010,-3.03616152e-013},
    blow={-5595.53991,-4.9011295},
    ahigh={3771388.47,-12674.9343,20.8055593,-0.0102635936,3.30820359e-006,-
4.90319225e-010,
        2.64827508e-014},
    bhigh={70626.0972,-113.02536},
    R=84.06744154827575);

constant IdealGases.Common.DataRecord BrF3(
  name="BrF3",
  MM=0.1368992096,
  Hf=-1867067.025053153,
  H0=107463.1478369032,
  Tlimit=1000,
  allow={111645.5127,-1959.42433,15.26220126,-0.0079993623,6.98203026e-006,-
3.24275796e-009,
        6.16475146e-013},
  blow={-23453.41551,-55.226336},
  ahigh={-188653.224,-82.45232970000001,10.06180694,-2.477319092e-005,
        5.47857686e-009,-6.28265526e-013,2.910628214e-017},
  bhigh={-33868.0037,-22.98937369},
  R=60.73425861474075);

constant IdealGases.Common.DataRecord BrF5(
  name="BrF5",
  MM=0.174896016,
  Hf=-2451742.525684519,
  H0=109633.9610160131,
  Tlimit=1000,
  allow={221136.6805,-4024.80922,27.34900212,-0.01832254979,1.721754649e-005,-
8.75350863e-009,
        1.860196959e-012},
  blow={-35374.4767,-124.4296466},
  ahigh={-375764.764,-157.5896472,16.11869838,-4.776497799999999e-005,
        1.059755365e-008,-1.218527723e-012,5.65753545e-017},
  bhigh={-56672.66800000001,-55.4083931},
  R=47.53951628034798);

constant IdealGases.Common.DataRecord BrO(
  name="BrO",
  MM=0.0959034,
  Hf=1311736.601622049,
  H0=94480.48765737189,
  Tlimit=1000,
  allow={14554.31116,122.5152439,-0.3058728672,0.02117309202,-3.49426119e-005,
        2.622639719e-008,-7.50805954e-012},
  blow={13891.49932,25.27867438},
  ahigh={-2392612.795,6932.79313,-2.513183892,0.00344654456,-
7.569536839999999e-007,
        6.51429134e-011,-1.715768736e-015},
  bhigh={-30844.99064,53.6104836},
  R=86.6963215068496);

constant IdealGases.Common.DataRecord OBrO(
  name="OBrO",
  MM=0.1119028,
  Hf=1357915.31579192,

```

---

**974 Modelica.Media.IdealGases.Common.SingleGasesData**

---

```
H0=101826.1383986817,
Tlimit=1000,
alow={34702.13230000001,-340.134182,3.93225162,0.01215938586,-1.901501702e-
005,
    1.398631495e-008,-3.99289976e-012},
blow={18759.55788,6.36123518},
ahigh={-162450.466,-145.3689372,7.10582213,-4.14136207e-005,
    8.981833259999999e-009,-1.013712869e-012,4.63513314e-017},
bhigh={16500.75386,-9.11181792},
R=74.300839657274);

constant IdealGases.Common.DataRecord BrOO (
  name="BrOO",
  MM=0.1119028,
  Hf=965123.3034383413,
  H0=114837.6984311385,
  Tlimit=1000,
  alow={-44176.199,495.963716,3.23988495,0.00594150211,-3.40269677e-006,-
3.44477816e-011,
    4.81195795e-013},
  blow={8815.22198,16.07446878},
  ahigh={6172.483480000001,-658.504966,7.48395533,-0.0001920530746,
    4.22694617e-008,-4.83739261e-012,2.23974132e-016},
  bhigh={14603.5347,-9.85038024},
  R=74.300839657274);

constant IdealGases.Common.DataRecord BrO3 (
  name="BrO3",
  MM=0.1279022,
  Hf=1726480.740753482,
  H0=102428.5821510498,
  Tlimit=1000,
  alow={106390.1881,-1501.928813,9.70365743,0.00813300076,-1.531857845e-005,
    1.199117619e-008,-3.51274787e-012},
  blow={32331.3599,-27.3512543},
  ahigh={-282285.1113,-235.8973461,10.17525279,-6.97845105e-005,
    1.535701661e-008,-1.75445495e-012,8.10410015e-017},
  bhigh={24011.4014,-25.88197197},
  R=65.00648151478239);

constant IdealGases.Common.DataRecord Br2 (
  name="Br2",
  MM=0.159808,
  Hf=193419.6035242291,
  H0=60855.00725871045,
  Tlimit=1000,
  alow={7497.04754,-235.0884557,5.49193432,-0.002227573303,2.932401703e-006,-
1.954889514e-009,
    5.31230789e-013},
  blow={3521.47505,-1.96415157},
  ahigh={-4311698.57,11112.68634,-5.55577561,0.00363051659,-2.754164226e-007,
    -6.21750676e-011,7.37534162e-015},
  bhigh={-70365.8416,78.7847802},
  R=52.02788346015218);

constant IdealGases.Common.DataRecord BrBrO (
  name="BrBrO",
```

```

MM=0.1758074,
Hf=955591.1753430174,
H0=74726.64973146751,
Tlimit=1000,
alow={16174.98037,-246.4406776,5.91285845,0.00498035428,-8.09995825e-006,
       6.072776260000001e-009,-1.753715313e-012},
blow={19740.12093,2.009704963},
ahigh={-85371.0175,-64.8704607,7.04715935,-1.843088186e-005,3.99226786e-009,
       -4.50068795e-013,2.055867395e-017},
bhigh={18215.66655,-3.162186125},
R=47.29307185021791);

constant IdealGases.Common.DataRecord BrOBr(
  name="BrOBr",
  MM=0.1758074,
  Hf=612255.2634303221,
  H0=70526.37147241812,
  Tlimit=1000,
  alow={64050.6891,-1062.722566,9.90786628999999,-0.00447049247,
         3.91370107e-006,-1.805444867e-009,3.36613255e-013},
  blow={16429.21537,-23.50319399},
  ahigh={-96866.6024999999,-39.5739527,7.02943438,-1.171881908e-005,
         2.576958549e-009,-2.941186078e-013,1.357198784e-017},
  bhigh={10769.42321,-5.69777938},
  R=47.29307185021791);

constant IdealGases.Common.DataRecord C(
  name="C",
  MM=0.0120107,
  Hf=59670127.46967287,
  H0=544172.696012722,
  Tlimit=1000,
  alow={649.503147,-0.964901086,2.504675479,-1.281448025e-005,
         1.980133654e-008,-1.606144025e-011,5.314483411e-015},
  blow={85457.6311,4.747924288},
  ahigh={-128913.6472,171.9528572,2.646044387,-0.000335306895,1.74209274e-007,
         -2.902817829e-011,1.642182385e-015},
  bhigh={84105.9785,4.130047418},
  R=692.2554055966764);

constant IdealGases.Common.DataRecord Cplus(
  name="Cplus",
  MM=0.0120101514,
  Hf=150659589.6867712,
  H0=553638.482858759,
  Tlimit=1000,
  alow={2258.535929,-1.574575687,2.50363773,-5.20287837e-006,4.51690839e-009,
         -2.181431053e-012,4.495047033e-016},
  blow={216895.1913,4.345699505},
  ahigh={12551.12551,-34.1187467,2.543383218,-2.805120849e-005,
         9.75164196999999e-009,-1.736855394e-012,1.246191931e-016},
  bhigh={217100.1786,4.063913515},
  R=692.2870264566357);

constant IdealGases.Common.DataRecord Cminus(
  name="Cminus",
  MM=0.0120112486,

```

---

**976 Modelica.Media.IdealGases.Common.SingleGasesData**

---

```
Hf=48980273.04172191,
H0=517751.0021730797,
Tlimit=1000,
allow={4.67129153,-0.001986169369,2.500008638,-1.976750928e-008,
       2.478947477e-011,-1.610664044e-014,4.23650681e-018},
blow={70012.1854999999,4.879570141},
ahigh={4.25317572,0.000577818647999999,2.499999424,2.836136231e-010,-
7.32725342e-014,
       9.478507810000001e-018,-4.83048731999999e-022},
bhigh={70012.1717000001,4.879624211},
R=692.22378762521);

constant IdealGases.Common.DataRecord CBr(
  name="CBr",
  MM=0.0919147,
  Hf=5335727.756278375,
  H0=104609.1212831027,
  Tlimit=1000,
  allow={4324.44688,-159.6366559,4.85785659,-0.000438943916,4.40886363e-007,-
2.281330618e-010,
       5.44561822e-014},
  blow={58476.7725,0.1351169646},
  ahigh={1021862.102,-3243.48436,8.38822656,-0.002242155508,
       6.98835536999999e-007,-9.719134370000001e-011,4.71697624e-015},
  bhigh={78059.2405999999,-25.18386915},
  R=90.45856647522105);

constant IdealGases.Common.DataRecord CBr2(
  name="CBr2",
  MM=0.1718187,
  Hf=1959177.429464895,
  H0=70986.5631622169,
  Tlimit=1000,
  allow={72481.9783999999,-1128.26123,9.82230337999999,-0.0038727481,
       2.904819596e-006,-1.061380159e-009,1.297474642e-013},
  blow={44377.7159999999,-23.61121795},
  ahigh={-744143.141,421.542783,8.796797740000001,-0.002726562507,
       1.336438546e-006,-2.304854483e-010,1.337493953e-014},
  bhigh={33646.1814,-15.99565135},
  R=48.39096093731357);

constant IdealGases.Common.DataRecord CBr3(
  name="CBr3",
  MM=0.2517227,
  Hf=933566.9766771132,
  H0=61903.25703641347,
  Tlimit=1000,
  allow={85635.1626,-1275.764123,11.17190031,0.002341705749,-6.54067141e-006,
       5.73671714e-009,-1.777823046e-012},
  blow={32432.1589,-27.66468801},
  ahigh={-192649.8949,-132.4072178,10.09808665,-3.89504082e-005,
       8.55044501e-009,-9.74749367999999e-013,4.49424559e-017},
  bhigh={25415.76129,-18.66868947},
  R=33.03028292641069);

constant IdealGases.Common.DataRecord CBr4(
  name="CBr4",
```

```

MM=0.3316267000000001,
Hf=239727.3802139574,
H0=61434.48039618039,
Tlimit=1000,
alow={102308.3739,-1735.600709,16.51317914,-0.003147664747,2.924135486e-007,
      1.360605681e-009,-6.43175865e-013},
blow={15005.02543,-55.3647676},
ahigh={-205978.5072,-104.8517002,13.0777255,-3.086857556e-005,
       6.77530981e-009,-7.72183754e-013,3.55921824e-017},
bhigh={5615.89394,-32.8378763},
R=25.0717810116013);

```

```

constant IdealGases.Common.DataRecord CCL(
  name="CCL",
  MM=0.0474637,
  Hf=9114566.668843769,
  H0=197943.1228496725,
  Tlimit=1000,
  alow={17070.70049,-14.63803497,2.334108251,0.00720209913,-1.034120045e-005,
        7.2281067e-009,-1.985123755e-012},
  blow={51233.503,12.00820386},
  ahigh={590050.763,-2075.31146,6.88001723,-0.001294796228,3.88313798e-007,-
5.00698163e-011,
        2.218229619e-015},
  bhigh={63589.2205,-16.10664472},
  R=175.1753866639137);

```

```

constant IdealGases.Common.DataRecord CCL2(
  name="CCL2",
  MM=0.08291670000000001,
  Hf=2688724.310060578,
  H0=137761.8501459899,
  Tlimit=1000,
  alow={75096.23150000001,-1034.728559,8.061704539999999,0.001487134213,-
4.56481711e-006,
        3.95488581e-009,-1.166197091e-012},
  blow={30524.09293,-17.39691692},
  ahigh={-6437455.17,19584.13353,-16.04038625,0.01238833906,-3.013367274e-006,
        3.47346892e-010,-1.553272793e-014},
  bhigh={-99612.42159999999,155.6949315},
  R=100.2749988844225);

```

```

constant IdealGases.Common.DataRecord CCL2Br2(
  name="CCL2Br2",
  MM=0.2427247,
  Hf=41198.93855054718,
  H0=77008.69338802355,
  Tlimit=1000,
  alow={101849.5619,-1740.854039,15.21338304,0.00112825651,-
5.875405490000001e-006,
        5.65988697e-009,-1.821576512e-012},
  blow={6918.57219,-50.15385818999999},
  ahigh={-255334.7012,-159.8005477,13.1186093,-4.717443870000001e-005,
        1.036912808e-008,-1.183321687e-012,5.46061269e-017},
  bhigh={-2588.63698,-34.61705319},
  R=34.25474210082452);

```

```
constant IdealGases.Common.DataRecord CCL3(
  name="CCL3",
  MM=0.1183697,
  Hf=600897.0200988936,
  H0=121652.7540409412,
  Tlimit=1000,
  allow={34144.7165,-554.796973,6.84554439,0.01242006495,-2.036896881e-005,
         1.554979993e-008,-4.56894011e-012},
  blow={9388.56882,-7.143598465},
  ahighelement={-542757.254,680.902062,9.047723619999999,0.0001324277766,
               6.30126304e-008,-2.6566392e-011,2.105668053e-015},
  bhighelement={386.621615,-15.28745849},
  R=70.24155674974254);

constant IdealGases.Common.DataRecord CCL3Br(
  name="CCL3Br",
  MM=0.1982737,
  Hf=-216871.9300643504,
  H0=90705.67099922984,
  Tlimit=1000,
  allow={104966.8463,-1793.828482,14.89250892,0.00239712831,-7.78961547e-006,
         7.01661711e-009,-2.196069746e-012},
  blow={910.1349309999999,-50.58348139},
  ahighelement={-277232.4186,-184.0310333,13.13673812,-5.44391001e-005,
               1.197658923e-008,-1.367818632e-012,6.316164790000001e-017},
  bhighelement={-8895.857689999999,-36.69528209},
  R=41.93431604897675);

constant IdealGases.Common.DataRecord CCL4(
  name="CCL4",
  MM=0.1538227,
  Hf=-621494.746874161,
  H0=111547.3333909754,
  Tlimit=1000,
  allow={109338.5626,-1861.731846,14.49632467,0.00394948516,-1.010805379e-005,
         8.64551417e-009,-2.642368852e-012},
  blow={-4948.00623,-51.8028475},
  ahighelement={-304307.8255,-216.8170491,13.16132386,-6.43112489e-005,
               1.416482497e-008,-1.61934332e-012,7.48397435e-017},
  bhighelement={-15123.2007,-39.968443},
  R=54.05230827439643);

constant IdealGases.Common.DataRecord CF(
  name="CF",
  MM=0.0310091032,
  Hf=7813834.487157951,
  H0=292337.0257286254,
  Tlimit=1000,
  allow={-45827.8668,797.263589,-1.364936319,0.01312359375,-1.457012892e-005,
         8.25327668e-009,-1.896351845e-012},
  blow={24382.5962,32.4806528},
  ahighelement={-132980.71,-121.4340891,4.45241362,0.0001293786948,-3.67055059e-008,
               6.59021238999999e-012,-3.73126161e-016},
  bhighelement={28159.34108,-0.664046873},
  R=268.1300373756052);

constant IdealGases.Common.DataRecord CFplus(
```

```

name="CFplus",
MM=0.0310085546,
Hf=36943485.87921605,
H0=280487.0821034657,
Tlimit=1000,
alow={-59752.132, 927.686364999999, -1.866581982, 0.01384340912, -1.51342886e-
005,
      8.51855634e-009, -1.959577655e-012},
blow={132351.941, 34.1201319},
ahigh={-80537.6278, -339.027509, 4.7083735, -3.96314831e-005, 2.497996218e-008,
      -5.23679043e-012, 5.31394913e-016},
bhigh={138133.7068, -3.92051473},
R=268.1347811032766);

constant IdealGases.Common.DataRecord CFB3 (
  name="CFBr3",
  MM=0.2707211032,
  Hf=-443260.6050343547,
  H0=67387.07763953881,
  Tlimit=1000,
  alow={51492.4428, -936.479664, 10.44260886, 0.01205370047, -1.876290291e-005,
        1.344488618e-008, -3.73054576e-012},
  blow={-12432.46293, -23.63948551},
  ahigh={-272915.8956, -332.816465, 13.24822519, -9.92707754e-005,
        2.193640054e-008, -2.515424945e-012, 1.165680878e-016},
  bhigh={-17301.85479, -36.2681843},
  R=30.71231574384335);

constant IdealGases.Common.DataRecord CFCL (
  name="CFCL",
  MM=0.0664621032,
  Hf=388879.8692124447,
  H0=164039.0760309253,
  Tlimit=1000,
  alow={15419.97945, -82.7403980999999, 2.501701822, 0.01390738086, -1.8957347e-
005,
        1.261934581e-008, -3.33345382e-012},
  blow={2411.755114, 13.31517094},
  ahigh={-268163.0077, -2.328091539, 6.83463383, 0.0001565270633, -
5.967755160000001e-008,
        9.82092446e-012, -5.05889721e-016},
  bhigh={301.7669604, -9.005085128999999},
  R=125.1009462487188);

constant IdealGases.Common.DataRecord CFCLBr2 (
  name="CFCLBr2",
  MM=0.2262701032,
  Hf=-773411.9422985264,
  H0=77332.45688465308,
  Tlimit=1000,
  alow={58987.9974, -1050.714861, 10.39349401, 0.01273690622, -1.999097478e-005,
        1.43820423e-008, -4.00013035e-012},
  blow={-18377.99602, -24.17479131},
  ahigh={-296081.1596, -353.12762, 13.26346169, -0.0001053860298, 2.32908508e-008,
        -2.670979639e-012, 1.237844667e-016},
  bhigh={-23875.94306, -36.84977271},
  R=36.74578250689551);

```

```
constant IdealGases.Common.DataRecord CFCL2 (
  name="CFCL2",
  MM=0.1019151032,
  Hf=-1030269.280049161,
  H0=129685.9207811703,
  Tlimit=1000,
  allow={26113.98792,-335.35659,4.09467308,0.01960244711,-2.762775193e-005,
         1.880124302e-008,-5.0474103e-012},
  blow={-12512.98871,6.870882561},
  ahighelement={-265311.8323,-446.972417,10.33334881,-0.0001333673368,
                2.948755634e-008,-3.3833062e-012,1.568746348e-016},
  bhigh={-13931.60489,-25.56817972},
  R=81.58233410884678);

constant IdealGases.Common.DataRecord CFCL2Br (
  name="CFCL2Br",
  MM=0.1818191032,
  Hf=-1292493.450160192,
  H0=92017.68519117852,
  Tlimit=1000,
  allow={65299.59220000001,-1135.235485,10.09526872,0.01401380147,-
         2.191705096e-005,
         1.573404132e-008,-4.36934034e-012},
  blow={-25044.88145,-24.51093103},
  ahighelement={-321545.505,-388.412962,13.28990796,-0.0001160048581,
                2.564498787e-008,-2.941639768e-012,1.363542881e-016},
  bhigh={-30974.21144,-38.71356695},
  R=45.72936426187367);

constant IdealGases.Common.DataRecord CFCL3 (
  name="CFCL3",
  MM=0.1373681032,
  Hf=-2065253.820873898,
  H0=116938.2383959423,
  Tlimit=1000,
  allow={74001.74000000001,-1252.282466,10.01327229,0.01475107601,-
         2.315635251e-005,
         1.664207025e-008,-4.62292996e-012},
  blow={-30205.08601,-27.08387087},
  ahighelement={-345046.478,-417.782145,13.31196736,-0.0001248759531,
                2.761391353e-008,-3.16820764e-012,1.468834601e-016},
  bhigh={-36740.6719,-41.56886057000001},
  R=60.52694771430753);

constant IdealGases.Common.DataRecord CF2 (
  name="CF2",
  MM=0.0500075064,
  Hf=-3731439.806404744,
  H0=206987.1654308282,
  Tlimit=1000,
  allow={-37970.2627,873.1800030000001,-3.46019157,0.02746253741,-3.4746219e-
         005,
         2.204470693e-008,-5.62175085e-012},
  blow={-27467.97157,44.5677807},
  ahighelement={-108642.8547,-585.498914,7.01864895,0.000392918615,-2.603822675e-007,
                6.142196389999999e-011,-4.17283326e-015},
  bhigh={-21529.45157,-13.56143285},
  R=166.2644790462898);
```

```

constant IdealGases.Common.DataRecord CF2plus(
  name="CF2plus",
  MM=0.0500069578,
  Hf=18984184.21686112,
  H0=206811.2009805164,
  Tlimit=1000,
  allow={-28923.73827, 604.935461, -0.972186666, 0.01808478341, -1.980065782e-005,
         1.101640055e-008, -2.500457119e-012},
  blow={110275.567, 32.4759147},
  ahighe={-64889.6607, -1064.283593, 7.7857021, -0.0003158347626,
          7.130804079999999e-008, -8.53318359e-012, 4.25163882e-016},
  bhigh={117844.6465, -18.19424707},
  R=166.2663030463333);

constant IdealGases.Common.DataRecord CF2Br2(
  name="CF2Br2",
  MM=0.2098155064,
  Hf=-1811114.948175251,
  H0=77591.96295512693,
  Tlimit=1000,
  allow={12614.04028, -281.5725114, 5.09803213, 0.02502548939, -3.43885319e-005,
         2.297615553e-008, -6.08016143e-012},
  blow={-46427.6375, 3.094167118},
  ahighe={-332766.93, -618.435249, 13.46170049, -0.0001849172548, 4.09259134e-008,
          -4.69981043e-012, 2.180811799e-016},
  bhigh={-47151.8661, -41.0536924},
  R=39.62753822469625);

constant IdealGases.Common.DataRecord CF2CL(
  name="CF2CL",
  MM=0.0854605064,
  Hf=-3217860.642117609,
  H0=145473.8981045869,
  Tlimit=1000,
  allow={8914.558349999999, -17.68816959, 1.607015025, 0.02473626554, -3.27073894e-
005,
         2.12635183e-008, -5.50983110000001e-012},
  blow={-34273.0625, 19.2968069},
  ahighe={-280234.6654, -686.031485, 10.51143635, -0.0002046954963,
          4.52889718e-008, -5.20022903e-012, 2.41297145e-016},
  bhigh={-33081.7948, -28.74899785},
  R=97.29022621377774);

constant IdealGases.Common.DataRecord CF2CLBr(
  name="CF2CLBr",
  MM=0.1653645064,
  Hf=-2630552.404926478,
  H0=93903.11946651207,
  Tlimit=1000,
  allow={26966.99365, -467.883227, 5.24480942, 0.02530121349, -3.508443520000001e-
005,
         2.354353958e-008, -6.24616113e-012},
  blow={-51983.8273, 0.8532887803},
  ahighe={-359282.884, -653.928359, 13.48831524, -0.0001956099469, 4.32977597e-008,
          -4.97264437e-012, 2.30757505e-016},
  bhigh={-53651.4117, -42.24913908000001},
  R=50.27966509262958);

```

```
constant IdealGases.Common.DataRecord CF2CL2(
  name="CF2CL2",
  MM=0.1209135064,
  Hf=-4059099.885635274,
  H0=123072.2228066988,
  Tlimit=1000,
  allow={38412.5875,-613.949982,5.26966719,0.02574105783,-3.58737648e-005,
         2.41160013e-008,-6.4022369e-012},
  blow={-57845.4133,-1.957454856},
  ahighelement={-382744.855,-693.37748,13.51791274,-0.0002075114435,4.59402161e-008,
             -5.276874600000001e-012,2.449037044e-016},
  bhighelement={-60215.2777,-44.79580046},
  R=68.76379858255437);

constant IdealGases.Common.DataRecord CF3(
  name="CF3",
  MM=0.0690059096,
  Hf=-6773332.932053692,
  H0=166528.7808915426,
  Tlimit=1000,
  allow={-29783.07106,715.367883,-3.49818538,0.0359545799,-4.50797443e-005,
         2.82180845e-008,-7.098047020000001e-012},
  blow={-60599.9703,45.0259264},
  ahighelement={-299730.5557,-1046.989457,10.77923191,-0.0003116087076,
               6.89135143e-008,-7.91122564e-012,3.67059302e-016},
  bhighelement={-54253.044,-34.1703879},
  R=120.489274733073);

constant IdealGases.Common.DataRecord CF3plus(
  name="CF3plus",
  MM=0.06900536099999999,
  Hf=6138906.801748347,
  H0=167251.9327882366,
  Tlimit=1000,
  allow={-35874.6354,374.35067,0.778741393,0.01929435768,-1.89281577e-005,
         9.36390876e-009,-1.885759656e-012},
  blow={47755.8172,22.24044005},
  ahighelement={-29109.19531,-1996.658183,11.44279678,-0.000565339304,
                1.232297113e-007,-1.399758046e-011,6.44301795e-016},
  bhighelement={59023.458,-40.8034999000001},
  R=120.4902326356934);

constant IdealGases.Common.DataRecord CF3Br(
  name="CF3Br",
  MM=0.1489099096,
  Hf=-4356996.80258217,
  H0=96998.43374292129,
  Tlimit=1000,
  allow={-5439.48901,124.361519,0.923194195,0.0348104438,-4.54964549e-005,
         2.932836552e-008,-7.54800036e-012},
  blow={-80233.9676,22.32999096},
  ahighelement={-383408.851,-971.9093300000001,13.72477396,-0.0002901710612,
               6.42189220000001e-008,-7.37568877e-012,3.42315425e-016},
  bhighelement={-77653.0631,-47.1736721},
  R=55.83558557206995);

constant IdealGases.Common.DataRecord CF3CL(
```

```

name="CF3CL",
MM=0.1044589096,
Hf=-6741406.766512906,
H0=132020.8209410603,
Tlimit=1000,
alow={14994.09978,-136.5768572,1.45291237,0.0340476507,-4.47377764e-005,
      2.888115112e-008,-7.43417133e-012},
blow={-85471.6664,17.27327095},
ahigh={-406604.2119999999,-1022.626507,13.76252628,-0.0003052722979,
       6.75596604e-008,-7.75931525e-012,3.60119229e-016},
bhigh={-84105.5937,-49.13411939},
R=79.59562311954289);

constant IdealGases.Common.DataRecord CF4 (
  name="CF4",
  MM=0.0880043128,
  Hf=-10603116.71452538,
  H0=144655.558289866,
  Tlimit=1000,
  alow={9817.458500000001,116.3343483,-1.288338636,0.0395956691,-4.99624421e-
005,
        3.125339346e-008,-7.841700320000001e-012},
  blow={-113850.2297,29.38656548},
  ahigh={-416445.678,-1414.797167,14.05124837,-0.000419909386,9.27892161e-008,
         -1.06457583e-011,4.937099680000001e-016},
  bhigh={-109469.1149,-54.87105},
  R=94.47800608244737);

constant IdealGases.Common.DataRecord CHplus (
  name="CHplus",
  MM=0.0130180914,
  Hf=125254229.2029076,
  H0=662777.9552999605,
  Tlimit=1000,
  alow={30196.07105,-461.814131,6.22564119,-0.0077757118,1.094489485e-005,-
6.67548791e-009,
        1.565232409e-012},
  blow={197249.1928,-14.31512811},
  ahigh={-7102094.67,18283.54883,-13.12691402,0.00619171736000001,-
2.909421253e-007,
        -1.134243575e-010,1.105962085e-014},
  bhigh={75412.9604,124.3984829},
  R=638.6859443927395);

constant IdealGases.Common.DataRecord CHBr3 (
  name="CHBr3",
  MM=0.25273064,
  Hf=66236.5275536041,
  H0=62938.94954723337,
  Tlimit=1000,
  alow={43465.7686,-499.963442,5.62022211,0.02079346321,-2.921336328e-005,
        2.066942448e-008,-5.78122091e-012},
  blow={2627.830062,1.247904015},
  ahigh={627423.495,-3378.7111,14.87852612,-0.000593274527,1.082421577e-007,-
1.061012665e-011,
        4.31915245e-016},
  bhigh={18772.34659,-53.1190965},
  R=32.89855159627658);

```

```
constant IdealGases.Common.DataRecord CHCL (
  name="CHCL",
  MM=0.04847164,
  Hf=6129357.290159771,
  H0=210441.4663914817,
  Tlimit=1000,
  allow={-269991.2334, 4351.19973, -22.75911813, 0.0755062176000001, -9.07199797e-005,
         5.25697186e-008, -1.189769182e-011},
  blow={14168.6268, 152.0980812},
  ahigh={-954806.190000001, 2174.413794, 4.86764537, 0.0008321641859999999, -1.536948638e-007,
         1.529236537e-011, -6.596159359999999e-016},
  bhigh={18801.2181, 2.674761385},
  R=171.5327147998293);

constant IdealGases.Common.DataRecord CHCLBr2 (
  name="CHCLBr2",
  MM=0.20827964,
  Hf=48012.37413316059,
  H0=73416.85437904541,
  Tlimit=1000,
  allow={38153.8025, -408.417511000001, 4.63924402, 0.02325990646, -3.22298886e-005,
         2.254671151e-008, -6.25650774e-012},
  blow={1483.809044, 6.175235281},
  ahigh={604286.53, -3438.24101, 14.94162631, -0.000622842797, 1.153724426e-007, -1.147291377e-011,
         4.733482859999999e-016},
  bhigh={18224.60958, -54.07258924},
  R=39.91975403836881);

constant IdealGases.Common.DataRecord CHCL2 (
  name="CHCL2",
  MM=0.08392464,
  Hf=1141500.27929819,
  H0=152517.7826202174,
  Tlimit=1000,
  allow={56385.5613, -656.245542, 6.06881802, 0.0097441226, -1.275310131e-005,
         8.55238672e-009, -2.2315827e-012},
  blow={13304.47798, -4.533535456},
  ahigh={884939.3709999999, -3526.96973, 11.86372153, -0.000500237612,
         6.1338604e-008, -1.885433405e-012, -1.234221421e-016},
  bhigh={30711.36475, -40.88905858},
  R=99.0706900857722);

constant IdealGases.Common.DataRecord CHCL2Br (
  name="CHCL2Br",
  MM=0.16382864,
  Hf=-274677.248129509,
  H0=89910.37830748031,
  Tlimit=1000,
  allow={32940.2265, -327.530117, 3.77323524, 0.02536466143, -3.47266699e-005,
         2.404666618e-008, -6.62173093e-012},
  blow={-5425.56508, 9.433412003999999},
  ahigh={592487.644, -3509.47561, 14.99349593, -0.000643306257, 1.198581809e-007,
         -1.198476716e-011, 4.96994807e-016},
  bhigh={11972.51405, -56.02687781},
```

```

R=50.75102863577455);

constant IdealGases.Common.DataRecord CHCL3 (
  name="CHCL3",
  MM=0.11937764,
  Hf=-860295.1105416392,
  H0=119833.9488031427,
  Tlimit=1000,
  allow={33953.3329,-304.7428785,2.923672263,0.02830547858,-3.71242469e-005,
    2.551365915e-008,-6.98765955e-012},
  blow={-12350.63157,11.15556408},
  ahigh={613605.274,-3715.08717,15.10777247,0.0002362584336,1.297140438e-007,
    -1.267494791e-011,5.25902230999999e-016},
  bhigh={6203.31345,-59.92576539},
  R=69.64848693607949);

constant IdealGases.Common.DataRecord CHF (
  name="CHF",
  MM=0.03201704320000001,
  Hf=3398190.123939989,
  H0=311746.9635672041,
  Tlimit=1000,
  allow={-78685.85829999999,1300.218466,-3.94768525,0.02114995848,-
    2.239962738e-005,
    1.283155181e-008,-2.904778136e-012},
  blow={5824.38972,47.8545377},
  ahigh={3994085.42,-7962.756200000001,6.75559509,0.00494983657,-2.101763812e-
    006,
    3.34823295e-010,-1.88682505e-014},
  bhigh={67061.34940000001,-23.81240379},
  R=259.6889396707313);

constant IdealGases.Common.DataRecord CHFBr2 (
  name="CHFBr2",
  MM=0.1918250432,
  Hf=-912289.6420648375,
  H0=74857.56687685709,
  Tlimit=1000,
  allow={-18201.9798,436.224967,-0.373385417,0.033886314,-4.37129853e-005,
    2.889569295e-008,-7.683915110000001e-012},
  blow={-24656.11586,33.2047879},
  ahigh={593575.1630000001,-3742.65658,15.17184031,-0.0007158039030000001,
    1.360636972e-007,-1.385897355e-011,5.844209110000001e-016},
  bhigh={-2347.676027,-57.6578343},
  R=43.34403819907494);

constant IdealGases.Common.DataRecord CHFCL (
  name="CHFCL",
  MM=0.06747004320000001,
  Hf=-1232320.301819519,
  H0=165198.5751211139,
  Tlimit=1000,
  allow={-71130.0851,1377.872745,-6.48461724,0.0389097249,-4.80073522e-005,
    3.073300019e-008,-7.98285341e-012},
  blow={-17517.93745,63.73895867},
  ahigh={662216.358,-3809.00798,12.23880469,-0.000746718275,1.43476785e-007,-
    1.475406631e-011,
    1.475406631e-011};

```

```
    6.27378068e-016},
bhigh={10189.87746,-46.58365563},
R=123.2320538961801);

constant IdealGases.Common.DataRecord CHFCLBr (
  name="CHFCLBr",
  MM=0.1473740432,
  Hf=-1560654.746289813,
  H0=93552.3359516542,
  Tlimit=1000,
  allow={-19592.40614,490.172327,-1.259408503,0.0361770847,-4.65424496e-005,
         3.065240659e-008,-8.1239036e-012},
  blow={-31399.00705,37.10450945},
  ahigh={580011.7509999999,-3823.33785,15.22809062,-0.000737390631,
         1.407097155e-007,-1.438245319e-011,6.08386715e-016},
  bhigh={-8552.249899999999,-59.03677705},
  R=56.41747908562503);

constant IdealGases.Common.DataRecord CHFCL2 (
  name="CHFCL2",
  MM=0.1029230432,
  Hf=-2768087.603534833,
  H0=129167.984026341,
  Tlimit=1000,
  allow={-17349.59015,492.49132,-1.854635884,0.0378278628,-4.86334304e-005,
         3.19719763e-008,-8.45905799e-012},
  blow={-37887.4927,38.01530544},
  ahigh={564011.302,-3887.0833,15.27760246,-0.0007576487420000001,
         1.452498792e-007,-1.490796515e-011,6.3289985e-016},
  bhigh={-14847.36185,-61.67535516},
  R=80.78338670809922);

constant IdealGases.Common.DataRecord CHF2 (
  name="CHF2",
  MM=0.0510154464,
  Hf=-4682895.414201452,
  H0=214052.8167562991,
  Tlimit=1000,
  allow={-146969.0117,2553.397313,-13.0262763,0.0543812846,-6.71341879e-005,
         4.28765173e-008,-1.110824216e-011},
  blow={-41793.7216,99.39930250000001},
  ahigh={552680.655,-3696.00917,12.08610573,-0.00064985881,1.12084622e-007,-
9.81414700999999e-012,
         3.34919239e-016},
  bhigh={-9437.06842,-47.044162},
  R=162.9795010477454);

constant IdealGases.Common.DataRecord CHF2Br (
  name="CHF2Br",
  MM=0.1309194464,
  Hf=-3223356.129315255,
  H0=100592.4204702412,
  Tlimit=1000,
  allow={-77145.1016,1422.544046,-6.68439365,0.0473838915,-5.83058875e-005,
         3.69572047e-008,-9.49548528e-012},
  blow={-58785.00930000001,66.0863909},
  ahigh={576770.684,-4177.71608,15.50150581,-0.000849014729,1.657177491e-007,
```

```

-1.727828054e-011, 7.43562723e-016},
bhigh={-29661.47348, -63.4938014},
R=63.50830398867315);

constant IdealGases.Common.DataRecord CHF2CL(
  name="CHF2CL",
  MM=0.08646844640000001,
  Hf=-5583539.662162821,
  H0=143029.0067059653,
  Tlimit=1000,
  allow={-72405.7941, 1365.510973, -7.15322208, 0.0485498797, -5.94878167e-005,
         3.75096619e-008, -9.58952442e-012},
  blow={-65659.32799999999, 66.5721223},
  ahigh={562949.373, -4298.67304999999, 15.58191398, -0.000878927848,
         1.720248689e-007, -1.797910869e-011, 7.75339169999999e-016},
  bhigh={-36340.6716, -66.1126147999999},
  R=96.15613956491762);

constant IdealGases.Common.DataRecord CHF3 (
  name="CHF3",
  MM=0.0700138496,
  Hf=-9902326.524836596,
  H0=165208.5418254162,
  Tlimit=1000,
  allow={-109513.226, 2042.273879, -11.66213079, 0.0578580777, -6.857135060000001e-
005,
         4.21211838e-008, -1.053196888e-011},
  blow={-93954.70050000001, 89.3592065},
  ahigh={568523.2020000001, -4728.3617, 15.86728516, -0.000738234865,
         1.970841706e-007, -2.062115571e-011, 8.97059963999999e-016},
  bhigh={-59231.9637, -71.6127322},
  R=118.7546756463453);

constant IdealGases.Common.DataRecord CHI3 (
  name="CHI3",
  MM=0.39373205,
  Hf=535576.4154835757,
  H0=43574.2708778724,
  Tlimit=1000,
  allow={52529.2359, -724.216647, 8.01862334, 0.01470995487, -2.163843469e-005,
         1.594571452e-008, -4.60143796e-012},
  blow={26781.86915, -8.60505483999999},
  ahigh={629420.543, -3184.06157, 14.79622178, -0.00057495555, 1.062025541e-007, -
1.052804696e-011,
         4.32972002e-016},
  bhigh={41036.098, -49.0135739},
  R=21.11708203586678);

constant IdealGases.Common.DataRecord CH2 (
  name="CH2",
  MM=0.01402658,
  Hf=27830341.89374745,
  H0=0,
  Tlimit=1000,
  allow={32189.2173, -287.7601815, 4.20358382, 0.00345540596, -6.74619334e-006,
         7.65457164e-009, -2.870328419e-012},
  blow={0, 47336.2471},

```

```
ahigh={10027.417,2550418.031,-7971.62539,12.28924487,-0.001699122922,
       2.991728605e-007,-2.767007492e-011},
bhigh={1.05134174e-015,0},
R=592.7654495964092);

constant IdealGases.Common.DataRecord CH2Br2 (
  name="CH2Br2",
  MM=0.17383458,
  Hf=-84965.83361032081,
  H0=72565.5792995847,
  Tlimit=1000,
  allow={4797.30801,361.385924,-1.819592338,0.0347704834,-4.49854618e-005,
         3.068623685e-008,-8.43484634e-012},
  blow={-4481.49706,38.2712359},
  ahigh={1528284.441,-6673.41494999999,16.70213316,-0.001166351237,
         2.122634493e-007,-2.07547224e-011,8.4285777e-016},
  bhigh={36007.0664,-74.48065490000001},
  R=47.82979312861688);

constant IdealGases.Common.DataRecord CH2CL (
  name="CH2CL",
  MM=0.04947958,
  Hf=2409074.612193556,
  H0=221908.1083550022,
  Tlimit=1000,
  allow={-31885.8563,633.316321,-1.164065495,0.0216058608,-2.545462163e-005,
         1.693887757e-008,-4.6600786e-012},
  blow={10201.4254,32.30835289},
  ahigh={1662438.334,-6441.12572,13.59753722,-0.00114053681,2.087760159e-007,
         -2.052347186e-011,8.37577227e-016},
  bhigh={52129.0612,-61.48586271},
  R=168.0384514177364);

constant IdealGases.Common.DataRecord CH2CLBr (
  name="CH2CLBr",
  MM=0.12938358,
  Hf=-347803.0210634147,
  H0=94221.74745821687,
  Tlimit=1000,
  allow={-13079.48755,645.586038,-3.57734991,0.0383368723,-4.86425205e-005,
         3.26162233e-008,-8.852877389999999e-012},
  blow={-9402.27253,47.48806758},
  ahigh={1525016.309,-6823.33416,16.82769102,-0.001219735119,2.244964495e-007,
         -2.219063248e-011,9.104705129999999e-016},
  bhigh={33202.3992,-76.37571062000001},
  R=64.26218844771493);

constant IdealGases.Common.DataRecord CH2CL2 (
  name="CH2CL2",
  MM=0.08493258000000001,
  Hf=-1118534.25387525,
  H0=139570.4333955238,
  Tlimit=1000,
  allow={-25098.41179,868.766738,-5.09466921,0.0415004999,-5.19977215e-005,
         3.44594426e-008,-9.270292519999999e-012},
  blow={-16389.7884,53.9689032},
  ahigh={1529279.337,-6976.95476,16.94154931,-0.001265053995,2.344766734e-007,
```

```

-2.333227421e-011,9.632834730000001e-016},
bhigh={28063.18171,-79.49453509999999},
R=97.89496563038588);

constant IdealGases.Common.DataRecord CH2F(
  name="CH2F",
  MM=0.0330249832,
  Hf=-962907.3785569707,
  H0=336987.3023886958,
  Tlimit=1000,
  allow={-85697.1253,1392.226749,-4.38205259,0.02645916948,-2.848145663e-005,
    1.732706028e-008,-4.44206144e-012},
  blow={-11694.44586,50.494992},
  ahigh={2535399.502,-9358.43902,17.00366206,-0.003062919929,
    7.612869080000001e-007,-9.664554980000001e-011,4.8447681e-015},
  bhigh={52283.9131,-87.0755157},
  R=251.7630955221803);

constant IdealGases.Common.DataRecord CH2FBr(
  name="CH2FBr",
  MM=0.1129289832,
  Hf=-1903851.375507647,
  H0=102919.4248514229,
  Tlimit=1000,
  allow={-92561.0953,1811.985865,-9.69834687,0.0503584867,-6.07027268e-005,
    3.8782508e-008,-1.012615992e-011},
  blow={-35375.0962,81.4039605},
  ahigh={1539975.462,-7231.55136,17.14566439,-0.001350184394,2.538022824e-007,
    -2.558977386e-011,1.069310946e-015},
  bhigh={15096.04224,-80.62053710000001},
  R=73.62566955265032);

constant IdealGases.Common.DataRecord CH2FCL(
  name="CH2FCL",
  MM=0.0684779832,
  Hf=-3880079.225230454,
  H0=164320.8002656247,
  Tlimit=1000,
  allow={-107134.0332,2081.328396,-11.55723697,0.0544148211,-6.51862274e-005,
    4.13315988e-008,-1.071935103e-011},
  blow={-42647.5621,90.35923546999999},
  ahigh={1536120.418,-7378.93973,17.25797178,-0.001395645457,2.639247747e-007,
    -2.675653563e-011,1.123582778e-015},
  bhigh={9813.71046,-83.13642553},
  R=121.4181786825755);

constant IdealGases.Common.DataRecord CH2F2(
  name="CH2F2",
  MM=0.0520233864,
  Hf=-8694166.821866099,
  H0=205548.4607976231,
  Tlimit=1000,
  allow={-181991.75,3174.42789,-17.14256906,0.06411353830000001,-7.31359212e-
005,
    4.426384159999999e-008,-1.103877216e-011},
  blow={-70270.5895,120.7331926},
  ahigh={1546609.496,-7876.88333,17.68770469,-0.001581298721,3.068917255e-007,
    1.103877216e-011});

```

---

**990 Modelica.Media.IdealGases.Common.SingleGasesData**

---

```
-3.18340546e-011,1.363827177e-015},  
bhigh={-9800.13596,-89.0742036},  
R=159.8218142908129);  
  
constant IdealGases.Common.DataRecord CH2I2(  
  name="CH2I2",  
  MM=0.26783552,  
  Hf=438964.9289235423,  
  H0=49472.25819786711,  
  Tlimit=1000,  
  allow={27381.33757,-89.7256804,1.387869408,0.02733482611,-3.61940877e-005,  
         2.53918761e-008,-7.145585160000001e-012},  
  blow={13387.67276,22.41870165},  
  ahigh={1512892.716,-6442.9124,16.59026137,-0.00113555152,2.07375025e-007,-  
2.033902231e-011,  
         8.282283700000001e-016},  
  bhigh={50582.1903,-71.2523514},  
  R=31.0432014394506);  
  
constant IdealGases.Common.DataRecord CH3(  
  name="CH3",  
  MM=0.01503452,  
  Hf=9754753.726756824,  
  H0=0,  
  Tlimit=1000,  
  allow={-28761.88806,509.326866,0.2002143949,0.01363605829,-1.433989346e-005,  
         1.013556725e-008,-3.027331936e-012},  
  blow={0,14082.71825},  
  ahigh={10366.34,2760802.663,-9336.53117,14.87729606,-0.001439429774,  
         2.444477951e-007,-2.224555778e-011},  
  bhigh={8.39506576e-016,0},  
  R=553.0254374599256);  
  
constant IdealGases.Common.DataRecord CH3Br(  
  name="CH3Br",  
  MM=0.09493852,  
  Hf=-397520.4163705101,  
  H0=111784.5001164965,  
  Tlimit=1000,  
  allow={-71155.85769999999,1524.705928,-8.230445209999999,0.0423997321,-  
4.94697989e-005,  
         3.19443334e-008,-8.531237080000001e-012},  
  blow={-12517.53591,70.4812776999999},  
  ahigh={2524874.348,-10118.76098,18.65902163,-0.00179767369,3.29852011e-007,  
         -3.25091203e-011,1.330179925e-015},  
  bhigh={55405.7639,-97.7866446},  
  R=87.57743432275962);  
  
constant IdealGases.Common.DataRecord CH3CL(  
  name="CH3CL",  
  MM=0.05048752,  
  Hf=-1621588.859979654,  
  H0=206311.2626645159,  
  Tlimit=1000,  
  allow={-98419.71100000001,1983.700841,-10.84512305,0.0477798005,-5.51551626e-  
005,  
         3.50617614e-008,-9.23610396e-012},
```

```

blow={-19946.89506,83.99658330999999},
ahigh={2522463.305,-10301.15447,18.82725852,-0.001872291294,3.47337286e-007,
       -3.45888722e-011,1.428941079e-015},
bhigh={51114.1741,-100.6571389},
R=164.6837079737725);

constant IdealGases.Common.DataRecord CH3F(
  name="CH3F",
  MM=0.0340329232,
  Hf=-6984413.257806781,
  H0=297806.5957025989,
  Tlimit=1000,
  alow={-202982.1878,3447.33113,-17.68994275,0.059452758,-6.46952825e-005,
         3.85941804e-008,-9.626541530000001e-012},
  blow={-45779.26220000001,122.8382176},
  ahigh={2561903.188,-10860.52758,19.2944692,-0.002070973131,3.92912453e-007,
         -3.99450342e-011,1.681447081e-015},
  bhigh={35635.0851,-106.1158456},
  R=244.3067247305985);

constant IdealGases.Common.DataRecord CH3I(
  name="CH3I",
  MM=0.14193899,
  Hf=96980.82253509061,
  H0=76200.58449056177,
  Tlimit=1000,
  alow={-45164.6274,1086.208429,-5.66317627,0.0368317171,-4.3213935e-005,
         2.833483666e-008,-7.684383690000001e-012},
  blow={-4303.83065,56.88562690000001},
  ahigh={2511915.982,-9960.28998999999,18.56907132,-0.001768191331,
         3.24220626e-007,-3.19294201e-011,1.305407821e-015},
  bhigh={60669.29949999999,-95.9077704},
  R=58.57778754097095);

constant IdealGases.Common.DataRecord CH2OH(
  name="CH2OH",
  MM=0.03103392,
  Hf=-573565.9562182283,
  H0=379616.8837194915,
  Tlimit=1000,
  alow={-156007.6238,2685.446279,-13.4202242,0.0575713947,-7.28444999e-005,
         4.836648860000001e-008,-1.293492601e-011},
  blow={-15968.2041,99.630337},
  ahigh={2250349.506,-8173.186060000001,15.99639179,-0.0008704133719999999,
         6.06918395e-008,4.40834946e-012,-5.7023095e-016},
  bhigh={46453.1343,-78.3515845},
  R=267.9156226477351);

constant IdealGases.Common.DataRecord CH2OHplus(
  name="CH2OHplus",
  MM=0.0310333714,
  Hf=23084826.67790326,
  H0=327035.0446036295,
  Tlimit=1000,
  alow={-107708.0841,2252.082711,-11.88167865,0.0460231696,-4.87973688e-005,
         2.876413471e-008,-7.1002265e-012},
  blow={74844.4777,90.2792857},

```

```
ahigh={2603333.487,-10099.5381,17.30843898,-0.000694639032,3.009715083e-008,
      5.22148598e-012,-5.09018379e-016},
bhigh={146540.2426,-92.2395528},
R=267.9203587915685);

constant IdealGases.Common.DataRecord CH3O (
  name="CH3O",
  MM=0.03103392,
  Hf=418896.4848784814,
  H0=364183.7705323724,
  Tlimit=1000,
  allow={86571.17660000001,-663.1685250000001,2.257455672,0.02266283789,-
2.970566403e-005,
        2.199341353e-008,-6.58804338e-012},
  blow={4174.102129999999,8.174777900000001},
  ahigh={2101188.243,-8841.968800000001,18.22645731,-0.001743485034,
        3.34043427e-007,-3.43067316e-011,1.473897771e-015},
  bhigh={53095.82060000001,-94.2250059},
  R=267.9156226477351);

constant IdealGases.Common.DataRecord CH4 (
  name="CH4",
  MM=0.01604246,
  Hf=-4650159.63885838,
  H0=624355.7409524474,
  Tlimit=1000,
  allow={-176685.0998,2786.18102,-12.0257785,0.0391761929,-3.61905443e-005,
        2.026853043e-008,-4.97670548999999e-012},
  blow={-23313.1436,89.0432275},
  ahigh={3730042.76,-13835.01485,20.49107091,-0.001961974759,4.72731304e-007,
        -3.72881469e-011,1.623737207e-015},
  bhigh={75320.6691,-121.9124889},
  R=518.2791167938085);

constant IdealGases.Common.DataRecord CH3OH (
  name="CH3OH",
  MM=0.03204186,
  Hf=-6271171.523750494,
  H0=356885.5553329301,
  Tlimit=1000,
  allow={-241664.2886,4032.14719,-20.46415436,0.0690369807,-7.59893269e-005,
        4.59820836e-008,-1.158706744e-011},
  blow={-44332.61169999999,140.014219},
  ahigh={3411570.76,-13455.00201,22.61407623,-0.002141029179,3.73005054e-007,
        -3.49884639e-011,1.366073444e-015},
  bhigh={56360.8156,-127.7814279},
  R=259.4878075117987);

constant IdealGases.Common.DataRecord CH3OOH (
  name="CH3OOH",
  MM=0.04804126,
  Hf=-2893346.261109721,
  H0=0,
  Tlimit=1000,
  allow={-149797.4156,2656.222273,-13.77060625,0.0658867383,-7.75180165e-005,
        4.9688007e-008,-1.31436764e-011},
  blow={0,-30584.14201},
```

```

ahigh={13918.692,3060740.61,-12829.59627,25.41021168,-0.002394481095,
       4.44342991e-007,-4.4166595e-011},
bhigh={1.819673372e-015,0},
R=173.069399095694);

constant IdealGases.Common.DataRecord CI(
  name="CI",
  MM=0.13891517,
  Hf=4104669.15888308,
  H0=68344.68834469267,
  Tlimit=1000,
  allow={104301.1064,-1715.427168,12.88874952,-0.01828504834,2.135468356e-005,
         -1.286980869e-008,3.16704766e-012},
  blow={75507.84700000001,-44.9681132},
  ahight=-240822.9894,344.738704,4.9769687,-0.000844481539,
         5.061239719999999e-007,-1.047700525e-010,6.74084314e-015},
  bhigh={64513.4202,1.090913159},
  R=59.85287279999731);

constant IdealGases.Common.DataRecord CI2(
  name="CI2",
  MM=0.26581964,
  Hf=1762073.084592245,
  H0=47562.82492896311,
  Tlimit=1000,
  allow={60282.89810000001,-1043.205435,10.17465942,-0.00553567171,
         5.63031901e-006,-3.108249657e-009,7.20246956e-013},
  blow={59648.9655,-23.10319208},
  ahight=-702865.553,308.2750393,9.04642705,-0.002946736644,1.424347233e-006,
         -2.45318341e-010,1.425270291e-014},
  bhigh={50211.34209999999,-15.60349377},
  R=31.27862185051489);

constant IdealGases.Common.DataRecord CI3(
  name="CI3",
  MM=0.39272411,
  Hf=1033763.755935433,
  H0=42858.20649004717,
  Tlimit=1000,
  allow={85967.84250000001,-1347.703793,12.72103659,-0.002442876357,
         2.511609978e-007,1.024658583e-009,-4.87269296e-013},
  blow={53107.1651,-32.384929},
  ahight=-153651.1918,-83.5704437,10.06203125,-2.466447336e-005,
         5.41905475e-009,-6.181417e-013,2.851262022e-017},
  bhigh={45825.8953,-14.94825345},
  R=21.17127975667193);

constant IdealGases.Common.DataRecord CI4(
  name="CI4",
  MM=0.51962858,
  Hf=515644.0009516028,
  H0=43022.17172119363,
  Tlimit=1000,
  allow={89190.4387,-1651.906475,17.96124325,-0.00851811425,8.50868225e-006,-
4.59909773e-009,
         1.039492403e-012},
  blow={36893.9215,-58.0595312},

```

---

**994 Modelica.Media.IdealGases.Common.SingleGasesData**

---

```
ahigh={-144676.277,-52.8036517,13.03985589,-1.606214814e-005,
       3.56755047e-009,-4.10536698e-013,1.907272766e-017},
bhigh={28177.46618,-28.10414436},
R=16.0007981085259);

constant IdealGases.Common.DataRecord CN(
  name="CN",
  MM=0.0260174,
  Hf=16861160.30041434,
  H0=333319.3939440528,
  Tlimit=1000,
  allow={3949.14857,-139.1590572,4.93083532,-0.006304670510000001,
         1.256836472e-005,-9.878300500000001e-009,2.843137221e-012},
  blow={52284.55379999999,-2.763115585},
  ahigh={-2228006.27,5040.73338999999,-0.2121897722,0.001354901134,
         1.325929798e-007,-6.93700637e-011,5.49495227e-015},
  bhigh={17844.96132,32.82563919},
  R=319.5735161853222);

constant IdealGases.Common.DataRecord CNplus(
  name="CNplus",
  MM=0.0260168514,
  Hf=69143297.79352164,
  H0=333710.7886929008,
  Tlimit=1000,
  allow={-830290.9570000001,8775.6875,-29.7744356,0.0497689706,-1.302225951e-
005,
         -2.058325353e-008,1.126843895e-011},
  blow={170386.0539,203.9918818},
  ahigh={-7153463.08,18572.50421,-10.84534159,0.00610668143,-1.191208566e-006,
         1.184848778e-010,-4.799838730000001e-015},
  bhigh={92426.44959999999,113.5340573},
  R=319.5802548189978);

constant IdealGases.Common.DataRecord CNminus(
  name="CNminus",
  MM=0.0260179486,
  Hf=2455424.32964911,
  H0=333273.93075102,
  Tlimit=1000,
  allow={-46065.4139,429.417475,2.32878188,-0.0001235303004,4.47846277e-006,-
4.40315129e-009,
         1.349001191e-012},
  blow={4362.07834,11.42928617},
  ahigh={351796.472,-1630.477359,5.60987575,-0.000397560597,
         8.85614707999999e-008,-9.722872320000001e-012,4.43420569e-016},
  bhigh={16479.76581,-11.75502699},
  R=319.5667778358206);

constant IdealGases.Common.DataRecord CNN(
  name="CNN",
  MM=0.0400241,
  Hf=15827565.29191162,
  H0=259285.2806184274,
  Tlimit=1000,
  allow={-73576.9236,965.2438659999999,-1.704121157,0.02037239025,-
2.183423906e-005,
```

```

    1.176082777e-008,-2.551355221e-012},
blow={70217.2983,35.2815091},
ahigh={-181714.8765,-672.986349,7.85794834,-6.13688072e-005,-1.088178985e-
008,
        4.45665581e-012,-2.836496278e-016},
bhigh={77234.8898,-19.66012324},
R=207.7366386751982);

constant IdealGases.Common.DataRecord CO (
  name="CO",
  MM=0.0280101,
  Hf=-3946262.098314536,
  H0=309570.6191695138,
  Tlimit=1000,
  alow={14890.45326,-292.2285939,5.72452717,-0.008176235030000001,
         1.456903469e-005,-1.087746302e-008,3.027941827e-012},
  blow={-13031.31878,-7.85924135},
  ahigh={461919.725,-1944.704863,5.91671418,-0.0005664282830000001,
         1.39881454e-007,-1.787680361e-011,9.62093557e-016},
  bhigh={-2466.261084,-13.87413108},
  R=296.8383547363272);

constant IdealGases.Common.DataRecord COplus (
  name="COplus",
  MM=0.0280095514,
  Hf=44548703.62543543,
  H0=309576.6824741077,
  Tlimit=1000,
  alow={-21787.86658,128.8857032,3.76905755,-0.00343173013,8.19394575e-006,-
6.463814690000001e-009,
         1.803727574e-012},
  blow={148234.5898,3.99054707},
  ahigh={231684.7506,-1057.646148,4.55425778,0.000449552032,-2.489507047e-007,
         5.26756642e-011,-3.28951027e-015},
  bhigh={155505.0724,-3.87346264},
  R=296.8441686645506);

constant IdealGases.Common.DataRecord COCL (
  name="COCL",
  MM=0.0634631,
  Hf=-252115.0085640317,
  H0=182007.3239409988,
  Tlimit=1000,
  alow={25131.7574,-596.918967,8.327671349999999,-0.00705613259,
         1.313150734e-005,-1.037059653e-008,3.033665179e-012},
  blow={-705.277032,-15.80716775},
  ahigh={344372.024,-1793.14347,8.392755899999999,-0.000537476959,
         9.113555710000001e-008,-3.111441728e-012,-2.040435218e-016},
  bhigh={6914.470149999999,-19.98919104},
  R=131.0126987178376);

constant IdealGases.Common.DataRecord COCL2 (
  name="COCL2",
  MM=0.09891610000000001,
  Hf=-2219052.307966044,
  H0=130197.4299431538,
  Tlimit=1000,

```

---

**996 Modelica.Media.IdealGases.Common.SingleGasesData**

---

```
alow={93193.2145,-1577.971273,12.08353907,-0.00480901561,
      7.68847731999999e-006,-5.8588412e-009,1.687786559e-012},
blow={-20542.52334,-38.3476732},
ahigh={-25458.81891,-1305.958516,10.92922584,-0.000360121016,
       7.78701765e-008,-8.79245203e-012,4.02869661e-016},
bhigh={-22198.4734,-32.3330345},
R=84.05580082514373);

constant IdealGases.Common.DataRecord COFCL(
  name="COFCL",
  MM=0.0824615031999999,
  Hf=-5208404.471578928,
  H0=144355.821056631,
  Tlimit=1000,
  alow={72621.739,-1019.738175,7.29091199,0.00742069171,-7.902548079999999e-
006,
        4.209427650000001e-009,-9.3144129e-013},
  blow={-48043.8646,-13.16923671},
  ahigh={-53168.8791,-1581.009678,11.12696501,-0.00043730008,
         9.464190710000001e-008,-1.069291346e-011,4.901756640000001e-016},
  bhigh={-45996.6723,-35.25879824},
  R=100.8285281901095);

constant IdealGases.Common.DataRecord COF2 (
  name="COF2",
  MM=0.06600690640000001,
  Hf=-9695955.088723866,
  H0=168678.2430391253,
  Tlimit=1000,
  alow={52633.9315,-461.860339,2.774114516,0.01831931082,-2.130172554e-005,
        1.266924542e-008,-3.101675983e-012},
  blow={-75642.55099999999,9.467181460000001},
  ahigh={-40738.0685,-1974.009812,11.40363654,-0.000543711147,
         1.175232744e-007,-1.326564192e-011,6.07677213e-016},
  bhigh={-69077.9541,-40.09695},
  R=125.9636673413314);

constant IdealGases.Common.DataRecord COHCL (
  name="COHCL",
  MM=0.06447104000000001,
  Hf=-2547064.170207274,
  H0=170716.6814743488,
  Tlimit=1000,
  alow={6332.94687,81.2036559,1.374648391,0.01650970564,-1.692917241e-005,
        9.68404683e-009,-2.37446939e-012},
  blow={-21203.5658,19.3837965},
  ahigh={831895.285,-4416.87084,12.70661114,-0.0009361408629999999,
         1.855335611e-007,-1.958378539e-011,8.512046829999999e-016},
  bhigh={4330.49644,-51.45071542},
  R=128.9644466724905);

constant IdealGases.Common.DataRecord COHF (
  name="COHF",
  MM=0.0480164432,
  Hf=-7801293.391093992,
  H0=217629.5098842307,
  Tlimit=1000,
```

```

alow={-45858.285,1048.202675,-4.77657422,0.03080719931,-3.4158959e-005,
      2.034926806e-008,-5.05419462e-012},
blow={-50859.8373,52.3224741},
ahigh={857885.316,-4791.95912,12.93975218,-0.001018285299,2.021278474e-007,
      -2.136698091e-011,9.299588910000002e-016},
bhigh={-18789.33107,-55.2744914},
R=173.1588482172291);

constant IdealGases.Common.DataRecord COS (
  name="COS",
  MM=0.0600751,
  Hf=-2358714.342547911,
  H0=165486.1498357889,
  Tlimit=1000,
  alow={85478.76430000001,-1319.464821,9.735257239999999,-0.00687083096,
        1.082331416e-005,-7.70559734e-009,2.078570344e-012},
  blow={-11916.57685,-29.91988593},
  ahigh={195909.8567,-1756.167688,8.71043034,-0.000413942496,1.015243648e-007,
        -1.159609663e-011,5.691053860000001e-016},
  bhigh={-8927.09669,-26.36328016},
  R=138.4013010382005);

constant IdealGases.Common.DataRecord CO2 (
  name="CO2",
  MM=0.0440095,
  Hf=-8941478.544405185,
  H0=212805.6215135368,
  Tlimit=1000,
  alow={49436.5054,-626.411601,5.30172524,0.002503813816,-2.127308728e-007,-
7.68998878e-010,
        2.849677801e-013},
  blow={-45281.9846,-7.04827944},
  ahigh={117696.2419,-1788.791477,8.29152319,-9.22315678e-005,4.86367688e-009,
        -1.891053312e-012,6.330036589999999e-016},
  bhigh={-39083.5059,-26.52669281},
  R=188.9244822140674);

constant IdealGases.Common.DataRecord CO2plus (
  name="CO2plus",
  MM=0.0440089514,
  Hf=21465814.54335674,
  H0=240089.5423288817,
  Tlimit=1000,
  alow={-73830.3098,1086.211742,-2.771112737,0.02318463595,-2.570240315e-005,
        1.450335497e-008,-3.33447042e-012},
  blow={107178.4918,40.54885210000001},
  ahigh={-169505.1682,-806.646973,8.00282846,-0.0001577214041,
        2.566759314e-008,-2.404195965e-012,1.6774468e-016},
  bhigh={115438.9478,-21.33567772},
  R=188.9268372797449);

constant IdealGases.Common.DataRecord COOH (
  name="COOH",
  MM=0.04501744,
  Hf=-4731499.614371675,
  H0=240203.5522233161,
  Tlimit=1000,

```

---

**998 Modelica.Media.IdealGases.Common.SingleGasesData**

---

```
alow={-11283.80671,377.517943,-0.5992550409999999,0.02181894272,-  
2.425918417e-005,  
    1.451245206e-008,-3.59623338e-012},  
blow={-28410.42565,29.34561769},  
ahigh={929318.873,-4483.030570000001,12.42199567,-0.000746313963999999,  
    1.332996131e-007,-1.28271055e-011,5.1379979e-016},  
bhigh={-851.8232680000001,-50.6806551},  
R=184.694465078423);  
  
constant IdealGases.Common.DataRecord CP (  
    name="CP",  
    MM=0.042984461,  
    Hf=12101159.18401303,  
    H0=202750.1287034866,  
    Tlimit=1000,  
    alow={-45239.4011,775.8828140000001,-1.453947907,0.01397713706,-  
1.624489706e-005,  
    9.51408387999999e-009,-2.210281109e-012},  
    blow={57926.3467,33.1164792},  
    ahigh={-5449937.15,16883.45926,-15.956922,0.01136028761,-2.84654133e-006,  
    3.37936404e-010,-1.554457397e-014},  
    bhigh={-45545.17739999999,145.1324059},  
    R=193.429714054109);  
  
constant IdealGases.Common.DataRecord CS (  
    name="CS",  
    MM=0.0440757,  
    Hf=6319810.643960278,  
    H0=197577.7582658926,  
    Tlimit=1000,  
    alow={-49248.4412,816.69681,-1.542998408,0.01380324735,-1.574407905e-005,  
    9.16971493e-009,-2.169700595e-012},  
    blow={28651.82876,33.08541327},  
    ahigh={-971957.476,2339.201284,1.709390402,0.001577178949,-  
4.146335910000001e-007,  
    4.50475708e-011,-5.94545773e-016},  
    bhigh={16810.20727,18.7404822},  
    R=188.6407249346012);  
  
constant IdealGases.Common.DataRecord CS2 (  
    name="CS2",  
    MM=0.0761406999999999,  
    Hf=1532688.824767831,  
    H0=140059.4294510032,  
    Tlimit=1000,  
    alow={16135.60482,-464.948147,6.29793879,0.001888896706,3.031927747e-007,-  
1.737645373e-009,  
    7.79398939e-013},  
    blow={14777.61119,-9.30338212999999},  
    ahigh={-1390419.724,3354.9755,3.019247723,0.002876437543,-  
9.07681271999999e-007,  
    1.374091042e-010,-6.99957557e-015},  
    bhigh={-10138.98046,15.65113703},  
    R=109.1987859318341);  
  
constant IdealGases.Common.DataRecord C2 (  
    name="C2",
```

```

MM=0.0240214,
Hf=34571562.10712116,
H0=423335.9421182778,
Tlimit=1000,
allow={555963.451,-9980.12644,66.8162037,-0.1743432724,0.0002448523051,-
1.70346758e-007,
        4.68452773e-011},
blow={144586.9634,-344.82297},
ahigh={-968926.793,3561.09299,-0.5064138930000001,0.002945154879,-
7.13944119e-007,
        8.67065725e-011,-4.07690681e-015},
bhigh={76817.96829999999,33.3998524},
R=346.1277027983382);

constant IdealGases.Common.DataRecord C2plus(
  name="C2plus",
  MM=0.0240208514,
  Hf=83459803.26076201,
  H0=361565.2024723819,
  Tlimit=1000,
  allow={-99134.2384,1347.170609,-3.47675316,0.01676429424,-1.865908025e-005,
         1.091134647e-008,-2.434913818e-012},
  blow={233545.48,44.0664462},
  ahigh={3836292.81,-6242.06244999999,2.779245639,0.006065865859999999,-
2.452799858e-006,
         3.8829425e-010,-2.190639912e-014},
  bhigh={285744.7553,0.729738349},
  R=346.1356078327848);

constant IdealGases.Common.DataRecord C2minus(
  name="C2minus",
  MM=0.0240219486,
  Hf=20013651.34883354,
  H0=361174.0306529504,
  Tlimit=1000,
  allow={-118192.866,1438.18971,-3.19613135,0.01465548163,-1.545537278e-005,
         9.061235610000001e-009,-2.135962274e-012},
  blow={49653.1779,42.2560888},
  ahigh={4478136.25,-11541.45714,13.10143499,-0.001862700578,4.00693125e-008,
         3.7102136e-011,-3.33726687e-015},
  bhigh={132535.6168,-69.75964399999999},
  R=346.1197981249531);

constant IdealGases.Common.DataRecord C2CL(
  name="C2CL",
  MM=0.0594744,
  Hf=8980049.752498554,
  H0=181264.9812356241,
  Tlimit=1000,
  allow={47258.8941,-898.0223609999999,9.016877149999999,-0.00633378283,
         1.08706005e-005,-8.089068920000001e-009,2.248275223e-012},
  blow={67022.15700000001,-23.54893403},
  ahigh={213736.8408,-1630.519518,8.62349025,-0.000425351786,9.04001864e-008,
         -1.007541495e-011,4.570764750000001e-016},
  bhigh={71710.9328,-24.13014598},
  R=139.7991740984356);

```

---

**1000 Modelica.Media.IdealGases.Common.SingleGasesData**

---

```
constant IdealGases.Common.DataRecord C2CL2 (
  name="C2CL2",
  MM=0.09492680000000001,
  Hf=2387102.48317651,
  H0=0,
  Tlimit=1000,
  allow={38711.6908,-933.687072999999,11.19377861,-0.0037462882,
    6.77047256e-006,-4.91038835e-009,1.299068749e-012},
  blow={0,29481.53198},
  ahigh={14592.612,296197.8472,-2044.286552,11.87529315,-0.000511501145,
    1.072661556e-007,-1.183411171e-011},
  bhigh={5.32649834e-016,0},
  R=87.58824694396103);

constant IdealGases.Common.DataRecord C2CL3 (
  name="C2CL3",
  MM=0.1303804,
  Hf=1459357.326714752,
  H0=123870.2903197106,
  Tlimit=1000,
  allow={46750.1604,-885.1019640000001,9.03227055,0.01242122796,-1.554614347e-
005,
    9.51388712e-009,-2.341473263e-012},
  blow={24958.63399,-17.79070154},
  ahigh={-220402.8219,-1072.926248,13.78367377,-0.0003093322915,
    6.77752895e-008,-7.72730182e-012,3.56669022e-016},
  bhigh={24310.20015,-43.42161815},
  R=63.77087353620636);

constant IdealGases.Common.DataRecord C2CL4 (
  name="C2CL4",
  MM=0.1658322,
  Hf=-145930.6455561706,
  H0=0,
  Tlimit=1000,
  allow={37462.583,-848.177563,10.0915611,0.01845463613,-2.390899444e-005,
    1.514828257e-008,-3.84111932e-012},
  blow={0,-1598.288471},
  ahigh={19605.643,-300131.5659,-1132.407598,16.8307247,-0.000328879249,
    7.22090703e-008,-8.24533269e-012},
  bhigh={3.81011278e-016,0},
  R=50.13786224870682);

constant IdealGases.Common.DataRecord C2CL6 (
  name="C2CL6",
  MM=0.2367376,
  Hf=-626009.5565723401,
  H0=0,
  Tlimit=1000,
  allow={159345.1813,-2606.697136,20.28011847,0.01769364822,-3.133105185e-005,
    2.389395238e-008,-6.8930383e-012},
  blow={0,-9038.00764},
  ahigh={27129.39,-557738.632,-497.649124,22.37041989,-0.0001477759098,
    3.25754658e-008,-3.72709921e-012},
  bhigh={1.723801704e-016,0},
  R=35.12104541061496);
```

```

constant IdealGases.Common.DataRecord C2F(
  name="C2F",
  MM=0.0430198032,
  Hf=8225223.16884983,
  H0=240982.8783224188,
  Tlimit=1000,
  allow={12497.97213,-348.387675,5.75508435,0.000852868538,2.353546435e-006,-
2.804737097e-009,
         9.083575639999999e-013},
  blow={42815.2531,-6.43720096},
  ahighe={289877.6676,-2016.644658,8.87504571,-0.000516684113,1.092094306e-007,
         -1.212216787e-011,5.48228768e-016},
  bhigh={52412.8792,-27.7308958},
  R=193.2708051067979);

constant IdealGases.Common.DataRecord C2FCL(
  name="C2FCL",
  MM=0.0784728032,
  Hf=430294.2882509389,
  H0=176832.0543441476,
  Tlimit=1000,
  allow={26976.75344,-741.4403179999999,9.6296043,-0.000386554493,
         3.003347997e-006,-2.710191661e-009,7.73758689e-013},
  blow={5500.62039,-25.15774889},
  ahighe={347145.489,-2348.4611,12.06875617,-0.000580469823,1.21273434e-007,-
1.334214186e-011,
         5.99255661e-016},
  bhigh={15020.61224,-42.52445687},
  R=105.9535490125068);

constant IdealGases.Common.DataRecord C2FCL3(
  name="C2FCL3",
  MM=0.1493788032,
  Hf=-1111268.777389696,
  H0=125398.7419816201,
  Tlimit=1000,
  allow={12372.82217,-470.396184,7.58162179,0.02305727066,-2.792144322e-005,
         1.679699826e-008,-4.07757244e-012},
  blow={-20313.33976,-9.274637513},
  ahighe={-282950.5545,-1473.853145,17.07879074,-0.000426521339,
         9.357199850000001e-008,-1.067932796e-011,4.93326161e-016},
  bhigh={-17384.33314,-61.97054796},
  R=55.66032008482446);

constant IdealGases.Common.DataRecord C2F2(
  name="C2F2",
  MM=0.0620182064,
  Hf=-2332632.567071465,
  H0=213911.684488831,
  Tlimit=1000,
  allow={17763.13572,-611.328094,8.6636068,0.0009156381209999999,
         2.363808425e-006,-2.753750054e-009,8.713935969999999e-013},
  blow={-16496.11017,-21.65150307},
  ahighe={419389.094,-2715.359989,12.30403413,-0.0006649305540000001,
         1.385254301e-007,-1.520813764e-011,6.81982103e-016},
  bhigh={-4179.221939999999,-46.7038623},
  R=134.0650186878026);

```

---

**1002 Modelica.Media.IdealGases.Common.SingleGasesData**

---

```
constant IdealGases.Common.DataRecord C2F2CL2 (
  name="C2F2CL2",
  MM=0.1329242064,
  Hf=-2541576.204588121,
  H0=134934.2567901161,
  Tlimit=1000,
  allow={-115043.3214,2032.113595,-11.19750271,0.08342451400000001,-
0.0001163698616,
  7.83309289e-008,-2.070004069e-011},
  blow={-52082.5681,88.88322147},
  ahighelement1={-870256.566,726.472448,15.30689121,0.0003045597224,-7.04546804e-008,
  8.28592434e-012,-3.89858107e-016},
  bhighelement1={-52088.8871,-49.98300753},
  R=62.55047312435939);
```

```
constant IdealGases.Common.DataRecord C2F3 (
  name="C2F3",
  MM=0.0810166096,
  Hf=-2816472.993952588,
  H0=174823.0032079743,
  Tlimit=1000,
  allow={-26147.75016,210.1767714,2.32862508,0.02368943293,-2.416431591e-005,
  1.224916093e-008,-2.488670434e-012},
  blow={-30285.66585,16.9960879},
  ahighelement1={-124260.3837,-2137.292446,14.55051371,-0.0006093891980000001,
  1.331427672e-007,-1.515124358e-011,6.98411206e-016},
  bhighelement1={-19771.46179,-54.21039870000001},
  R=102.6267581555277);
```

```
constant IdealGases.Common.DataRecord C2F3CL (
  name="C2F3CL",
  MM=0.1164696096,
  Hf=-4423471.511318606,
  H0=147061.1008212738,
  Tlimit=1000,
  allow={-8122.270309999999,-157.4007092,5.04181029,0.02621692321,-
2.843001445e-005,
  1.537643979e-008,-3.36073067e-012},
  blow={-63540.49950000001,2.792301872},
  ahighelement1={-216272.4659,-2195.103372,17.59605428,-0.000628261729,
  1.37416661e-007,-1.565004037e-011,7.218309900000001e-016},
  bhighelement1={-55151.095,-69.21931823999999},
  R=71.38748063597872);
```

```
constant IdealGases.Common.DataRecord C2F4 (
  name="C2F4",
  MM=0.1000150128,
  Hf=-6594010.054458545,
  H0=163281.0269459867,
  Tlimit=1000,
  allow={-9991.530069999999,-129.1088427,4.50422905,0.0259202964,-2.63030872e-
005,
  1.316489777e-008,-2.625017169e-012},
  blow={-80904.47080000001,3.27424147},
  ahighelement1={-162991.5758,-2603.903955,17.88488423,-0.000739703801,
  1.61446034e-007,-1.835820392e-011,8.45764107e-016},
  bhighelement1={-70063.7166,-74.54165860000001},
  R=83.13223952314488);
```

```

constant IdealGases.Common.DataRecord C2F6(
  name="C2F6",
  MM=0.1380118192,
  Hf=-9738296.384980919,
  H0=146886.2603037118,
  Tlimit=1000,
  allow={-37953.717,799.82764,-4.67156181,0.0750145099,-9.812116100000001e-005,
         6.31980759e-008,-1.622597311e-011},
  blow={-167521.1878,50.537592},
  ahighelement={-1011551.484,-942.2140710000001,22.02553906,-0.000131451875,
                1.8660453e-008,-3.46711861e-012,2.488311205e-016},
  bhigh={-165743.4402,-93.02161150000001},
  R=60.24463736653651);

constant IdealGases.Common.DataRecord C2H(
  name="C2H",
  MM=0.02502934,
  Hf=22621470.72196071,
  H0=417457.3520516323,
  Tlimit=1000,
  allow={13436.69487,-506.797072,7.77210741,-0.00651233982,1.030117855e-005,-
5.880147670000001e-009,
        1.226901861e-012},
  blow={68922.69989999999,-18.71881626},
  ahighelement={3922334.57,-12047.51703,17.5617292,-0.00365544294,6.98768543e-007,-
6.82516201e-011,
                2.719262793e-015},
  bhigh={143326.6627,-95.6163438},
  R=332.1890229626511);

constant IdealGases.Common.DataRecord C2HCL(
  name="C2HCL",
  MM=0.06048204,
  Hf=3743259.982632861,
  H0=0,
  Tlimit=1000,
  allow={132978.5007,-2174.542126,14.97980854,-0.01290343389,1.602596678e-005,
         -9.152049509999999e-009,2.021223607e-012},
  blow={0,36048.0132},
  ahighelement={11787.625,1152145.326,-4461.192999999999,12.81366744,-0.000679734676,
            1.151382756e-007,-1.046641954e-011},
  bhigh={3.94986052e-016,0},
  R=137.4700985614903);

constant IdealGases.Common.DataRecord C2HCL3(
  name="C2HCL3",
  MM=0.13138744,
  Hf=-133193.8577994974,
  H0=0,
  Tlimit=1000,
  allow={39592.3125,-517.875358,5.15502877,0.02816922486,-3.62474172e-005,
         2.39207427e-008,-6.36240196e-012},
  blow={0,-1534.346259},
  ahighelement={16604.624,604925.466,-4236.992319999999,18.46273557,-0.000813719506,
            1.551211545e-007,-1.584630634e-011},
  bhigh={6.70093249e-016,0},
  R=63.28209149976589);

```

---

**1004 Modelica.Media.IdealGases.Common.SingleGasesData**

---

```
constant IdealGases.Common.DataRecord C2HF (
  name="C2HF",
  MM=0.0440277432,
  Hf=946955.282504691,
  H0=259970.3543287678,
  Tlimit=1000,
  allow={91611.6911,-1537.245636,11.33665074,-0.00462472931,5.87798546e-006,-
2.690992345e-009,
  3.52471953e-013},
  blow={10859.0282,-40.23994380000001},
  ahighe={1234347.179,-4819.58196,13.02337752,-0.000749543582,1.285452942e-007,
  -1.184559365e-011,4.53678476e-016},
  bhigh={32368.1171,-56.4079182},
  R=188.8462000477917);

constant IdealGases.Common.DataRecord C2HFCL2 (
  name="C2HFCL2",
  MM=0.1149337432,
  Hf=-1467349.755645999,
  H0=141466.6358834818,
  Tlimit=1000,
  allow={31173.99823,-136.9505457,1.176399869,0.040807766,-5.4181378e-005,
  3.61360066e-008,-9.625862350000001e-012},
  blow={-21151.42871,21.4643316},
  ahighe={407124.352,-3808.20452,18.2114405,-0.000727896606,1.381532849e-007,-
1.405342513e-011,
  5.92000972e-016},
  bhigh={-2676.849943,-75.67353566999999},
  R=72.34143575687527);

constant IdealGases.Common.DataRecord C2HF2CL (
  name="C2HF2CL",
  MM=0.09847914640000001,
  Hf=-3388067.54726298,
  H0=154984.6293143743,
  Tlimit=1000,
  allow={170443.7675,-1667.236861,4.7887885,0.0444483001,-7.32219777e-005,
  5.62934755e-008,-1.654638029e-011},
  blow={-32918.0976,-5.78859364},
  ahighe={583037.47,-3810.31741,17.99003171,-0.000588821086,1.002861284e-007,-
9.149788100000001e-012,
  3.45998516e-016},
  bhigh={-22058.75409,-75.90948533},
  R=84.42875780247421);

constant IdealGases.Common.DataRecord C2HF3 (
  name="C2HF3",
  MM=0.0820245496,
  Hf=-5986012.75343059,
  H0=174673.461419409,
  Tlimit=1000,
  allow={-14109.69036,368.762492,-1.144797568,0.0395630602,-4.60200136e-005,
  2.785149385e-008,-6.90376008e-012},
  blow={-62264.1033,32.8966169},
  ahighe={691741.378,-5260.30164,19.15276614,-0.001072493938,2.099276744e-007,
  -2.195126633e-011,9.47270239999999e-016},
  bhigh={-32476.4021,-87.5045695},
  R=101.3656526070093);
```

```

constant IdealGases.Common.DataRecord C2H2_vinylidene(
  name="C2H2_vinylidene",
  MM=0.02603728,
  Hf=15930556.80163212,
  H0=417638.4015534649,
  Tlimit=1000,
  allow={-14660.42239, 278.9475593, 1.276229776, 0.01395015463, -1.475702649e-005,
         9.476298110000001e-009, -2.567602217e-012},
  blow={47361.1018, 16.58225704},
  ahighe={1940838.725, -6892.718150000001, 13.39582494, -0.0009368968669999999,
          1.470804368e-007, -1.220040365e-011, 4.12239166e-016},
  bhigh={91071.1293, -63.3750293},
  R=319.3295152181795);

constant IdealGases.Common.DataRecord C2H2CL2 (
  name="C2H2CL2",
  MM=0.09694328000000001,
  Hf=35175.20760593204,
  H0=153517.366030941,
  Tlimit=1000,
  allow={-12037.24514, 462.903119, -1.318184584, 0.039304496, -4.85152902e-005,
         3.17841146e-008, -8.476938410000001e-012},
  blow={-3251.80686, 34.89218118},
  ahighe={1561121.185, -7358.096350000001, 20.11869185, -0.001310978834,
          2.411885716e-007, -2.38416674e-011, 9.78541575e-016},
  bhigh={41222.009, -95.50252712000001},
  R=85.76635739991467);

constant IdealGases.Common.DataRecord C2H2FCL (
  name="C2H2FCL",
  MM=0.0804886831999999,
  Hf=-2050996.406411579,
  H0=167344.1838591292,
  Tlimit=1000,
  allow={226015.4427, -2557.428772, 9.2362412, 0.02879408894, -5.16397547e-005,
         4.25219103e-008, -1.311612713e-011},
  blow={-8180.627839999999, -32.34787581},
  ahighe={1527690.968, -6689.71229999999, 19.5055383, -0.001040920895,
          1.780587502e-007, -1.633098515e-011, 6.213465560000001e-016},
  bhigh={17052.66451, -91.94078300999999},
  R=103.2998884991077);

constant IdealGases.Common.DataRecord C2H2F2 (
  name="C2H2F2",
  MM=0.06403408640000001,
  Hf=-5253452.011458697,
  H0=194891.2477964236,
  Tlimit=1000,
  allow={57029.3581, -676.28604, 4.11741171, 0.01822618762, -6.86133154e-006, -
3.29575674e-009,
         2.180965597e-012},
  blow={-38386.5078, 1.487528685},
  ahighe={-822719.275, -2915.718833, 18.30163995, -0.001039630129,
          2.546725128e-007, -3.157578705e-011, 1.55154096e-015},
  bhigh={-31100.28589, -84.0083470000002},
  R=129.8444698353657);

```

---

**1006 Modelica.Media.IdealGases.Common.SingleGasesData**

---

```
constant IdealGases.Common.DataRecord CH2CO_ketene (
  name="CH2CO_ketene",
  MM=0.04203668,
  Hf=-1179353.245784396,
  H0=0,
  Tlimit=1000,
  allow={35495.9809,-406.306283,3.71892192,0.01583501817,-1.726195691e-005,
         1.157376959e-008,-3.30584263e-012},
  blow={0,-5209.99258},
  ahigh={11795.805,2013564.915,-8200.88746,17.59694074,-0.001464544521,
         2.695886969e-007,-2.66567484e-011},
  bhigh={1.094204522e-015,0},
  R=197.7908816776206);

constant IdealGases.Common.DataRecord O_CH_2O (
  name="O_CH_2O",
  MM=0.05803608,
  Hf=-3652900.058032865,
  H0=0,
  Tlimit=1000,
  allow={-229245.9698,3724.09805,-18.93769993,0.0751174414,-8.083855420000001e-
005,
         4.35823319e-008,-9.36453933e-012},
  blow={0,-44544.8601},
  ahigh={13682.443,267806.3593,-4436.61748,17.81696797,-0.000709717378,
         1.272621878e-007,-1.237226678e-011},
  bhigh={5.02505752e-016,0},
  R=143.2638455250596);

constant IdealGases.Common.DataRecord HO_CO_2OH (
  name="HO_CO_2OH",
  MM=0.09003488,
  Hf=-8127961.074641295,
  H0=0,
  Tlimit=1000,
  allow={30725.13274,-391.617469,3.17838864,0.0374128975,-4.00975396e-005,
         2.288662646e-008,-5.386771549999999e-012},
  blow={0,-87979.4255},
  ahigh={17321.662,1805560.696,-9240.33315,26.25742604,-0.001418458557,
         2.526840574e-007,-2.521005198e-011},
  bhigh={1.040036111e-015,0},
  R=92.34723253921148);

constant IdealGases.Common.DataRecord C2H3_vinyl (
  name="C2H3_vinyl",
  MM=0.02704522,
  Hf=11080953.19616553,
  H0=389044.3856622353,
  Tlimit=1000,
  allow={-33478.9687,1064.104103,-6.40385706,0.0393451548,-4.76004609e-005,
         3.17007135e-008,-8.633406430000001e-012},
  blow={30391.22649,58.0922618},
  ahigh={2718080.093,-10309.56829,18.36579807,-0.001580131153,
         2.680594939e-007,-2.439003999e-011,9.20909639e-016},
  bhigh={97650.55589999999,-97.6008686},
  R=307.4285215649937);
```

```

constant IdealGases.Common.DataRecord CH2BrminusCOOH (
  name="CH2BrminusCOOH",
  MM=0.13894802,
  Hf=-2760024.935943672,
  H0=0,
  Tlimit=1000,
  allow={-78324.8478999999,1648.865353,-9.34953363,0.06879635440000001,-
8.374880770000001e-005,
           5.40603842e-008,-1.423159923e-011},
  blow={0,-55411.8762},
  ahigh={16862.437,2486349.85,-11402.56346,27.56020307,-0.00184318701,
         3.26539274e-007,-3.122663159e-011},
  bhigh={1.223452495e-015,0},
  R=59.83872242296076);

constant IdealGases.Common.DataRecord C2H3CL (
  name="C2H3CL",
  MM=0.06249792,
  Hf=352011.7149498736,
  H0=0,
  Tlimit=1000,
  allow={-16888.95457,854.510455,-6.51496755,0.0494468193,-6.11775667e-005,
         4.067293850000001e-008,-1.101392611e-011},
  blow={0,-2069.321321},
  ahigh={11819.647,2456176.938,-10474.52327,21.78736181,-0.001816891849,
         3.29636995e-007,-3.21439569e-011},
  bhigh={1.302254061e-015,0},
  R=133.0359794373957);

constant IdealGases.Common.DataRecord CH2CLminusCOOH (
  name="CH2CLminusCOOH",
  MM=0.0944967199999999,
  Hf=-4525024.783929008,
  H0=0,
  Tlimit=1000,
  allow={-112250.6518,2168.288658,-12.27791286,0.0751438825,-
9.083158770000001e-005,
           5.81236419e-008,-1.518247503e-011},
  blow={0,-63143.1455},
  ahigh={16513.941,2472883.176,-11522.75963,27.66275606,-0.001886430963,
         3.36177605e-007,-3.23127225e-011},
  bhigh={1.273275883e-015,0},
  R=87.98688462414359);

constant IdealGases.Common.DataRecord C2H3F (
  name="C2H3F",
  MM=0.0460436232,
  Hf=-3042766.625715937,
  H0=246199.7386860728,
  Tlimit=1000,
  allow={-45791.3496,1411.541724,-10.21500877,0.0577581657,-7.08067749e-005,
         4.64085076e-008,-1.240455865e-011},
  blow={-24027.88827,78.6082839},
  ahigh={2478832.34,-10759.32307,21.94796688,-0.00186911411,3.39528067e-007,-
3.31538293e-011,
           1.345081799e-015},
  bhigh={45634.5555,-118.0410277},
  R=180.5781435549581);

```

## 1008 Modelica.Media.IdealGases.Common.SingleGasesData

---

```
constant IdealGases.Common.DataRecord CH3CN (
  name="CH3CN",
  MM=0.04105192000000001,
  Hf=1618194.715375066,
  H0=294594.9422097675,
  Tlimit=1000,
  allow={-99659.88380000001,1739.278534,-7.89842082,0.0429489432,-4.49997388e-005,
         2.717105086e-008,-7.026117590000001e-012},
  blow={-1461.161333,68.52508274},
  ahighelement1={2923231.393,-12337.92258,23.24477222,-0.002411565845,
                 4.622157170000001e-007,-4.74060124e-011,2.010639467e-015},
  bhighelement1={80585.65550000001,-129.2249102},
  R=202.5355208721054);

constant IdealGases.Common.DataRecord CH3CO_acetyl (
  name="CH3CO_acetyl",
  MM=0.04304462,
  Hf=-232317.0700542832,
  H0=302859.3817299352,
  Tlimit=1000,
  allow={-71938.94130000001,1464.465167,-6.63227613,0.04108468379999999,-4.22625664e-005,
         2.485766819e-008,-6.29255848e-012},
  blow={-9309.37081,64.22897619999999},
  ahighelement1={2485388.15,-11207.14204,22.77525438,-0.00231426055,4.53618917e-007,-4.74263555e-011,
                 2.044663903e-015},
  bhighelement1={63800.8841,-121.5350925},
  R=193.1593774088376);

constant IdealGases.Common.DataRecord C2H4 (
  name="C2H4",
  MM=0.02805316,
  Hf=1871446.924339362,
  H0=374955.5843263291,
  Tlimit=1000,
  allow={-116360.5836,2554.85151,-16.09746428,0.0662577932,-7.885081859999999e-005,
         5.12522482e-008,-1.370340031e-011},
  blow={-6176.19107,109.3338343},
  ahighelement1={3408763.67,-13748.47903,23.65898074,-0.002423804419,4.43139566e-007,-4.35268339e-011,1.775410633e-015},
  bhighelement1={88204.2938,-137.1278108},
  R=296.3827247982046);

constant IdealGases.Common.DataRecord C2H4O_ethylene_o (
  name="C2H4O_ethylene_o",
  MM=0.04405256,
  Hf=-1194816.373895183,
  H0=245856.6539606325,
  Tlimit=1000,
  allow={-172823.3345,3816.6788,-26.29851977,0.1014103162,-0.0001240578373,
         8.03404035e-008,-2.120942544e-011},
  blow={-24375.19333,165.4885056},
  ahighelement1={3151809.957,-14236.46316,27.08080476,-0.002606238456,
                 4.853891929999999e-007,-4.85214476e-011,2.011778721e-015},
  bhighelement1={76625.61440000001,-156.3952401},
```

```

R=188.7398144398419);

constant IdealGases.Common.DataRecord CH3CHO_ethanal (
  name="CH3CHO_ethanal",
  MM=0.04405256,
  Hf=-3772538.98524853,
  H0=292757.4697134514,
  Tlimit=1000,
  allow={-137390.4369,2559.937679,-13.40470172,0.0592212861999999,-
6.24000605e-005,
         3.70332441e-008,-9.34269741e-012},
  blow={-33187.3131,100.7417652},
  ahigh={3321176.59,-14497.19957,27.08421279,-0.002879320054,5.55630992e-007,
         -5.73267488e-011,2.443965239e-015},
  bhigh={65077.5564,-153.6236027},
  R=188.7398144398419);

constant IdealGases.Common.DataRecord CH3COOH (
  name="CH3COOH",
  MM=0.06005196,
  Hf=-7197917.270310578,
  H0=226427.097466927,
  Tlimit=1000,
  allow={-32191.9198,1196.329795,-8.70582402,0.0569625759,-5.75788716e-005,
         3.35211522e-008,-8.61443822999999e-012},
  blow={-58401.1286999999,72.8241391999999},
  ahight={2103514.223,-14678.22192,33.8280283,-0.00569485868,1.343221353e-006,
         -1.606041158e-010,7.652794250000001e-015},
  bhigh={29242.28407,-193.527885},
  R=138.4546316223484);

constant IdealGases.Common.DataRecord OHCH2COOH (
  name="OHCH2COOH",
  MM=0.07605136,
  Hf=-7665872.115896415,
  H0=223625.7050498505,
  Tlimit=1000,
  allow={-313897.858,5693.06877,-36.8516029,0.1563502811,-0.0002039487993,
         1.340232412e-007,-3.50913026e-011},
  blow={-98016.4008999999,226.9492355},
  ahight={1946628.253,-9804.02020000001,28.44111243,-0.000787640475,
         6.416275950000001e-008,1.069321262e-012,-3.23717828e-016},
  bhigh={-16946.5047,-147.4109363},
  R=109.3270652885103);

constant IdealGases.Common.DataRecord C2H5 (
  name="C2H5",
  MM=0.0290611,
  Hf=4083060.861426443,
  H0=419296.7919314823,
  Tlimit=1000,
  allow={-141131.2551,2714.285088,-15.34977725,0.0645167258000001,-
7.259143960000001e-005,
         4.59911601e-008,-1.218367535e-011},
  blow={598.141884,109.096652},
  ahight={4169220.4,-16629.82142,27.95442134,-0.003051715761,
         5.685160040000001e-007,-5.6828636e-011,2.355648561e-015},
  bhigh={-16946.5047,-147.4109363});

```

---

**1010 Modelica.Media.IdealGases.Common.SingleGasesData**

---

```
bhigh={113701.0087,-163.9357995},  
R=286.1031413126138);  
  
constant IdealGases.Common.DataRecord C2H5Br (  
  name="C2H5Br",  
  MM=0.1089651,  
  Hf=-583673.1210268242,  
  H0=124526.1097360531,  
  Tlimit=1000,  
  allow={-137417.2662,2861.429418,-18.24108956,0.08566230600000001,-  
0.0001047174473,  
  6.90607457000001e-008,-1.862276022e-011},  
  blow={-21984.80205,125.8435579},  
  ahigh={2378649.403,-12670.33647,27.77558646,-0.002010898783,4.43608341e-007,  
-5.34819809e-011,2.63983585e-015},  
  bhigh={64140.6205,-152.8277085},  
  R=76.30399091085127);  
  
constant IdealGases.Common.DataRecord C2H6 (  
  name="C2H6",  
  MM=0.03006904,  
  Hf=-2788633.890539904,  
  H0=395476.3437741943,  
  Tlimit=1000,  
  allow={-186204.4161,3406.19186,-19.51705092,0.0756583559,-8.20417322e-005,  
5.0611358e-008,-1.319281992e-011},  
  blow={-27029.3289,129.8140496},  
  ahigh={5025782.13,-20330.22397,33.2255293,-0.00383670341,7.23840586e-007,-  
7.3191825e-011,  
  3.065468699e-015},  
  bhigh={111596.395,-203.9410584},  
  R=276.5127187299628);  
  
constant IdealGases.Common.DataRecord CH3N2CH3 (  
  name="CH3N2CH3",  
  MM=0.05808244,  
  Hf=2560143.134482642,  
  H0=284509.3801155737,  
  Tlimit=1000,  
  allow={-373849.232,5880.45313,-29.86398524,0.1087380861,-0.0001167950177,  
6.91689488999999e-008,-1.719950055e-011},  
  blow={-11899.84084,194.8246321},  
  ahigh={4993357.09,-21609.96161,39.6444992,-0.00419645011,  
8.02336198000001e-007,-8.21226002000001e-011,3.47723744e-015},  
  bhigh={144996.261,-237.2109745},  
  R=143.1494957856454);  
  
constant IdealGases.Common.DataRecord C2H5OH (  
  name="C2H5OH",  
  MM=0.04606844,  
  Hf=-5100020.751733725,  
  H0=315659.1801241805,  
  Tlimit=1000,  
  allow={-234279.1392,4479.18055,-27.44817302,0.1088679162,-0.0001305309334,  
8.43734639999999e-008,-2.234559017e-011},  
  blow={-50222.29,176.4829211},  
  ahigh={4694817.65,-19297.98213,34.4758404,-0.00323616598,5.78494772e-007,-
```

```

5.56460027e-011,
  2.2262264e-015},
bhigh={86016.2270999999,-203.4801732},
R=180.4808671619877);

constant IdealGases.Common.DataRecord CH3OCH3 (
  name="CH3OCH3",
  MM=0.04606844,
  Hf=-3996445.288792067,
  H0=311588.1284454173,
  Tlimit=1000,
  allow={-269310.3242,4300.70970999999,-21.52788028,0.08131833390000001,-
8.29567132e-005,
  4.80191151e-008,-1.188699808e-011},
  blow={-44102.3709,146.7666934},
  ahighe={4933577.19,-20830.94065,36.2905061,-0.004108351640000001,
  7.90322031e-007,-8.13143563e-011,3.45816611e-015},
  bhigh={101330.1012,-218.5447466},
  R=180.4808671619877);

constant IdealGases.Common.DataRecord CH3O2CH3 (
  name="CH3O2CH3",
  MM=0.06206784,
  Hf=-2021981.109701901,
  H0=276365.9247687692,
  Tlimit=1000,
  allow={-228578.4757,3820.14257,-19.76647823,0.0884074386,-9.641284560000001e-
005,
  5.90720083e-008,-1.526491225e-011},
  blow={-34920.1696,138.6769151},
  ahighe={5316368.47,-22212.67874,40.3433509,-0.00461274809,
  8.792987200000001e-007,-9.068221189999999e-011,3.865664890000001e-015},
  bhigh={116159.6028,-239.5296055},
  R=133.9578113238676);

constant IdealGases.Common.DataRecord CCN (
  name="CCN",
  MM=0.0380281,
  Hf=21157945.62441983,
  H0=290272.8245692001,
  Tlimit=1000,
  allow={-16962.81385,98.3789163,3.81266294,0.00534689423,-2.473598508e-006,-
3.73056422e-010,
  4.48175686e-013},
  blow={94800.72570000001,5.553165572},
  ahighe={79486.74890000001,-1344.786906,8.309986459999999,-0.0002220105361,
  1.753683113e-008,2.545998719e-012,-2.645649117e-016},
  bhigh={102318.7495,-22.5979394},
  R=218.6402160507625);

constant IdealGases.Common.DataRecord CNC (
  name="CNC",
  MM=0.0380281,
  Hf=18010748.86728498,
  H0=298637.3497492644,
  Tlimit=1000,
  allow={-70751.9271,1007.523898,-1.576789967,0.02052532634,-2.278935009e-005,

```

## 1012 Modelica.Media.IdealGases.Common.SingleGasesData

---

```
    1.283362343e-008,-2.933174091e-012},  
blow={76133.2485,34.87094132},  
ahigh={-90313.13559999999,-831.32365,8.11473595,-0.0002447691991,  
      5.39750877e-008,-6.18398486e-012,2.865172411e-016},  
bhigh={84518.33600000001,-21.02937255},  
R=218.6402160507625);  
  
constant IdealGases.Common.DataRecord OCCN (  
  name="OCCN",  
  MM=0.05402754,  
  Hf=3886906.566539954,  
  H0=251618.9150940428,  
  Tlimit=1000,  
  alow={24288.01276,-552.586462,8.58783817,-0.00337938777,1.119841826e-005,-  
1.008408077e-008,  
      3.086448824e-012},  
blow={25996.20072,-16.59592359},  
ahigh={935913.1680000001,-4441.08228999999,13.68959297,-0.001647530917,  
      3.81987344e-007,-3.945161e-011,1.509598839e-015},  
bhigh={49512.2278,-54.1708968},  
R=153.8932181624409);  
  
constant IdealGases.Common.DataRecord C2N2 (  
  name="C2N2",  
  MM=0.0520348,  
  Hf=5940255.367561709,  
  H0=244358.6407558019,  
  Tlimit=1000,  
  alow={108240.4484,-1928.137871,15.53891898,-0.01821159329,2.77884084e-005,-  
1.899434373e-008,  
      4.94967772e-012},  
blow={44490.9759,-60.90964741},  
ahigh={793442.372,-3997.37627,13.1449743,-0.0008747782000000001,  
      2.059156733e-007,-2.200469389e-011,9.97448577e-016},  
bhigh={58636.323,-54.73201251},  
R=159.7867580926611);  
  
constant IdealGases.Common.DataRecord C2O (  
  name="C2O",  
  MM=0.0400208,  
  Hf=7272185.113740855,  
  H0=262006.7065126135,  
  Tlimit=1000,  
  alow={-3959.92942,-111.7516348,4.59396006,0.00371060202,-7.01476018e-007,-  
1.371129839e-009,  
      7.12385399999999e-013},  
blow={34101.0974,0.4622805600000001},  
ahigh={-634805.659,1184.133091,4.87917334,0.001757538773,-3.95755227e-007,  
      3.98917994e-011,-1.546043135e-015},  
bhigh={24898.03938,0.980904059},  
R=207.7537680406189);  
  
constant IdealGases.Common.DataRecord C3 (  
  name="C3",  
  MM=0.0360321,  
  Hf=23311121.08370037,  
  H0=336065.5082551392,
```

```

Tlimit=1000,
alow={-43546.1448,666.018322,1.451033157,0.00743451312,-3.81015299e-006,-
2.336961396e-011,
        4.40705453e-013},
blow={96351.7019999999,20.25173297},
ahigh={4508098.93,-14610.33761,22.81974644,-0.008544340610000001,
        2.146069341e-006,-2.103867761e-010,6.351589060000001e-015},
bhigh={191197.6065,-127.1869723},
R=230.7518018655588);

constant IdealGases.Common.DataRecord C3H3_1_propynl(
  name="C3H3_1_propynl",
  MM=0.03905592,
  Hf=11521940.84789195,
  H0=317493.4811419114,
  Tlimit=1000,
  alow={-65058.5935,1350.858921,-5.82543393,0.0375661048,-3.73490334e-005,
        2.117676603e-008,-5.139113250000001e-012},
  blow={46565.1053,57.8147755},
  ahigh={4550654.87,-16405.74172,27.12605991,-0.00447460038,1.037712415e-006,
        -1.250211369e-010,6.02658205e-015},
  bhigh={153408.7662,-156.5931809},
  R=212.8863434787863);

constant IdealGases.Common.DataRecord C3H3_2_propynl(
  name="C3H3_2_propynl",
  MM=0.03905592,
  Hf=8495511.051845662,
  H0=338745.0609280232,
  Tlimit=1000,
  alow={61885.78320000001,-890.957867,6.34755882,0.01633173115,-1.949975695e-
005,
        1.417349778e-008,-4.19986632e-012},
  blow={42717.8583,-12.31400729},
  ahigh={2989723.833,-11189.54446,22.22225052,-0.002068106902,4.12188364e-007,
        -4.43898059e-011,1.970824701e-015},
  bhigh={106187.8289,-118.6744583},
  R=212.8863434787863);

constant IdealGases.Common.DataRecord C3H4_allene(
  name="C3H4_allene",
  MM=0.04006386,
  Hf=4765392.051589637,
  H0=314611.9470265721,
  Tlimit=1000,
  alow={-16451.55745,962.945781,-7.53232668,0.05518219110000001,-6.73358512e-
005,
        4.53270905e-008,-1.251837614e-011},
  blow={17724.94269,61.5196976},
  ahigh={3479355.1,-14304.12453,27.02534756,-0.002557412369,4.70664675e-007,-
4.65168807e-011,
        1.908219044e-015},
  bhigh={107231.2354,-154.8846158},
  R=207.5304775925235);

constant IdealGases.Common.DataRecord C3H4_propyne(
  name="C3H4_propyne",

```

---

**1014 Modelica.Media.IdealGases.Common.SingleGasesData**

---

```
MM=0.04006386,
Hf=4615131.941854829,
H0=325243.6485151456,
Tlimit=1000,
alow={-35638.844,832.81391,-4.07375944,0.04113929609999999,-4.47044495e-005,
      2.847458197e-008,-7.69529824e-012},
blow={17102.06236,45.1672095},
ahigh={3710441.42,-14891.45507,27.32397127,-0.00264526477,
       4.858300350000001e-007,-4.79412848e-011,1.964338121e-015},
bhigh={110489.8462,-156.7992462},
R=207.5304775925235);

constant IdealGases.Common.DataRecord C3H4_cyclo(
  name="C3H4_cyclo",
  MM=0.04006386,
  Hf=6916457.875002559,
  H0=283888.3472536096,
  Tlimit=1000,
  alow={-19695.20627,1505.379338,-14.18206573,0.07642632960000001,-
9.765583660000001e-005,
      6.61200382e-008,-1.811251145e-011},
  blow={26256.31782,96.0463189},
  ahigh={3168399.58,-13710.44699,26.64303646,-0.00242040805,4.42799413e-007,-
4.3518202e-011,
      1.775948955e-015},
  bhigh={113368.3702,-152.2619086},
  R=207.5304775925235);

constant IdealGases.Common.DataRecord C3H5_allyl(
  name="C3H5_allyl",
  MM=0.04107180000000001,
  Hf=3983131.97863254,
  H0=310157.4072721429,
  Tlimit=1000,
  alow={-43159.9614,1441.600907,-11.97014426,0.0731979646,-9.066357849999999e-
005,
      6.077059450000001e-008,-1.658826363e-011},
  blow={12321.5746,85.63173239999999},
  ahigh={4094570.59,-16766.76186,31.23006342,-0.002885449982,5.21134354e-007,-
5.05828422e-011,2.039932554e-015},
  bhigh={118572.0481,-182.3070197},
  R=202.437487521852);

constant IdealGases.Common.DataRecord C3H6_propylene(
  name="C3H6_propylene",
  MM=0.04207974,
  Hf=475288.1077687267,
  H0=322020.9535515191,
  Tlimit=1000,
  alow={-191246.2174,3542.07424,-21.14878626,0.0890148479,-0.0001001429154,
      6.267959389999999e-008,-1.637870781e-011},
  blow={-15299.61824,140.7641382},
  ahigh={5017620.34,-20860.84035,36.4415634,-0.00388119117,7.27867719e-007,-
7.321204500000001e-011,
      3.052176369e-015},
  bhigh={126124.5355,-219.5715757},
  R=197.588483198803);
```

```

constant IdealGases.Common.DataRecord C3H6_cyclo(
  name="C3H6_cyclo",
  MM=0.04207974,
  Hf=1266642.807203657,
  H0=271151.4139583562,
  Tlimit=1000,
  allow={-156578.777, 4111.12987, -32.3344746, 0.1306337881, -0.0001645563833,
         1.095708326e-007, -2.956394783e-011},
  blow={-12452.71686, 193.1559109},
  ahigh={4785000.67, -20421.18175, 36.3149578, -0.00356131944, 6.47624124e-007, -
6.328430100000001e-011,
         2.568705857e-015},
  bhigh={126827.4126, -222.3729099},
  R=197.588483198803);

constant IdealGases.Common.DataRecord C3H6O_propylox(
  name="C3H6O_propylox",
  MM=0.05807914,
  Hf=-1613660.25736607,
  H0=248031.4446804825,
  Tlimit=1000,
  allow={-229280.8804, 4495.75054, -29.41117945, 0.1213113827, -0.0001440060464,
         9.202051750000001e-008, -2.416278343e-011},
  blow={-33176.9721, 184.6878218},
  ahigh={4789729.989999999, -21068.95971, 39.5464773, -0.00391092998,
         7.32553151e-007, -7.35708597e-011, 3.0606185e-015},
  bhigh={112034.454, -237.2004192},
  R=143.1576294001599);

constant IdealGases.Common.DataRecord C3H6O_acetone(
  name="C3H6O_acetone",
  MM=0.05807914,
  Hf=-3738857.014756073,
  H0=278812.5478442002,
  Tlimit=1000,
  allow={-227780.2525, 4215.28001, -24.15785316, 0.0990748332, -0.0001084940903,
         6.583355959999999e-008, -1.676046146e-011},
  blow={-47262.4394, 160.7926432},
  ahigh={5001601.92, -21701.55542, 39.6449399, -0.00417994505, 7.96242953e-007, -
8.122558050000001e-011,
         3.42878096e-015},
  bhigh={101514.5028, -236.8533477},
  R=143.1576294001599);

constant IdealGases.Common.DataRecord C3H6O_propanal(
  name="C3H6O_propanal",
  MM=0.05807914,
  Hf=-3202526.759177219,
  H0=298339.0594282216,
  Tlimit=1000,
  allow={-265578.1702, 4250.64045, -21.09249262, 0.09194256120000001, -
0.0001004653044,
         6.13331482e-008, -1.576298535e-011},
  blow={-44503.7105, 145.6678605},
  ahigh={4830933.489999999, -20754.51152, 39.2485518, -0.004095514,
         7.88169216e-007, -8.107684080000001e-011, 3.43710543e-015},
  bhigh={99449.37879999999, -231.6690627},
  R=143.1576294001599);

```

---

**1016 Modelica.Media.IdealGases.Common.SingleGasesData**

---

```
constant IdealGases.Common.DataRecord C3H7_n_propyl(
  name="C3H7_n_propyl",
  MM=0.04308768,
  Hf=2332453.267384088,
  H0=344879.5108021598,
  Tlimit=1000,
  allow={-189533.7073, 3949.51726, -26.06216089, 0.1121920441, -0.0001365292213,
         9.02366272e-008, -2.44105699e-011},
  blow={-7227.87744, 167.3705556},
  ahighe={5646512.94, -22910.87136, 39.8727518, -0.004106232870000001,
          7.56255777e-007, -7.47826302e-011, 3.068983677e-015},
  bhigh={148300.6853, -240.378119},
  R=192.9663421191394);

constant IdealGases.Common.DataRecord C3H7_i_propyl (
  name="C3H7_i_propyl",
  MM=0.04308768,
  Hf=2165352.137780452,
  H0=343652.9188853984,
  Tlimit=1000,
  allow={-295206.3445, 5294.4323, -31.05287013, 0.1143871563, -0.0001291752393,
         8.05784376e-008, -2.093908432e-011},
  blow={-14768.15514, 198.808236},
  ahighe={5807002.520000001, -24112.19997, 40.852884, -0.00451785133,
          8.499427170000001e-007, -8.573514339999999e-011, 3.58338396e-015},
  bhigh={154650.405, -248.7098372},
  R=192.9663421191394);

constant IdealGases.Common.DataRecord C3H8 (
  name="C3H8",
  MM=0.04409562,
  Hf=-2373931.923397381,
  H0=334301.1845620949,
  Tlimit=1000,
  allow={-243314.4337, 4656.27081, -29.39466091, 0.1188952745, -0.0001376308269,
         8.814823909999999e-008, -2.342987994e-011},
  blow={-35403.3527, 184.1749277},
  ahighe={6420731.680000001, -26597.91134, 45.3435684, -0.00502066392,
          9.471216939999999e-007, -9.57540523e-011, 4.00967288e-015},
  bhigh={145558.2459, -281.8374734},
  R=188.5555073270316);

constant IdealGases.Common.DataRecord C3H8O_1propanol (
  name="C3H8O_1propanol",
  MM=0.06009502,
  Hf=-4246608.121604752,
  H0=291517.6498818038,
  Tlimit=1000,
  allow={-261697.3337, 5192.37666, -32.9648116, 0.1354568128, -0.0001593156164,
         1.01949816e-007, -2.688552974e-011},
  blow={-56128.5435, 208.5024431},
  ahighe={6308672.12, -26422.10376, 47.1511259, -0.004642511930000001,
          8.59346536e-007, -8.68209182e-011, 3.64222401e-015},
  bhigh={125500.3155, -285.9463804},
  R=138.3554244594644);

constant IdealGases.Common.DataRecord C3H8O_2propanol (
```

```

name="C3H8O_2propanol",
MM=0.06009502,
Hf=-4537813.615837053,
H0=287291.2763819697,
Tlimit=1000,
alow={-338651.024, 6106.048000000001, -37.9141804, 0.1530494531, -
0.0001864354461,
1.213257738e-007, -3.22043349e-011},
blow={-62799.5935, 233.432261},
ahigh={6001074.75, -25058.7683, 46.22096120000001, -0.00427246631,
7.69367877e-007, -7.45058484e-011, 2.998959935e-015},
bhigh={114851.8732, -279.6132222},
R=138.3554244594644);

constant IdealGases.Common.DataRecord CNCOCN (
name="CNCOCN",
MM=0.08004498,
Hf=3092011.516524834,
H0=214226.5261356802,
Tlimit=1000,
alow={113105.2075, -1978.834961, 16.26886206, -0.008330803440000001,
1.884127045e-005, -1.530681289e-008, 4.4181253e-012},
blow={36802.6174, -59.6332728},
ahigh={700052.2440000001, -5086.002009999999, 19.4675349, -0.001302852727,
2.75360075e-007, -3.056290852e-011, 1.382140838e-015},
bhigh={55393.3888, -86.2755086},
R=103.8724976881748);

constant IdealGases.Common.DataRecord C3O2 (
name="C3O2",
MM=0.06803090000000001,
Hf=-1376403.957613379,
H0=221737.2252902725,
Tlimit=1000,
alow={157987.3382, -2529.493506, 18.01761578, -0.01786032042, 2.978671986e-005,
-2.182900022e-008, 6.013797219999999e-012},
blow={-1121.054014, -72.77721170000001},
ahigh={696869.9889999999, -4624.73319, 16.63905725, -0.001175486554,
2.478106444e-007, -2.745165984e-011, 1.239566766e-015},
bhigh={12525.80069, -72.75968780000001},
R=122.2161106203211);

constant IdealGases.Common.DataRecord C4 (
name="C4",
MM=0.0480428,
Hf=21520472.20395147,
H0=273042.8284779405,
Tlimit=1000,
alow={39037.805, -894.828077999999, 10.50952925, -0.00655289446,
1.243940464e-005, -8.645341370000001e-009, 2.263638846e-012},
blow={126642.5869, -30.77594475},
ahigh={920068.513, -1530.3118, 6.0500692, 0.00525274367, -1.779154772e-006,
2.589873632e-010, -1.385553481e-014},
bhigh={133438.9611, -7.26114882},
R=173.0638513991691);

constant IdealGases.Common.DataRecord C4H2_butadiyne(

```

---

**1018 Modelica.Media.IdealGases.Common.SingleGasesData**

---

```
name="C4H2_butadiyne",
MM=0.05005868,
Hf=8989449.98150171,
H0=287539.743357196,
Tlimit=1000,
alow={246754.2569,-3897.85564,23.66080456,-0.02208077805,2.78110114e-005,-
1.57734001e-008,
      3.42316546e-012},
blow={70869.0782,-110.917356},
ahigh={2328179.913,-8925.186090000001,21.14326883,-0.001368871276,
      2.327503159e-007,-2.124517624e-011,8.053313019999999e-016},
bhigh={105778.8416,-108.8313574},
R=166.0945114813255);

constant IdealGases.Common.DataRecord C4H4_1_3minuscyclo(
name="C4H4_1_3minuscyclo",
MM=0.05207456,
Hf=7393245.377397331,
H0=232427.0046640816,
Tlimit=1000,
alow={-27784.28049,1768.176915,-17.57895171,0.0938351291999999,-
0.0001195524281,
      7.97808619e-008,-2.1527511e-011},
blow={38116.2712,112.8478124},
ahigh={2991498.949,-14167.81502,30.01978876,-0.002579200423,4.78999045e-007,
      -4.77532971e-011,1.974923662e-015},
bhigh={127466.692,-172.7532057},
R=159.6647576090898);

constant IdealGases.Common.DataRecord C4H6_butadiene(
name="C4H6_butadiene",
MM=0.05409044,
Hf=2033631.081573749,
H0=279716.7114928257,
Tlimit=1000,
alow={-91811.30530000001,3312.57053,-29.85828611,0.1479201147,-
0.0002056618326,
      1.466496826e-007,-4.14528573e-011},
blow={-2077.309444,178.0687329},
ahigh={-23619031.88,56513.2337,-32.7573832,0.02293070572,-2.297106441e-006,
      4.29259621e-011,4.23676604e-015},
bhigh={-367135.862,301.3437302},
R=153.7142607824969);

constant IdealGases.Common.DataRecord C4H6_1butyne(
name="C4H6_1butyne",
MM=0.05409044,
Hf=3054144.133418031,
H0=296170.635698286,
Tlimit=1000,
alow={-55970.3997,1433.191711,-9.69121072000001,0.0715000239,-8.1579678e-
005,
      5.29036497e-008,-1.435655372e-011},
blow={11849.87013,76.6022536},
ahigh={6364402.7,-23920.87731,40.7375041,-0.00317672649,1.199856984e-007,
      3.20180251e-011,-2.854392633e-015},
bhigh={163433.5546,-246.0284791},
R=153.7142607824969);
```

```

constant IdealGases.Common.DataRecord C4H6_2butyne (
  name="C4H6_2butyne",
  MM=0.05409044,
  Hf=2693636.805320866,
  H0=307632.9199762472,
  Tlimit=1000,
  allow={-265075.6405, 4490.72869, -24.00723889, 0.0981957955, -0.0001079717182,
         6.675555770000001e-008, -1.740766886e-011},
  blow={-5328.36371, 159.5035268},
  ahighe={3981304.33, -18730.32899, 36.5456149, -0.002369686378, 3.99021181e-007, -
3.69870727e-011,
         1.442072006e-015},
  bhigh={126146.0214, -215.5866205},
  R=153.7142607824969);

constant IdealGases.Common.DataRecord C4H6_cyclo (
  name="C4H6_cyclo",
  MM=0.05409044,
  Hf=2896999.913478241,
  H0=232158.6587204689,
  Tlimit=1000,
  allow={-204670.7734, 4919.47057, -37.5327816, 0.1497293031, -0.0001860139375,
         1.219909019e-007, -3.25407359e-011},
  blow={-3915.95054, 223.3282138},
  ahighe={4517367.819999999, -20973.11985, 40.0803917, -0.00394979398,
         7.44848499e-007, -7.52975417e-011, 3.153253502e-015},
  bhigh={140646.9848, -243.4836999},
  R=153.7142607824969);

constant IdealGases.Common.DataRecord C4H8_1_butene (
  name="C4H8_1_butene",
  MM=0.0561063199999999,
  Hf=-9624.584182316718,
  H0=305134.9651875226,
  Tlimit=1000,
  allow={-272149.2014, 5100.079250000001, -31.8378625, 0.1317754442, -
0.0001527359339,
         9.714761109999999e-008, -2.56020447e-011},
  blow={-25230.96386, 200.6932108},
  ahighe={6257948.609999999, -26603.76305, 47.6492005, -0.00438326711,
         7.12883844e-007, -5.991020839999999e-011, 2.051753504e-015},
  bhigh={156925.2657, -291.3869761},
  R=148.1913623991023);

constant IdealGases.Common.DataRecord C4H8_cis2_buten (
  name="C4H8_cis2_buten",
  MM=0.0561063199999999,
  Hf=-131892.4499058217,
  H0=299431.5078942979,
  Tlimit=1000,
  allow={-277387.0877, 5382.38404, -33.751881, 0.1322980623, -0.0001490975922,
         9.277722e-008, -2.408282948e-011},
  blow={-27158.84347, 211.4462085},
  ahighe={6461018.35, -27753.76432, 48.6353236, -0.00486238635,
         8.412626100000001e-007, -7.63389037e-011, 2.861702826e-015},
  bhigh={163085.6187, -300.3105998},
  R=148.1913623991023);

```

---

**1020 Modelica.Media.IdealGases.Common.SingleGasesData**

---

```
constant IdealGases.Common.DataRecord C4H8_isobutene(
  name="C4H8_isobutene",
  MM=0.0561063199999999,
  Hf=-304778.4991066961,
  H0=303174.4017429766,
  Tlimit=1000,
  allow={-232720.5032,3941.99424,-22.24581184,0.1012790864,-0.0001073194065,
         6.45469691e-008,-1.646330345e-011},
  blow={-22337.66063,147.9597621},
  ahighelement{6484970.99,-27325.04764,48.3632108,-0.00476800405,8.23387584e-007,-
7.449252999999999e-011,
              2.782303056e-015},
  bhigh={159594.1773,-298.2986237},
  R=148.1913623991023);

constant IdealGases.Common.DataRecord C4H8_cyclo(
  name="C4H8_cyclo",
  MM=0.0561063199999999,
  Hf=506181.8347736941,
  H0=241223.5020938818,
  Tlimit=1000,
  allow={-304765.5983,6519.48273,-46.6298759,0.1743593052,-0.0002090964176,
         1.343528679e-007,-3.5427274e-011},
  blow={-27000.31171,273.8348195},
  ahighelement{4456213.810000001,-23010.18492,44.9244846,-0.003080145176,
               5.317165260000001e-007,-5.08107938e-011,2.048919092e-015},
  bhigh={135548.7603,-277.1801965},
  R=148.1913623991023);

constant IdealGases.Common.DataRecord C4H9_n_butyl(
  name="C4H9_n_butyl",
  MM=0.05711426,
  Hf=1164857.95316266,
  H0=346620.9664626662,
  Tlimit=1000,
  allow={-223956.0407,4676.554099999999,-29.85424449,0.1345493704,-
0.000160066035,
         1.045338096e-007,-2.812951048e-011},
  blow={-15252.97704,190.1651559},
  ahighelement{7198686.94,-29592.41524,52.4204281,-0.00544157244,
               9.917758369999999e-007,-9.464455870000001e-011,3.72948704e-015},
  bhigh={183456.6472,-321.420917},
  R=145.5761135660341);

constant IdealGases.Common.DataRecord C4H9_i_butyl(
  name="C4H9_i_butyl",
  MM=0.05711426,
  Hf=1003602.252747387,
  H0=320707.9983177581,
  Tlimit=1000,
  allow={-239946.1281,4697.44454,-30.8747256,0.1391655831,-0.0001670233968,
         1.092423088e-007,-2.936209962e-011},
  blow={-16381.51036,193.6662372},
  ahighelement{6752936.06,-28374.23721,51.4068761,-0.00498239901,8.80150663e-007,-
8.12573869e-011,
               3.099140009e-015},
  bhigh={174344.848,-314.9940745},
  R=145.5761135660341);
```

```

constant IdealGases.Common.DataRecord C4H9_s_butyl(
  name="C4H9_s_butyl",
  MM=0.05711426,
  Hf=1243122.120465187,
  H0=307062.5269416079,
  Tlimit=1000,
  allow={-335106.289, 6312.945949999999, -40.0302526, 0.1563875047, -
0.0001842331788,
  1.182727848e-007, -3.131156589e-011},
  blow={-22160.45659, 248.1295714},
  ahigh={7224744.35, -30352.91099, 52.8855183, -0.00565208355, 1.060032692e-006, -
1.066129835e-010,
  4.44381096000001e-015},
  bhigh={188349.4279, -325.571988},
  R=145.5761135660341);

constant IdealGases.Common.DataRecord C4H9_t_butyl(
  name="C4H9_t_butyl",
  MM=0.05711426,
  Hf=905203.0088457768,
  H0=297818.3556961081,
  Tlimit=1000,
  allow={-472346.195, 8090.198770000001, -46.8367534, 0.1575300095, -
0.0001686559436,
  9.98104013e-008, -2.487608823e-011},
  blow={-33193.6741, 289.4838706},
  ahigh={7151064.95, -31712.2441, 54.4251032, -0.006392331160000001,
  1.241097612e-006, -1.2872483e-010, 5.51261874e-015},
  bhigh={193450.6997, -339.9325290000001},
  R=145.5761135660341);

constant IdealGases.Common.DataRecord C4H10_n_butane(
  name="C4H10_n_butane",
  MM=0.0581222,
  Hf=-2164233.28779709,
  H0=330832.0228759407,
  Tlimit=1000,
  allow={-317587.254, 6176.331819999999, -38.9156212, 0.1584654284, -
0.0001860050159,
  1.199676349e-007, -3.20167055e-011},
  blow={-45403.63390000001, 237.9488665},
  ahigh={7682322.45, -32560.5151, 57.3673275, -0.00619791681, 1.180186048e-006, -
1.221893698e-010,
  5.250635250000001e-015},
  bhigh={177452.656, -358.791876},
  R=143.0515706563069);

constant IdealGases.Common.DataRecord C4H10_isobutane(
  name="C4H10_isobutane",
  MM=0.0581222,
  Hf=-2322520.482707124,
  H0=308599.8121199817,
  Tlimit=1000,
  allow={-383446.933, 7000.03964, -44.400269, 0.1746183447, -0.0002078195348,
  1.339792433e-007, -3.55168163e-011},
  blow={-50340.18889999999, 265.8966497},
  ahigh={7528018.92, -32025.1706, 57.00161, -0.00606001309, 1.143975809e-006, -
1.157061835e-010,
  
```

---

**1022 Modelica.Media.IdealGases.Common.SingleGasesData**

---

```
        4.84604291e-015},
bhigh={172850.0802,-357.617689},
R=143.0515706563069);

constant IdealGases.Common.DataRecord C4N2 (
  name="C4N2",
  MM=0.0760562,
  Hf=6958012.627504397,
  H0=234029.5860166561,
  Tlimit=1000,
  allow={158780.2866,-2987.184206,23.48081602,-0.02607502448,4.04283003e-005,-
2.804912444e-008,
         7.39765205e-012},
  blow={75052.9947,-101.757825},
  ahigh={1167686.152,-6198.644179999999,20.62070093,-0.001518619449,
         3.16236168e-007,-3.4699228e-011,1.555154128e-015},
  bhigh={96674.09389999999,-96.69734738000001},
  R=109.3201080253812);

constant IdealGases.Common.DataRecord C5 (
  name="C5",
  MM=0.0600535,
  Hf=17499801.5436236,
  H0=269623.4357697719,
  Tlimit=1000,
  allow={-12008.01119,-555.3702910000001,11.23828271,-0.00434788452,
         1.73898749e-005,-1.707945418e-008,5.57454191e-012},
  blow={126240.4627,-32.6230899},
  ahigh={217205.5356,-2958.510027,15.61080967,-0.0008200361920000001,
         1.776898025e-007,-2.009853583e-011,9.222677770000001e-016},
  bhigh={139520.8064,-64.33077179999999},
  R=138.4510811193353);

constant IdealGases.Common.DataRecord C5H6_1_3cyclo (
  name="C5H6_1_3cyclo",
  MM=0.06610114,
  Hf=2031735.004872836,
  H0=204769.9782484841,
  Tlimit=1000,
  allow={-188625.9728,4738.22087,-38.8895801,0.1667438533,-0.0002141149581,
         1.438132328e-007,-3.90647811e-011},
  blow={-5667.022010000001,227.9898557},
  ahigh={4428478.19,-21235.47976,43.0981901,-0.003914783460000001,
         7.31184701e-007,-7.32724119e-011,3.044403403e-015},
  bhigh={138254.2782,-260.5959678},
  R=125.78409389006);

constant IdealGases.Common.DataRecord C5H8_cyclo (
  name="C5H8_cyclo",
  MM=0.06811702,
  Hf=497672.9751242788,
  H0=218112.5803800577,
  Tlimit=1000,
  allow={-263111.4588,5987.75349,-45.222446,0.1804626339,-0.0002177216062,
         1.402022671e-007,-3.6990943e-011},
  blow={-23795.04749,266.0136401},
  ahigh={4569848.1,-24060.18294,48.8100646,-0.00341305498,6.04700166e-007,-
```

```

5.927495780000001e-011,
    2.44786539e-015},
bhigh={141398.3093,-298.9533527},
R=122.0615934167408);

constant IdealGases.Common.DataRecord C5H10_1_pentene(
  name="C5H10_1_pentene",
  MM=0.07013290000000001,
  Hf=-303423.9279995551,
  H0=309127.3852927798,
  Tlimit=1000,
  allow={-534054.813,9298.917380000001,-56.6779245,0.2123100266,-
0.000257129829,
        1.666834304e-007,-4.43408047e-011},
  blow={-47906.8218,339.60364},
  ahigh={3744014.97,-21044.85321,47.3612699,-0.00042442012,-3.89897505e-008,
        1.367074243e-011,-9.31319423e-016},
  bhigh={115409.1373,-278.6177449000001},
  R=118.5530899192818);

constant IdealGases.Common.DataRecord C5H10_cyclo(
  name="C5H10_cyclo",
  MM=0.07013290000000001,
  Hf=-1099341.393269065,
  H0=214212.4024530569,
  Tlimit=1000,
  allow={-414111.971,8627.5928,-62.0295998,0.2259910921,-0.000268230333,
        1.706289935e-007,-4.46405092e-011},
  blow={-49315.2214,358.391623},
  ahigh={7501938.73,-35058.6485,63.2248075,-0.00694035658,1.337306593e-006,-
1.377905033e-010,
        5.86735764e-015},
  bhigh={195492.5511,-402.65509},
  R=118.5530899192818);

constant IdealGases.Common.DataRecord C5H11_pentyl(
  name="C5H11_pentyl",
  MM=0.07114084,
  Hf=643933.9203754131,
  H0=343290.8579656918,
  Tlimit=1000,
  allow={-465371.592,8564.22042,-52.9524289,0.2094288859,-0.0002561602906,
        1.692755961e-007,-4.596788110000001e-011},
  blow={-36417.0427,319.686224},
  ahigh={5697252.899999999,-28917.06175,58.1102968,-0.00359399501,
        4.41996763e-007,-1.509664551e-011,-6.62696443e-016},
  bhigh={171700.955,-352.247712},
  R=116.873402113329);

constant IdealGases.Common.DataRecord C5H11_t_pentyl(
  name="C5H11_t_pentyl",
  MM=0.07114084,
  Hf=458245.9245631623,
  H0=276124.417423241,
  Tlimit=1000,
  allow={-515218.198,9144.32783,-56.0239789,0.1998204194,-0.0002239112591,
        1.375455547e-007,-3.52383305e-011},

```

---

**1024 Modelica.Media.IdealGases.Common.SingleGasesData**

---

```
blow={-40362.6661,340.2811},
ahigh={8602108.26,-38052.05770000001,66.4969037,-0.007533860630000001,
1.451612787e-006,-1.495593207e-010,6.368022329999999e-015},
bhigh={228185.8805,-416.940855},
R=116.873402113329);

constant IdealGases.Common.DataRecord C5H12_n_pentane(
  name="C5H12_n_pentane",
  MM=0.07214878,
  Hf=-2034130.029641527,
  H0=335196.2430965569,
  Tlimit=1000,
  allow={-276889.4625,5834.28347,-36.1754148,0.1533339707,-0.0001528395882,
  8.191092e-008,-1.792327902e-011},
  blow={-46653.7525,226.5544053},
  ahigh={-2530779.286,-8972.59326,45.3622326,-0.002626989916,3.135136419e-006,
  -5.31872894e-010,2.886896868e-014},
  bhigh={14846.16529,-251.6550384},
  R=115.2406457877736);

constant IdealGases.Common.DataRecord C5H12_i_pentane(
  name="C5H12_i_pentane",
  MM=0.07214878,
  Hf=-2130320.152329672,
  H0=305036.3429568733,
  Tlimit=1000,
  allow={-423190.339,6497.1891,-36.8112697,0.1532424729,-0.0001548790714,
  8.74989712e-008,-2.07054771e-011},
  blow={-51554.1659,230.9518218},
  ahigh={11568885.94,-45562.4687,74.9544363,-0.007845415580000001,
  1.444393314e-006,-1.464370213e-010,6.230285000000001e-015},
  bhigh={254492.7135,-480.198578},
  R=115.2406457877736);

constant IdealGases.Common.DataRecord CH3C_CH3_2CH3(
  name="CH3C_CH3_2CH3",
  MM=0.07214878,
  Hf=-2327412.882102788,
  H0=321266.6936294696,
  Tlimit=1000,
  allow={-8973222.270000001,128922.5617,-719.934483,2.056862183,-
  0.002953159699,
  2.158893146e-006,-6.26831877e-010},
  blow={-639493.201999999,4020.80636},
  ahigh={16847055.2,-59794.3057,86.8545147999999,-0.0096219176,
  1.653363091e-006,-1.674727926e-010,7.37211936e-015},
  bhigh={345849.682,-576.046697},
  R=115.2406457877736);

constant IdealGases.Common.DataRecord C6D5_phenyl(
  name="C6D5_phenyl",
  MM=0.08213471,
  Hf=3844172.579412528,
  H0=193816.6823746014,
  Tlimit=1000,
  allow={201283.7008,-1979.349332,2.6282675,0.0595186526,-6.08128045e-005,
  3.38139165e-008,-8.125429080000001e-012},
```

```

blow={46972.47489999999,0.335470098},
ahigh={1411628.125,-13625.69472,40.2898494,-0.00349098927,7.37961213e-007,-
8.19224405e-011,
3.70533097e-015},
bhigh={108807.3731,-229.3621057},
R=101.2296993560944);

constant IdealGases.Common.DataRecord C6D6(
  name="C6D6",
  MM=0.084148812,
  Hf=691125.3601536288,
  H0=193997.8190066427,
  Tlimit=1000,
  alow={276291.1236,-2865.868902,5.39507043,0.05939315170000001,-6.02363118e-
005,
3.38139857e-008,-8.306046370000001e-012},
  blow={20470.83778,-20.11790696},
  ahigh={1758057.871,-15721.21558,44.6816249,-0.00400336191999999,
8.4454344e-007,-9.36055698e-011,4.22847553e-015},
  bhigh={89620.45359999999,-261.5332268},
  R=98.80676627971884);

constant IdealGases.Common.DataRecord C6H2(
  name="C6H2",
  MM=0.0740800799999999,
  Hf=9044266.690856706,
  H0=264682.6785284249,
  Tlimit=1000,
  alow={290372.2964,-4929.7515,31.8932321,-0.03119447315,4.32576368e-005,-
2.732517022e-008,
6.67444611e-012},
  blow={101189.8682,-153.0593012},
  ahigh={2592848.577,-10833.61978,28.47459495,-0.001876727704,3.41213428e-007,
-3.33739289e-011,1.356853889e-015},
  bhigh={141851.7294,-149.4853494},
  R=112.2362718830757);

constant IdealGases.Common.DataRecord C6H5_phenyl(
  name="C6H5_phenyl",
  MM=0.0771038999999999,
  Hf=4373319.637528064,
  H0=183578.6905720723,
  Tlimit=1000,
  alow={-121127.8245,3529.04558,-31.16903422,0.146755063,-0.0001831398296,
1.192576957e-007,-3.14926586e-011},
  blow={24209.72928,186.8799946},
  ahigh={3670279.23,-18946.01209,41.8058182,-0.00350391415,6.56182174e-007,-
6.59471444999999e-011,
2.748094212e-015},
  bhigh={147674.4628,-247.5301142},
  R=107.8346490903833);

constant IdealGases.Common.DataRecord C6H5O_phenoxy(
  name="C6H5O_phenoxy",
  MM=0.0931033,
  Hf=512334.1492729044,
  H0=174087.2128055611,

```

---

**1026 Modelica.Media.IdealGases.Common.SingleGasesData**

---

```
Tlimit=1000,
alow={-129226.4217,3406.74068,-29.11367361,0.1459180378,-0.0001780202758,
      1.138615885e-007,-2.967142152e-011},
blow={-10550.26758,177.0682011},
ahigh={3678640.34,-19729.80806,45.4118441,-0.00375106439,7.11445078e-007,-
7.233328850000001e-011,
      3.045637067e-015},
bhigh={116359.5961,-268.1058539},
R=89.30373037260763);

constant IdealGases.Common.DataRecord C6H6(
  name="C6H6",
  MM=0.07811184,
  Hf=1061042.730525872,
  H0=181735.4577743912,
  Tlimit=1000,
  alow={-167734.0902,4404.50004,-37.1737791,0.1640509559,-0.0002020812374,
        1.307915264e-007,-3.4442841e-011},
  blow={-10354.55401,216.9853345},
  ahigh={4538575.72,-22605.02547,46.940073,-0.004206676830000001,
        7.90799433e-007,-7.9683021e-011,3.32821208e-015},
  bhigh={139146.4686,-286.8751333},
  R=106.4431717393932);

constant IdealGases.Common.DataRecord C6H5OH_phenol(
  name="C6H5OH_phenol",
  MM=0.09411124,
  Hf=-1024309.104842312,
  H0=185915.0936700016,
  Tlimit=1000,
  alow={-120941.8144,3378.29227,-29.91846148,0.1567802942,-0.0001970937198,
        1.291815064e-007,-3.42435126e-011},
  blow={-27793.87017,179.9708977},
  ahigh={4462081.569999999,-21655.21026,48.1501505,-0.00356828243,
        6.32717573e-007,-6.039905720000001e-011,2.399168678e-015},
  bhigh={111372.6204,-286.0454446},
  R=88.34727924103434);

constant IdealGases.Common.DataRecord C6H10_cyclo(
  name="C6H10_cyclo",
  MM=0.08214360000000001,
  Hf=-55999.49356979728,
  H0=210251.1212072517,
  Tlimit=1000,
  alow={-375028.221,7643.87675,-55.1043476,0.2166231405,-0.0002545452198,
        1.607607304e-007,-4.18524841e-011},
  blow={-36610.7301,320.240946},
  ahigh={7510128.98,-35568.5821,67.03034940000001,-0.00704535187,
        1.3581664e-006,-1.400074492e-010,5.964551700000001e-015},
  bhigh={206027.4332,-423.819766},
  R=101.218743760926);

constant IdealGases.Common.DataRecord C6H12_1_hexene(
  name="C6H12_1_hexene",
  MM=0.08415948000000001,
  Hf=-498458.4030224521,
  H0=311788.9986962847,
```

```

Tlimit=1000,
alow={-666883.165,11768.64939,-72.70998330000001,0.2709398396,-
0.00033332464,
2.182347097e-007,-5.85946882e-011},
blow={-62157.8054,428.682564},
ahigh={733290.696,-14488.48641,46.7121549,0.00317297847,-5.24264652e-007,
4.28035582e-011,-1.472353254e-015},
bhigh={66977.4041,-262.3643854},
R=98.79424159940152);

constant IdealGases.Common.DataRecord C6H12_cyclo(
name="C6H12_cyclo",
MM=0.08415948000000001,
Hf=-1465075.5921971,
H0=208471.0361803566,
Tlimit=1000,
alow={-567998.704,10342.38704,-68.0004125,0.2387797658,-0.0002511890049,
1.425293184e-007,-3.40783319e-011},
blow={-64046.3516,393.480821},
ahigh={5225149.47,-33641.9458,71.74607469999999,-0.006698979119999999,
1.318443254e-006,-1.390794789e-010,6.06010224e-015},
bhigh={173253.7609,-454.681417},
R=98.79424159940152);

constant IdealGases.Common.DataRecord C6H13_n_hexyl(
name="C6H13_n_hexyl",
MM=0.08516742000000001,
Hf=294713.635801108,
H0=340306.1875069128,
Tlimit=1000,
alow={-1427278.22,22488.28093,-129.749224,0.427979733,-0.000555601318,
3.79125694e-007,-1.048462404e-010},
blow={-106026.1015,749.718676},
ahigh={5967938.62,-32990.2316,68.6907344,-0.00422500906,
5.496523820000001e-007,-2.292851471e-011,-5.08634189e-016},
bhigh={190697.8683,-418.5362},
R=97.62503079229123);

constant IdealGases.Common.DataRecord C6H14_n_hexane(
name="C6H14_n_hexane",
MM=0.0861753599999999,
Hf=-1936980.593988816,
H0=333065.0431863586,
Tlimit=1000,
alow={-581592.67,10790.97724,-66.3394703,0.2523715155,-0.0002904344705,
1.802201514e-007,-4.617223680000001e-011},
blow={-72715.4457,393.828354},
ahigh={-3106625.684,-7346.087920000001,46.94131760000001,0.001693963977,
2.068996667e-006,-4.21214168e-010,2.452345845e-014},
bhigh={523.750312,-254.9967718},
R=96.48317105956971);

constant IdealGases.Common.DataRecord C7H7_benzyl(
name="C7H7_benzyl",
MM=0.09113048,
Hf=2309874.808077385,
H0=203607.9695838319,

```

---

**1028 Modelica.Media.IdealGases.Common.SingleGasesData**

---

```
Tlimit=1000,
alow={-183676.4826,4102.46566,-32.024061,0.1588249575,-0.0001894466924,
1.203649671e-007,-3.136589637e-011},
blow={5266.33323,193.875172},
ahigh={5297322.16,-25999.09398,55.0975079,-0.00497766147,9.46315243e-007,-
9.639008860000001e-011,
4.06447042e-015},
bhigh={173838.2912,-334.382955},
R=91.23700434805129);

constant IdealGases.Common.DataRecord C7H8 (
  name="C7H8",
  MM=0.09213842,
  Hf=544506.8409030674,
  H0=194710.794910527,
  Tlimit=1000,
  alow={-287796.222,6133.941519999999,-45.74706759999999,0.1936895724,-
0.0002304305304,
1.459301178e-007,-3.7907961e-011},
  blow={-23084.02499,269.3915042},
  ahigh={6184538.350000001,-29902.84056,59.8200597,-0.00569698396,
1.080748416e-006,-1.098702235e-010,4.62474022e-015},
  bhigh={178204.7857,-369.808225},
  R=90.23892530390688);

constant IdealGases.Common.DataRecord C7H8O_cresol_mx (
  name="C7H8O_cresol_mx",
  MM=0.10813782,
  Hf=-1223420.261292488,
  H0=201949.1792973078,
  Tlimit=1000,
  alow={-244141.7503,5080.87784,-37.8904804,0.186494507,-0.0002269511868,
1.458989279e-007,-3.82240335e-011},
  blow={-40936.57829999999,228.1352234},
  ahigh={6017373.45,-28498.82774,60.81543070000001,-0.004996807109999999,
9.13427995e-007,-8.979444109999999e-011,3.66775697e-015},
  bhigh={147221.84,-367.485014},
  R=76.88773455947235);

constant IdealGases.Common.DataRecord C7H14_1_heptene (
  name="C7H14_1_heptene",
  MM=0.0981860599999999,
  Hf=-639194.6066478277,
  H0=313588.3036756949,
  Tlimit=1000,
  alow={-744940.284,13321.79893,-82.81694379999999,0.3108065994,-
0.000378677992,
2.446841042e-007,-6.488763869999999e-011},
  blow={-72178.8501,485.667149},
  ahigh={-1927608.174,-9125.024420000002,47.4817797,0.00606766053,-
8.684859080000001e-007,
5.81399526e-011,-1.473979569e-015},
  bhigh={26009.14656,-256.2880707},
  R=84.68077851377274);

constant IdealGases.Common.DataRecord C7H15_n_heptyl (
  name="C7H15_n_heptyl",
```

```

MM=0.099194,
Hf=44256.70907514567,
H0=338155.5336008226,
Tlimit=1000,
alow={-1671733.521,26400.1025,-152.6867707,0.5027121410000001,-
0.000652101403,
        4.44348811e-007,-1.227815006e-010},
blow={-127375.4623,878.393331999999},
ahigh={5444527.57,-34568.2929,76.3865194999999,-0.003298972,
      2.343496957e-007,2.467674021e-011,-3.162012849e-015},
bhigh={193907.9354,-464.142466},
R=83.82031171240196);

constant IdealGases.Common.DataRecord C7H16_n_heptane(
  name="C7H16_n_heptane",
  MM=0.10020194,
  Hf=-1874015.612871368,
  H0=331540.487140269,
  Tlimit=1000,
  alow={-612743.289,11840.85437,-74.87188599999999,0.2918466052,-
0.000341679549,
        2.159285269e-007,-5.65585273e-011},
  blow={-80134.0894,440.721332},
  ahigh={9135632.469999999,-39233.1969,78.8978085,-0.00465425193,
        2.071774142e-006,-3.4425393e-010,1.976834775e-014},
  bhigh={205070.8295,-485.110402},
  R=82.97715593131233);

constant IdealGases.Common.DataRecord C7H16_2_methylh(
  name="C7H16_2_methylh",
  MM=0.10020194,
  Hf=-1942078.167348856,
  H0=308576.8598891399,
  Tlimit=1000,
  alow={-710477.777,11912.5112,-73.45339440000001,0.2902952369,-
0.000346276768,
        2.260184498e-007,-6.12881392e-011},
  blow={-82021.477,432.004229},
  ahigh={1289912.969,-1784.340963,10.83537673,0.0527060923999999,-
1.886832314e-005,
        2.432255843e-009,-1.135553789e-013},
  bhigh={-16375.29884,-29.8186241},
  R=82.97715593131233);

constant IdealGases.Common.DataRecord C8H8_styrene(
  name="C8H8_styrene",
  MM=0.10414912,
  Hf=1423919.85645198,
  H0=201057.8677957144,
  Tlimit=1000,
  alow={-268693.052,6167.99947,-48.3605494,0.2182873229,-0.0002738561832,
        1.810084981e-007,-4.86775027e-011},
  blow={-11406.39978,281.7679014},
  ahigh={-6629183.62,15145.94166,1.609822364,0.033833186,-1.093737395e-005,
        1.338825116e-009,-6.03253492e-014},
  bhigh={-89973.2415,43.1128279},
  R=79.83237880454487);

```

---

**1030 Modelica.Media.IdealGases.Common.SingleGasesData**

---

```
constant IdealGases.Common.DataRecord C8H10_ethylbenz(
  name="C8H10_ethylbenz",
  MM=0.106165,
  Hf=281825.4603682946,
  H0=209862.0072528611,
  Tlimit=1000,
  allow={-469494, 9307.16836, -65.2176947, 0.2612080237, -0.000318175348,
         2.051355473e-007, -5.40181735e-011},
  blow={-40738.7021, 378.090436},
  ahighe={5551564.100000001, -28313.80598, 60.6124072, 0.001042112857, -
1.327426719e-006,
         2.166031743e-010, -1.142545514e-014},
  bhigh={164224.1062, -369.176982},
  R=78.31650732350586);
```

```
constant IdealGases.Common.DataRecord C8H16_1_octene(
  name="C8H16_1_octene",
  MM=0.11221264,
  Hf=-744924.9924072726,
  H0=315026.8989304592,
  Tlimit=1000,
  allow={-928190.522, 16409.74476, -101.5939534, 0.374800141, -0.00045908294,
         2.96533534e-007, -7.84044521e-011},
  blow={-89524.2608, 590.759427},
  ahighe={-4409336.07, -4383.678800000001, 49.39154259999999,
         0.007912339629999999, -7.88866951e-007, 9.97021235e-012, 1.913144872e-015},
  bhigh={-11226.19342, -257.7650649},
  R=74.09568119955115);
```

```
constant IdealGases.Common.DataRecord C8H17_n_octyl(
  name="C8H17_n_octyl",
  MM=0.11322058,
  Hf=-144143.4057306543,
  H0=336537.7566516617,
  Tlimit=1000,
  allow={-1934340.995, 30549.7983, -176.7903454, 0.5801596649999999, -
0.000751741401,
         5.11246903e-007, -1.410193662e-010},
  blow={-149889.4706, 1013.724329},
  ahighe={5632173.390000001, -38211.4367, 86.37927500000001, -0.00360893158,
         2.544260445e-007, 2.908638837e-011, -3.67954974e-015},
  bhigh={210313.547, -526.242283},
  R=73.43604846398067);
```

```
constant IdealGases.Common.DataRecord C8H18_n_octane(
  name="C8H18_n_octane",
  MM=0.11422852,
  Hf=-1827477.060895125,
  H0=330740.51909278,
  Tlimit=1000,
  allow={-698664.715, 13385.01096, -84.1516592, 0.327193666, -0.000377720959,
         2.339836988e-007, -6.01089265e-011},
  blow={-90262.2325, 493.922214},
  ahighe={6365406.949999999, -31053.64657, 69.6916234, 0.01048059637, -4.12962195e-
006,
         5.54322631999999e-010, -2.651436499e-014},
  bhigh={150096.8785, -416.989565},
  R=72.78805678301707);
```

```

constant IdealGases.Common.DataRecord C8H18_isooctane(
  name="C8H18_isooctane",
  MM=0.11422852,
  Hf=-1961068.916939482,
  H0=281628.440953275,
  Tlimit=1000,
  allow={-168875.8565, 3126.903227, -21.23502828, 0.1489151508, -0.0001151180135,
         4.47321617e-008, -5.55488207e-012},
  blow={-44680.6062, 141.7455793},
  ahight={13527650.32, -46633.7034, 77.95313179999999, 0.01423729984, -
5.073593909999999e-006,
         7.24823297e-010, -3.81919011e-014},
  bhigh={254117.8017, -493.388719},
  R=72.78805678301707);

constant IdealGases.Common.DataRecord C9H19_n_nonyl(
  name="C9H19_n_nonyl",
  MM=0.12724716,
  Hf=-291008.4594422383,
  H0=335284.496722756,
  Tlimit=1000,
  allow={-2194880.612, 34685.6578, -200.9261419, 0.658050311, -0.000852593001,
         5.79463896e-007, -1.597637099e-010},
  blow={-172319.4608, 1149.114017},
  ahight={5277361.74, -40257.0196, 94.57296720000001, -0.002940301447,
         6.97810699e-009, 6.80525024e-011, -5.90709533e-015},
  bhigh={216532.0614, -575.44495},
  R=65.34112038335474);

constant IdealGases.Common.DataRecord C10H8_naphthale(
  name="C10H8_naphthale",
  MM=0.12817052,
  Hf=1174841.141317052,
  H0=161605.6172667475,
  Tlimit=1000,
  allow={-260284.5316, 6237.40957, -52.2609504, 0.2397692776, -0.0002912244803,
         1.854944401e-007, -4.81661927e-011},
  blow={-11147.0088, 297.2139517},
  ahight={5906172.11, -31632.2924, 70.3034203, -0.00601886554, 1.142052144e-006, -
1.161605689e-010,
         4.89284402e-015},
  bhigh={196256.7046, -434.7848950000001},
  R=64.87039297336079);

constant IdealGases.Common.DataRecord C10H21_n_decyl(
  name="C10H21_n_decyl",
  MM=0.14127374,
  Hf=-408710.0688351565,
  H0=334273.0220067792,
  Tlimit=1000,
  allow={-2446511.152, 38700.813, -224.4388176, 0.734362497, -0.00095135256,
         6.46297033e-007, -1.781502862e-010},
  blow={-194163.3889, 1280.987834},
  ahight={4967237.76, -42424.6844, 102.8853417, -0.00232484818, -2.2842339e-007,
         1.056127364e-010, -8.0680659e-015},
  bhigh={223542.989, -625.5191159999999},
  R=58.85362700810497);

```

---

**1032 Modelica.Media.IdealGases.Common.SingleGasesData**

---

```
constant IdealGases.Common.DataRecord C12H9_o_bipheny(
  name="C12H9_o_bipheny",
  MM=0.15319986,
  Hf=2791973.830785485,
  H0=173559.8648719392,
  Tlimit=1000,
  alow={-359584.082,7661.37856,-58.7329046,0.2697557882,-0.000322975668,
         2.0329071e-007,-5.23131672e-011},
  blow={14584.14642,339.268711},
  ahigh={6736469.42,-36321.9435,82.102396,-0.00750234286,1.456251733e-006,-
1.507145019e-010,
         6.43647043e-015},
  bhigh={255552.4521,-504.246297},
  R=54.27206003974155);

constant IdealGases.Common.DataRecord C12H10_biphenyl(
  name="C12H10_biphenyl",
  MM=0.1542078,
  Hf=1181068.66189648,
  H0=173684.6190659617,
  Tlimit=1000,
  alow={-367103.405,8128.41259,-63.9099457,0.2901422744,-0.000350959074,
         2.230989996e-007,-5.78847029e-011},
  blow={-16792.63066,363.346419},
  ahigh={7480385.359999999,-39280.8723,86.61482219999999,-
0.007946398570000001,
         1.531868544e-006,-1.576450171e-010,6.70060273e-015},
  bhigh={243805.0641,-538.138149},
  R=53.91732454519163);

constant IdealGases.Common.DataRecord Ca (
  name="Ca",
  MM=0.040078,
  Hf=4436349.119217526,
  H0=154634.1633814063,
  Tlimit=1000,
  alow={0,0,2.5,0,0,0,0},
  blow={20638.92786,4.38454833},
  ahigh={7547341.239999999,-21486.42662,25.30849567,-0.01103773705,
         2.293249636e-006,-1.209075383e-010,-4.015333268e-015},
  bhigh={158586.2323,-160.9512955},
  R=207.4572583462249);

constant IdealGases.Common.DataRecord Caplus (
  name="Caplus",
  MM=0.0400774514,
  Hf=19308306.81513845,
  H0=154636.2800904052,
  Tlimit=1000,
  alow={0,0,2.5,0,0,0,0},
  blow={92324.1779,5.07767498},
  ahigh={3747070.82,-11747.07738,16.72546969,-0.00833479771,2.394593294e-006,
         -2.988243468e-010,1.356563002e-014},
  bhigh={166432.9088,-95.8282126},
  R=207.4600981239042);

constant IdealGases.Common.DataRecord CaBr (
```

```

name="CaBr",
MM=0.119982,
Hf=-207270.0988481606,
H0=82146.64699704957,
Tlimit=1000,
allow={569.3832620000001,-125.7513537,5.00150833,-0.001039868753,
1.377594927e-006,-8.94901396e-010,2.395658928e-013},
blow={-3728.10398,1.784864847},
ahigh={2783236.402,-8458.203729999999,14.3175656,-0.00543078168,
1.522218659e-006,-1.831824807e-010,7.86428866e-015},
bhigh={49312.7068,-65.37001100000001},
R=69.29766131586405);

constant IdealGases.Common.DataRecord CaBr2 (
  name="CaBr2",
  MM=0.199886,
  Hf=-1937090.641665749,
  H0=78060.49448185465,
  Tlimit=1000,
  allow={1572.504318,-223.8551236,8.38845789,-0.001944164774,2.397926088e-006,
  -1.55657235e-009,4.12430958e-013},
  blow={-47721.0661,-10.74965233},
  ahigh={-22198.46426,-4.46315533,7.50375511,-1.641654626e-006,
  3.87942806e-010,-4.68342426e-014,2.259325541e-018},
  bhigh={-48854.3682,-5.59032425},
  R=41.59606975976306);

constant IdealGases.Common.DataRecord CaCL (
  name="CaCL",
  MM=0.075531,
  Hf=-1373911.175543816,
  H0=127035.4556407303,
  Tlimit=1000,
  allow={6395.335260000001,-224.9042269,5.28327839,-0.001447983237,
  1.643742771e-006,-9.43010428e-010,2.237516133e-013},
  blow={-12701.69,-1.391964289},
  ahigh={1629182.545,-4766.22302,9.65892977,-0.002523044131,5.83354216e-007,
  4.10172699e-011,
  8.813177639999999e-017},
  bhigh={16625.99241,-33.98731174},
  R=110.0802584369332);

constant IdealGases.Common.DataRecord CaCLplus (
  name="CaCLplus",
  MM=0.0755304514,
  Hf=6185459.034606007,
  H0=123938.2636603705,
  Tlimit=1000,
  allow={2285.284542,-200.2872721,4.93699019,-0.000386097303,
  8.686661880000001e-008,1.47020568e-010,-7.68931587e-014},
  blow={55882.7795,-0.6359815341},
  ahigh={177028.0396,-589.137239,5.1033688,-0.0002356761168,5.48373287e-008,
  2.137973112e-014,-5.57288731e-016},
  bhigh={58534.17249999999,-2.191253644},
  R=110.0810579823968);

constant IdealGases.Common.DataRecord CaCL2 (

```

---

**1034 Modelica.Media.IdealGases.Common.SingleGasesData**

---

```
name="CaCL2",
MM=0.110984,
Hf=-4372193.081885677,
H0=133866.3501045195,
Tlimit=1000,
alow={11068.0203,-424.910825,9.13471186,-0.00349718531,
      4.24218462999999e-006,-2.719673231e-009,7.13826863e-013},
blow={-58503.4676,-18.08678294},
ahigh={-35957.6878,-8.796495630000001,7.50725691,-3.129465934e-006,
      7.322886480000001e-010,-8.776403220000001e-014,4.21048604e-018},
bhigh={-60667.7966,-8.553523451},
R=74.91595184891516);

constant IdealGases.Common.DataRecord CaF(
  name="CaF",
  MM=0.0590764032,
  Hf=-4678751.159989374,
  H0=154581.3472950229,
  Tlimit=1000,
  alow={31842.354,-491.823508999999,5.70112089,-0.001465138218,
        9.09293081e-007,-1.367585222e-010,-5.10632338e-014},
  blow={-31976.9411,-5.99666226},
  ahigh={519588.2669999999,-1512.321567,5.86927625,-0.000388252448,-
2.484582298e-008,
        3.74991596e-011,-3.47097645e-015},
  bhigh={-24888.14584,-8.6146637},
  R=140.7409989374573);

constant IdealGases.Common.DataRecord CaFplus(
  name="CaFplus",
  MM=0.0590758546,
  Hf=4412371.344688089,
  H0=152399.1156955688,
  Tlimit=1000,
  alow={41583.408,-562.766166,5.68752215,-0.001264710572,6.66908648e-007,-
6.92675282e-011,
        -3.97871462e-014},
  blow={33051.195,-6.97448501},
  ahigh={-139867.0845,219.1526375,4.21432434,0.0002142798048,-5.7786726e-008,
        9.98575631e-012,-6.05962188e-016},
  bhigh={28416.05897,2.740342729},
  R=140.7423059098666);

constant IdealGases.Common.DataRecord CaF2(
  name="CaF2",
  MM=0.0780748064,
  Hf=-10129111.36722332,
  H0=164093.7914640798,
  Tlimit=1000,
  alow={59093.5229,-1019.733042,10.13165309,-0.00550350644,5.62931998e-006,-
3.115401465e-009,
        7.205525940000001e-013},
  blow={-91926.0818999999,-26.3950595},
  ahigh={-82804.0944,-31.15950071,7.02357737,-9.521000020000001e-006,
        2.118160895e-009,-2.440713888e-013,1.135135468e-017},
  bhigh={-97294.66770000001,-7.54624334},
  R=106.4936614431362);
```

```

constant IdealGases.Common.DataRecord CaH(
  name="CaH",
  MM=0.04108594,
  Hf=5583640.169848858,
  H0=211875.5223806489,
  Tlimit=1000,
  allow={-45137.8223, 762.942921, -1.280874223, 0.01318774659, -1.481595334e-005,
         8.53657322e-009, -1.989958945e-012},
  blow={23003.78814, 30.53421525},
  ahigh={-2696952.529, 8607.05975, -7.02745482, 0.007467916309999999, -
2.318610699e-006,
         3.42307242e-010, -1.892679792e-014},
  bhigh={-27738.19107, 78.45822010000001},
  R=202.367817311713);

constant IdealGases.Common.DataRecord CaI(
  name="CaI",
  MM=0.16698247,
  Hf=72961.67076699728,
  H0=59995.04019793216,
  Tlimit=1000,
  allow={-826.179018, -85.08975899999999, 4.86417169, -0.000786316965,
         1.100493506e-006, -7.34803061e-010, 2.010892767e-013},
  blow={523.675344, 3.58539168},
  ahigh={1771071.309, -5683.64373, 11.53857476, -0.0041940202, 1.291359961e-006, -
1.730043579e-010,
         8.38076235e-015},
  bhigh={35832.932, -44.0591757},
  R=49.79248420507854);

constant IdealGases.Common.DataRecord CaI2(
  name="CaI2",
  MM=0.29388694,
  Hf=-882378.4581921197,
  H0=54616.22418471539,
  Tlimit=1000,
  allow={-1097.851401, -145.9821761, 8.08934184, -0.001305332923, 1.624199331e-006,
         -1.061194001e-009, 2.825551217e-013},
  blow={-32726.9298, -6.98171494},
  ahigh={-16237.27104, -2.859530722, 7.50243219, -1.071178707e-006,
         2.544559095e-010, -3.083631004e-014, 1.491836724e-018},
  bhigh={-33463.4117, -3.56730574},
  R=28.29139668472509);

constant IdealGases.Common.DataRecord CaO(
  name="CaO",
  MM=0.0560774,
  Hf=677729.4953047038,
  H0=159656.2608109506,
  Tlimit=1000,
  allow={38897.3307, -483.567735, 5.07771325, 0.000307623525, -1.159759897e-006,
         8.49343333999999e-010, -1.495333366e-013},
  blow={5937.643480000001, -3.95532073},
  ahigh={-49131061.7, 149586.595, -168.1654149, 0.09381950259999999, -
2.455529428e-005,
         3.07498072e-009, -1.485914237e-013},
  bhigh={-946151.172, 1235.694769},
  R=148.2677870229362);

```

---

**1036 Modelica.Media.IdealGases.Common.SingleGasesData**

---

```
constant IdealGases.Common.DataRecord CaOplus(
  name="CaOplus",
  MM=0.0560768514,
  Hf=12665431.88977975,
  H0=163402.7191476731,
  Tlimit=1000,
  allow={109806.0332,-1459.992448,9.88077794,-0.009157564600000001,
         8.707582560000001e-006,-4.39041108e-009,9.26485465999999e-013},
  blow={91500.57350000001,-30.09947701},
  ahighelement={939784.313,-2993.362243,8.33619182,-0.002303295087,7.37396753e-007,-
1.052888279e-010,
              5.25713841e-015},
  bhigh={102809.8401,-24.61547836},
  R=148.2692375271269);

constant IdealGases.Common.DataRecord CaOH(
  name="CaOH",
  MM=0.05708534000000001,
  Hf=-3035934.444815429,
  H0=193564.5298775482,
  Tlimit=1000,
  allow={46200.289,-928.567282,9.175828770000001,-0.0039628288,
         2.505447308e-006,3.85206821e-011,-3.35277847e-013},
  blow={-17980.09717,-25.3370485},
  ahighelement={1979972.994,-5598.88099,11.51348706,-0.001668264707,3.31257391e-007,-
1.789056647e-011,-3.58071641e-016},
  bhigh={13401.96822,-46.46084260000001},
  R=145.6498638704788);

constant IdealGases.Common.DataRecord CaOHplus(
  name="CaOHplus",
  MM=0.0570847914,
  Hf=6533055.317427332,
  H0=194407.6299103372,
  Tlimit=1000,
  allow={48843.4266,-983.2415100000001,9.61018295,-0.00520436066,
         4.26304982e-006,-1.197590878e-009,9.02841188999999e-015},
  blow={47950.56340000001,-28.27077334},
  ahighelement={863761.537,-2347.302046,7.9819112,0.0001003160947,-6.24007417e-008,-
1.019540904e-011,-5.69892528e-016},
  bhigh={58305.8011999999,-21.52181937},
  R=145.6512636043372);

constant IdealGases.Common.DataRecord Ca_OH_2(
  name="Ca_OH_2",
  MM=0.07409268000000001,
  Hf=-8075546.315776401,
  H0=223939.5443652463,
  Tlimit=1000,
  allow={83892.5753999999,-1791.902135,16.21891031,-0.00784085717,
         5.09511161e-006,-6.95548755e-011,-6.13215264e-013},
  blow={-66004.0563,-60.7091392},
  ahighelement={1721854.884,-4702.21767,13.96947691,0.0001983791405,-1.243050592e-
007,
              2.033402404e-011,-1.137157819e-015},
  bhigh={-44437.6135,-52.8744486},
  R=112.2171852873995);
```

```

constant IdealGases.Common.DataRecord CaS (
  name="CaS",
  MM=0.072143,
  Hf=1683812.885519038,
  H0=129743.8836754779,
  Tlimit=1000,
  allow={23209.90615,-451.8446080000001,6.17233375,-0.00359885202,
         4.82373170999999e-006,-3.63173354e-009,1.212849867e-012},
  blow={15545.97691,-7.686936462},
  ahighe={-15683532.14,52938.5581,-63.2246724,0.0402772638,-1.114855081e-005,
          1.464389344e-009,-7.37607897e-014},
  bhigh={-317202.267,479.9606012},
  R=115.2498787131115);

constant IdealGases.Common.DataRecord Ca2 (
  name="Ca2",
  MM=0.08015600000000001,
  Hf=4263752.382853436,
  H0=140639.9520934178,
  Tlimit=1000,
  allow={-85822.2862,158.818896,11.03952055,-0.0333319676,5.34593881e-005,-
4.01157323999999e-008,
         1.160486682e-011},
  blow={37703.508,-23.97744561},
  ahighe={240596.6267,57.7580382,2.347436675,0.0001199275034,-4.32915031e-008,
          7.01530269e-012,-3.70566032e-016},
  bhigh={40818.754,18.95399607},
  R=103.7286291731124);

constant IdealGases.Common.DataRecord Cd (
  name="Cd",
  MM=0.112411,
  Hf=994564.5888747543,
  H0=55131.86431932818,
  Tlimit=1000,
  allow={-0.0001081751543,1.816433041e-006,2.49999989,3.129989231e-011,-
4.60071016e-014,
         3.40704874e-017,-9.989497436e-021},
  blow={12700.99766,5.93154976},
  ahighe={-269975.7467,786.600114,1.628169079,0.000459412329,-1.150420443e-007,
          1.074836707e-011,8.790199554999999e-017},
  bhigh={7675.14826,12.20006052},
  R=73.96493225751928);

constant IdealGases.Common.DataRecord Cdplus (
  name="Cdplus",
  MM=0.1124104514,
  Hf=8769240.899961371,
  H0=55132.13338097137,
  Tlimit=1000,
  allow={0.000409834494,-4.34358162e-006,2.500000019,-4.389036810000001e-011,
         5.5639692e-014,-3.63822826e-017,9.60788199499999e-021},
  blow={117812.9439,6.62468945},
  ahighe={14848.80812,-46.6915368,2.557883006,-3.6247017e-005,1.21374757e-008,
          -2.072802814e-012,1.420665367e-016},
  bhigh={118107.0109,6.21641448},
  R=73.96529323073246);

```

---

**1038 Modelica.Media.IdealGases.Common.SingleGasesData**

---

```
constant IdealGases.Common.DataRecord CL(
  name="CL",
  MM=0.03545300000000001,
  Hf=3421459.396948072,
  H0=176898.6545567371,
  Tlimit=1000,
  alow={22762.15854,-216.8413293,2.745185115,0.002451101694,-5.45801199e-006,
        4.41798688e-009,-1.288134004e-012},
  blow={15013.57068,3.102963457},
  ahighelement1=-169793.293,element2=608.172646,element3=2.12866409,element4=0.0001307367034,-2.644883596e-008,
        2.842504775e-012,-1.252911731e-016},
  bhigh={9934.3874,8.844772103},
  R=234.5209714269596);

constant IdealGases.Common.DataRecord CLplus(
  name="CLplus",
  MM=0.0354524514,
  Hf=38891517.52705033,
  H0=180138.0369426302,
  Tlimit=1000,
  alow={103469.7859,-1293.758873,8.18670269000001,-0.0099160146,
        9.20847237e-006,-4.50742624e-009,9.18212788e-013},
  blow={171475.878,-27.66417931},
  ahighelement1=40564.0948,element2=-49.6016572,element3=3.10165363,-0.000586873829,element4=2.252039316e-007,-
  3.29970302e-011,
        1.708780842e-015},
  bhigh={165298.2337,2.574942598},
  R=234.5246004624662);

constant IdealGases.Common.DataRecord CLminus(
  name="CLminus",
  MM=0.0354535486,
  Hf=-6599000.135066875,
  H0=174804.1661477012,
  Tlimit=1000,
  alow={0,0,2.5,0,0,0,0},
  blow={-28883.89093,4.200642023},
  ahighelement1=0,element2=0,element3=2.5,element4=0,element5=0,element6=0},
  bhigh={-28883.89093,4.200642023},
  R=234.5173425037629);

constant IdealGases.Common.DataRecord CLCN(
  name="CLCN",
  MM=0.0614703999999999,
  Hf=2183164.580025509,
  H0=173563.8941669487,
  Tlimit=1000,
  alow={72740.43429999999,-1297.344947,11.07676527,-0.01108430216,
        1.614009477e-005,-1.088916772e-008,2.830952246e-012},
  blow={20843.88986,-35.97370046},
  ahighelement1=346757.312,element2=-1957.85737,element3=8.807806149999999,-0.0004388362779999999,
        1.024761895e-007,-1.110675026e-011,4.98617942e-016},
  bhigh={25819.23944,-26.36885363},
  R=135.2597673026367);

constant IdealGases.Common.DataRecord CLF(
  name="CLF",
```

```

MM=0.0544514032,
Hf=-1022948.71622335,
H0=163597.3818210069,
Tlimit=1000,
alow={33522.1081,-368.83119,4.27228862,0.002549434508,-4.45683089e-006,
      3.41385412e-009,-9.8386852e-013},
blow={-5839.36969,0.2318014199},
ahigh={3045867.173,-9979.32826,16.84162254,-0.007465850620000001,
       2.336213612e-006,-3.36586546e-010,1.763082415e-014},
bhigh={54471.20800000001,-87.07987094000001},
R=152.6952752615198);

constant IdealGases.Common.DataRecord CLF3 (
  name="CLF3",
  MM=0.0924482096,
  Hf=-1780456.330221889,
  H0=148498.8087860168,
  Tlimit=1000,
  alow={128517.5171,-2140.445721,14.93096474,-0.00604247104,3.71896861e-006,-
8.0532011e-010,
      -7.98560053e-014},
  blow={-11384.5964,-55.94838353},
  ahigh={-229495.6235,-122.0741,10.09124644,-3.64972475e-005,8.05891075e-009,
         -9.23084995e-013,4.27256949e-017},
  bhigh={-22828.40447,-25.11605479},
  R=89.93653891161999);

constant IdealGases.Common.DataRecord CLF5 (
  name="CLF5",
  MM=0.130445016,
  Hf=-1824523.521849237,
  H0=137455.5851179473,
  Tlimit=1000,
  alow={245970.3939,-4315.34807,27.27005972,-0.01670193839,1.421668208e-005,-
6.426479740000001e-009,
      1.183099799e-012},
  blow={-10714.23726,-126.7466837},
  ahigh={-426932.7449999999,-205.805739,16.15465278,-6.21268818e-005,
         1.3766379595e-008,-1.581335709e-012,7.33642249e-017},
  bhigh={-33614.666,-57.57531175},
  R=63.73928460402045);

constant IdealGases.Common.DataRecord CLO (
  name="CLO",
  MM=0.0514524,
  Hf=1975051.018028314,
  H0=185066.4886380422,
  Tlimit=1000,
  alow={-16872.68145,257.3812247,2.17584612,0.006432061130000001,-
8.568249500000001e-006,
      5.764971250000001e-009,-1.545771209e-012},
  blow={9829.518979999999,13.86010503},
  ahigh={409376.052,-1765.985112,7.08790063,-0.001828450169,7.1038181e-007,-
1.209332942e-010,
      7.07644104e-015},
  bhigh={21518.91784,-16.68645548},
  R=161.5954163459819);

```

---

**1040 Modelica.Media.IdealGases.Common.SingleGasesData**

---

```
constant IdealGases.Common.DataRecord CL02 (
  name="CL02",
  MM=0.06745180000000001,
  Hf=1556667.131195906,
  H0=160123.9107036432,
  Tlimit=1000,
  allow={-11272.77696, 390.303203, -0.384301853, 0.02108677947, -2.793137755e-005,
         1.841289286e-008, -4.83977915e-012},
  blow={9756.93367, 29.132627},
  ahighelement={-163379.3802, -316.148067, 7.00978726, 0.0002837971144, -1.179925338e-007,
               2.920383252e-011, -2.024030202e-015},
  bhigh={11849.46452, -10.91168827},
  R=123.2653835776065);

constant IdealGases.Common.DataRecord CL2 (
  name="CL2",
  MM=0.07090600000000001,
  Hf=0,
  H0=129482.8364313316,
  Tlimit=1000,
  allow={34628.1517, -554.7126520000001, 6.20758937, -0.002989632078,
         3.17302729e-006, -1.793629562e-009, 4.260043590000001e-013},
  blow={1534.069331, -9.438331107},
  ahighelement={6092569.42, -19496.27662, 28.54535795, -0.01449968764, 4.46389077e-006, -6.35852586e-010,
               3.32736029e-014},
  bhigh={121211.7724, -169.0778824},
  R=117.2604857134798);

constant IdealGases.Common.DataRecord CL2O (
  name="CL2O",
  MM=0.08690539999999999,
  Hf=909034.4213363037,
  H0=134577.1609129007,
  Tlimit=1000,
  allow={77988.554, -1182.537879, 9.52651466, -0.002596163176,
         8.696781310000001e-007, 4.484098269999999e-010, -3.044511967e-013},
  blow={13767.29572, -24.84690718},
  ahighelement={-127174.3461, -67.4196322, 7.05002245, -1.988084386e-005,
               4.366209050000001e-009, -4.978580680000001e-013, 2.295679186e-017},
  bhigh={7387.2893, -8.797477254},
  R=95.67267396502405);

constant IdealGases.Common.DataRecord Co (
  name="Co",
  MM=0.0589332,
  Hf=7269953.099441402,
  H0=107914.8595358813,
  Tlimit=1000,
  allow={-2598.939184, 246.1989844, -0.610605837, 0.01393005772, -2.210012979e-005,
         1.623755261e-008, -4.534904351e-012},
  blow={49846.1376, 22.57584199},
  ahighelement={1381841.305, -3756.03668, 6.65713065, -0.001269246675, 1.464092329e-007,
               6.57494657e-012, -1.102384178e-015},
  bhigh={74944.42909999999, -22.58500836},
  R=141.0829888755405);
```

```

constant IdealGases.Common.DataRecord Coplus(
  name="Coplus",
  MM=0.0589326513999999,
  Hf=20243503.02691455,
  H0=106757.5758181482,
  Tlimit=1000,
  allow={102849.416,-874.473126,4.27950028,0.002225857835,-7.45727457e-006,
    7.27922195e-009,-2.347541963e-012},
  blow={147489.5959,-5.67901922},
  ahighe={2907386.174,-8619.705749999999,11.88134934,-0.00351064742,
    5.74800468e-007,-2.534065135e-011,2.976607469e-016},
  bhigh={197741.9221,-60.9653344},
  R=141.0843022073838);

constant IdealGases.Common.DataRecord Cominus(
  name="Cominus",
  MM=0.0589337486,
  Hf=6081648.978968902,
  H0=107014.6079253475,
  Tlimit=1000,
  allow={34594.9376,-135.5041712,1.116330898,0.009042761829999999,-
    1.454296726e-005,
    9.99494063e-009,-2.595633259e-012},
  blow={43370.3782,12.70494975},
  ahighe={-574139.417,1763.109207,1.415561988,0.000377686969,-7.53417998e-008,
    7.99570154e-012,-3.49044e-016},
  bhigh={30834.93114,16.34360353},
  R=141.0816755681481);

constant IdealGases.Common.DataRecord Cr(
  name="Cr",
  MM=0.0519961,
  Hf=7644419.485307553,
  H0=119190.2469608298,
  Tlimit=1000,
  allow={1335.658217,-21.02424026,2.631908173,-0.000424626325,7.43919416e-007,
    -6.76393163e-010,2.507855625e-013},
  blow={47158.6664,6.00542545},
  ahighe={-11202207.89,34011.63690000001,-36.5706217,0.02110296902,-
    5.51818014e-006,
    7.17360171e-010,-3.505127367e-014},
  bhigh={-168899.344,286.4481267},
  R=159.9056852340849);

constant IdealGases.Common.DataRecord Crplus(
  name="Crplus",
  MM=0.0519955514,
  Hf=20319944.67895959,
  H0=119191.5045255199,
  Tlimit=1000,
  allow={181.9187467,-2.188843517,2.510676511,-2.706791825e-005,
    3.76849263e-008,-2.736784742e-011,8.115389931999999e-015},
  blow={126338.0825,6.50627617},
  ahighe={3342330.79,-10642.61051,15.57884307,-0.00770897148,2.158300274e-006,
    -2.36810811e-010,8.952805604e-015},
  bhigh={193299.767,-86.0435667},
  R=159.9073723833997);

```

```
constant IdealGases.Common.DataRecord Crminus(
  name="Crminus",
  MM=0.0519966486,
  Hf=6289317.423431017,
  H0=119188.989422676,
  Tlimit=1000,
  allow={0,0,2.5,0,0,0,0},
  blow={38586.2787,6.56683537},
  ahight={0,0,2.5,0,0,0,0},
  bhigh={38586.2787,6.56683537},
  R=159.9039981203712);

constant IdealGases.Common.DataRecord CrN(
  name="CrN",
  MM=0.0660028,
  Hf=7651323.883229196,
  H0=132991.1609810493,
  Tlimit=1000,
  allow={-8239.129220000001,300.8144202,0.514787492,0.01129636791,-
1.515183504e-005,
         1.010997229e-008,-2.68777575e-012},
  blow={58456.2812,22.98028696},
  ahight={1110672.49,-3690.47854,8.59905668,-0.002125587223,5.28235848e-007,-
4.92113915e-011,
         1.404331106e-015},
  bhigh={82548.04580000001,-27.99968411},
  R=125.9715042392141);

constant IdealGases.Common.DataRecord CrO(
  name="CrO",
  MM=0.0679955,
  Hf=2744024.501621431,
  H0=144849.5562206323,
  Tlimit=1000,
  allow={9373.33411,136.9743818,1.621443428,0.008814095959999999,-1.23284536e-
005,
         8.497960940000001e-009,-2.315804197e-012},
  blow={20909.48871,17.81935787},
  ahight={1092367.332,-3749.75865,9.00787021,-0.002545445236,
         6.928051680000001e-007,-6.390831950000001e-011,1.659741645e-015},
  bhigh={44470.9821,-29.42600453},
  R=122.2797391003817);

constant IdealGases.Common.DataRecord CrO2(
  name="CrO2",
  MM=0.0839949,
  Hf=-1286307.085311132,
  H0=127315.6941671459,
  Tlimit=1000,
  allow={35486.299,-229.8628537,2.286289393,0.01616929338,-2.34519891e-005,
         1.631365714e-008,-4.44426962e-012},
  blow={-12789.1284,14.42954956},
  ahight={-432710.914,191.5584657,7.18824737,-0.000569484619,3.54636613e-007,-
5.65512306e-011,
         2.908946349e-015},
  bhigh={-17433.39545,-10.0642472},
  R=98.9878194985648);
```

```

constant IdealGases.Common.DataRecord CrO3 (
  name="CrO3",
  MM=0.0999942999999999,
  Hf=-3220554.411601461,
  H0=130408.1532647361,
  Tlimit=1000,
  allow={41830.2006,-505.934885,4.43271567,0.01995079387,-2.920597649e-005,
         2.03933028e-008,-5.580086630000001e-012},
  blow={-37697.024,0.865382727999999},
  ahighe={-628331.401,692.815817999999,8.971274599999999,0.000682336564,-
2.235048825e-007,
         3.36678579e-011,-1.614026492e-015},
  bhigh={-47238.802,-19.45305345},
  R=83.1494595191926);

constant IdealGases.Common.DataRecord CrO3minus (
  name="CrO3minus",
  MM=0.0999948485999999,
  Hf=-6328834.473579072,
  H0=134244.3054611516,
  Tlimit=1000,
  allow={187345.5703,-2300.722991,13.43953022,-0.001663085846,-1.443973096e-
006,
         2.045898933e-009,-6.8375553e-013},
  blow={-66301.1158,-49.306442},
  ahighe={-649968.203,452.201762,9.9805759,-0.000421413673,2.631277945e-007,-
4.594543650000001e-011,
         2.659097549e-015},
  bhigh={-83500.1973,-24.648109},
  R=83.14900333775795);

constant IdealGases.Common.DataRecord Cs (
  name="Cs",
  MM=0.13290545,
  Hf=575597.1632465035,
  H0=46630.35263038498,
  Tlimit=1000,
  allow={54.6658407,-0.827934604,2.50494221,-1.49462069e-005,2.425976774e-008,
         -2.013172322e-011,6.704271991e-015},
  blow={8459.321389999999,6.848825772},
  ahighe={6166040.899999999,-18961.75522,24.83229903,-0.01251977234,
         3.30901739e-006,-3.35401202e-010,9.626500908000001e-015},
  bhigh={128511.1231,-152.2942188},
  R=62.559300615588);

constant IdealGases.Common.DataRecord Csplus (
  name="Csplus",
  MM=0.1329049014,
  Hf=3449096.483058675,
  H0=46630.54510945222,
  Tlimit=1000,
  allow={0,0,2.5,0,0,0,0},
  blow={54387.3782,6.182757992},
  ahighe={0,0,2.5,0,0,0,0},
  bhigh={54387.3782,6.182757992},
  R=62.55955884558522);

```

---

**1044 Modelica.Media.IdealGases.Common.SingleGasesData**

---

```
constant IdealGases.Common.DataRecord Csminus (
  name="Csminus",
  MM=0.1329059986,
  Hf=186577.1918589685,
  H0=46630.16015290675,
  Tlimit=1000,
  allow={0,0,2.5,0,0,0,0},
  blow={2237.029001,6.182770382},
  ahight={0,0,2.5,0,0,0,0},
  bhigh={2237.029001,6.182770382},
  R=62.5590423877226);

constant IdealGases.Common.DataRecord CsBO2 (
  name="CsBO2",
  MM=0.17571525,
  Hf=-3909176.386227149,
  H0=82326.91243361063,
  Tlimit=1000,
  allow={41936.0958,-666.423740000001,8.134835880000001,0.002902960302,-
7.88146792e-007,
         -8.327694539999999e-010,4.44171911e-013},
  blow={-81223.2007,-11.21622595},
  ahight={89755.79890000001,-1656.134283,11.16468934,-0.000447636699,
         9.621091809999999e-008,-1.081470115e-011,4.93844674e-016},
  bhigh={-76104.79670000001,-30.36616012},
  R=47.31787366207544);

constant IdealGases.Common.DataRecord CsBr (
  name="CsBr",
  MM=0.21280945,
  Hf=-971897.0421661255,
  H0=48898.79185346327,
  Tlimit=1000,
  allow={1639.263647,-69.4747797,4.86005872,-0.0008587780150000001,
         1.37517755e-006,-9.89176698e-010,2.895433432e-013},
  blow={-25895.50471,4.46535569},
  ahight={-152832.9963,1259.712657,1.704268408,0.002776255318,-1.228389838e-
006,
         2.698111786e-010,-1.98353448e-014},
  bhigh={-33278.4203,24.84116659},
  R=39.07003189942927);

constant IdealGases.Common.DataRecord CsCL (
  name="CsCL",
  MM=0.16835845,
  Hf=-1438768.757968489,
  H0=60175.90444673255,
  Tlimit=1000,
  allow={-25381.65292,297.2839326,2.667813848,0.00558327241,-8.52900509e-006,
         6.595191390000001e-009,-1.989715823e-012},
  blow={-31892.5145,15.11365709},
  ahight={-3674923.48,11861.52941,-10.63287842,0.00980411791999999,-
3.27779746e-006,
         5.53423339e-010,-3.4012315e-014},
  bhigh={-104865.8658,111.4096064},
  R=49.38553425741328);
```

```

constant IdealGases.Common.DataRecord CsF(
  name="CsF",
  MM=0.1519038532,
  Hf=-2397667.184389764,
  H0=63494.87387460162,
  Tlimit=1000,
  allow={18436.85799,-404.240939,6.38317886,-0.00467432337,6.63013401e-006,-
4.76090553e-009,
         1.374569797e-012},
  blow={-43184.8959,-7.22659323},
  ahigh={-1850863.231,5625.298800000001,-2.250022212,0.00410924354,-
1.250243837e-006,
         1.941483152e-010,-1.071166179e-014},
  bhigh={-80798.2779,51.3555945},
  R=54.73509608115722);

constant IdealGases.Common.DataRecord CsH(
  name="CsH",
  MM=0.13391339,
  Hf=865858.9406182609,
  H0=66058.4128293668,
  Tlimit=1000,
  allow={16205.44411,-70.8701628,2.480847135,0.00702198599,-1.007126309e-005,
         7.09063903e-009,-1.950395735e-012},
  blow={13427.77328,9.89424717},
  ahigh={-911214.6649999999,3576.47275,-1.258058765,0.00431485515,-
1.407820502e-006,
         2.154598897e-010,-1.254525606e-014},
  bhigh={-9158.717270000001,39.0880003},
  R=62.08842894650043);

constant IdealGases.Common.DataRecord CsI(
  name="CsI",
  MM=0.25980992,
  Hf=-586274.2769791085,
  H0=40607.09845105222,
  Tlimit=1000,
  allow={-4072.68565,23.30073667,4.38727769,0.000341208429,-2.175588153e-007,
         8.906786999999999e-011,-2.182639837e-015},
  blow={-19787.66885,8.074657820000001},
  ahigh={4511259.05,-13417.06362,19.80984712,-0.008359855550000002,
         2.348201245e-006,-2.874285248e-010,1.20808775e-014},
  bhigh={65734.3913,-102.1032592},
  R=32.0021344835486);

constant IdealGases.Common.DataRecord CsLi(
  name="CsLi",
  MM=0.13984645,
  Hf=1159459.707414811,
  H0=73946.27464622805,
  Tlimit=1000,
  allow={1368.709568,-74.5129706,4.84562093,-0.0005573667210000001,
         4.95332676e-007,4.36906891e-010,-4.60814577e-013},
  blow={18505.77392,2.112012017},
  ahigh={7481630.23,-28852.97839,46.3110499,-0.02784288108,8.78741349e-006,-
1.264901197e-009,
         6.72018628e-014},
  bhigh={194513.7819,-285.7846194},

```

---

**1046 Modelica.Media.IdealGases.Common.SingleGasesData**

---

```
R=59.45429433496524);

constant IdealGases.Common.DataRecord CsNO2 (
  name="CsNO2",
  MM=0.17891095,
  Hf=-1175667.716257725,
  H0=89319.13893476056,
  Tlimit=1000,
  allow={-71060.1044,1272.448257,-2.447362349,0.03064441948,-3.69922594e-005,
    2.259121812e-008,-5.58063591e-012},
  blow={-33133.7092,49.54693336},
  ahight={-163350.3154,-846.8879579999999,10.62853461,-0.0002508799215,
    5.54125577e-008,-6.35559195e-012,2.946880031e-016},
  bhigh={-24030.40739,-24.43488888},
  R=46.4726837569193};

constant IdealGases.Common.DataRecord CsNO3 (
  name="CsNO3",
  MM=0.19491035,
  Hf=-1634014.76627588,
  H0=84980.65905684332,
  Tlimit=1000,
  allow={-26772.19779,901.169269,-3.26534082,0.04298377699999999,-5.38675138e-
005,
    3.38452752e-008,-8.56522366e-012},
  blow={-44053.0414,51.18862689},
  ahight={-314751.7676,-1367.011655,14.01012848,-0.000401864713,8.8536147e-008,
    -1.013466457e-011,4.69175873e-016},
  bhigh={-35490.2667,-45.00692791},
  R=42.65792965843015);

constant IdealGases.Common.DataRecord CsNa (
  name="CsNa",
  MM=0.15589522,
  Hf=807640.7281762712,
  H0=68649.50060688198,
  Tlimit=1000,
  allow={25217.38609,-429.5362489999999,7.22309474,-0.008460463000000001,
    1.443905157e-005,-1.143624855e-008,3.18372193e-012},
  blow={15790.99265,-8.83879029999999},
  ahight={3879556.79,-17801.67288,34.913558,-0.02320080793,
    7.861975840000001e-006,-1.171011257e-009,6.323927369999999e-014},
  bhigh={119577.6556,-200.1754433},
  R=53.33371991777555);

constant IdealGases.Common.DataRecord CsO (
  name="CsO",
  MM=0.14890485,
  Hf=252425.0754760506,
  H0=66049.68206206849,
  Tlimit=1000,
  allow={8683.95881,425.027817,-3.4840553,0.0380196983,-6.651927559999999e-005,
    5.17347219e-008,-1.512567066e-011},
  blow={1969.690015,42.4079614},
  ahight={837554.435000001,-2418.205772,8.908065349999999,-0.00337818004,
    1.312755917e-006,-2.150508925e-010,1.21913898e-014},
  bhigh={18094.93447,-24.6085858},
```

```

R=55.83748279522124);

constant IdealGases.Common.DataRecord CsOH(
  name="CsOH",
  MM=0.14991279,
  Hf=-1707659.499899908,
  H0=78943.82460629277,
  Tlimit=1000,
  allow={9386.960789999999,-500.935426,8.30135198,-0.00323559684,
    2.612406777e-006,-4.95061341e-010,-1.038364927e-013},
  blow={-30257.22427,-17.42194837},
  ahight={896717.113,-2323.978587,7.95964487,0.0001101149275,-
6.46601396999999e-008,
  1.045968201e-011,-5.82251417e-016},
  bhigh={-17362.58234,-18.64239624},
  R=55.46205897442107);

constant IdealGases.Common.DataRecord CsRb(
  name="CsRb",
  MM=0.21837325,
  Hf=510489.8699817858,
  H0=50244.85370804345,
  Tlimit=1000,
  allow={-4910.92268,-9.20901555,5.19337789,-0.00439024699,1.254790436e-005,-
1.414878514e-008,
  5.10144131e-012},
  blow={12004.98171,5.36154293},
  ahight={-13286933.78,34145.7947,-23.73696406,0.0070704654,3.86345097e-007,-
3.158132986e-010,
  2.667747207e-014},
  bhigh={-212400.8997,223.9970251},
  R=38.07459017988696);

constant IdealGases.Common.DataRecord Cs2(
  name="Cs2",
  MM=0.2658109,
  Hf=411587.083148208,
  H0=41492.40305796338,
  Tlimit=1000,
  allow={-46741.5873,595.201951,1.895333975,0.00408206581,2.531487119e-006,-
9.085139030000001e-009,
  4.29017993e-012},
  blow={8857.282570000001,23.91592727},
  ahight={-25927395.9,75398.1891,-74.840868,0.0369286679,-8.08249724e-006,
  8.17185072999999e-010,-3.0892859e-014},
  bhigh={-471504.572,586.093375},
  R=31.279650307794);

constant IdealGases.Common.DataRecord Cs2Br2(
  name="Cs2Br2",
  MM=0.4256189,
  Hf=-1329426.08751632,
  H0=51955.23507062304,
  Tlimit=1000,
  allow={-6321.28518,-12.13153529,10.04967112,-0.0001102100044,
  1.365293456e-007,-8.85695707e-011,2.339447704e-014},
  blow={-70997.7170999999,-7.53942479},

```

---

**1048 Modelica.Media.IdealGases.Common.SingleGasesData**

---

```
ahigh={-7545.84665,-0.2087093009,10.00018042,-8.03426073e-008,
       1.923604258e-011,-2.34469814e-015,1.139365834e-019},
bhigh={-71058.8354,-7.25179304},
R=19.53501594971464);

constant IdealGases.Common.DataRecord Cs2CO3 (
  name="Cs2CO3",
  MM=0.3258198,
  Hf=-2475134.841406201,
  H0=65002.37554623753,
  Tlimit=1000,
  allow={-46731.3,943.6549060000001,-0.2374670436,0.0416848461,-5.09437221e-
005,
         3.132078394e-008,-7.78042167e-012},
  blow={-103916.2411,40.55853980000001},
  ahigh={-303714.9985,-1517.0784,17.11941656,-0.000444970012,
         9.798325240000001e-008,-1.121256063e-011,5.189734780000001e-016},
  bhigh={-94204.1477,-56.8200626},
  R=25.51862102917011);

constant IdealGases.Common.DataRecord Cs2Cl2 (
  name="Cs2Cl2",
  MM=0.3367169,
  Hf=-1914541.260031796,
  H0=62202.73470087186,
  Tlimit=1000,
  allow={-10779.56357,-53.4610695,10.21710683,-0.000478968703,5.90894653e-007,
         -3.82141652e-010,1.006995985e-013},
  blow={-80295.1517,-12.07856486},
  ahigh={-16251.43966,-0.900613735,10.00077173,-3.41619408e-007,
         8.14496808e-011,-9.897475279999999e-015,4.79833081e-019},
  bhigh={-80565.13219999999,-10.81977337},
  R=24.69276712870664);

constant IdealGases.Common.DataRecord Cs2F2 (
  name="Cs2F2",
  MM=0.3038077064,
  Hf=-2935601.985111462,
  H0=63818.65762968019,
  Tlimit=1000,
  allow={-9942.67748,-253.6723355,11.00684073,-0.002185640152,2.664791874e-006,
         -1.708229209e-009,4.47115198e-013},
  blow={-109058.2601,-20.00699602},
  ahigh={-36779.4642,-4.35575348,10.00366687,-1.603814186e-006,
         3.79146974e-010,-4.57868743e-014,2.209385787e-018},
  bhigh={-110345.6448,-14.15086452},
  R=27.36754804057861);

constant IdealGases.Common.DataRecord Cs2I2 (
  name="Cs2I2",
  MM=0.5196198399999999,
  Hf=-873779.3710879093,
  H0=43464.2815024153,
  Tlimit=1000,
  allow={-4449.28697,-5.32179366,10.02183919,-4.85337485e-005,
         6.019338399999999e-008,-3.90822639e-011,1.032981014e-014},
  blow={-57578.2338,-5.27126709},
```

```

ahigh={-4983.37601,-0.0946117987999999,10.00008229,-3.68096204e-008,
       8.844660910000001e-012,-1.081178562e-015,5.26614198e-020},
bhigh={-57605.01040000001,-5.14486489},
R=16.0010672417743);

constant IdealGases.Common.DataRecord Cs2O (
  name="Cs2O",
  MM=0.2818103,
  Hf=-506920.240317689,
  H0=49997.65799901566,
  Tlimit=1000,
  alow={19105.33804,-496.218153,8.7596063,-0.00351162279,4.01647983e-006,-
2.450899853e-009,
       6.17240489e-013},
  blow={-16776.64956,-11.59258822},
  ahigh={-41191.4741,-10.79255391,7.00850047,-3.54295578e-006,8.08169912e-010,
       -9.499419930000001e-014,4.48912254e-018},
  bhigh={-19343.7785,-1.20397045},
  R=29.503790315684);

constant IdealGases.Common.DataRecord Cs2Oplus (
  name="Cs2Oplus",
  MM=0.2818097514,
  Hf=1006707.271805216,
  H0=51267.65815670069,
  Tlimit=1000,
  alow={683.705609,-243.5575007,7.78303747,-0.001414552725,1.470799739e-006,-
8.218821330000001e-010,
       1.911749489e-013},
  blow={33241.9487,-4.75042356},
  ahigh={-31530.85844,-5.77155903,7.00448263,-1.848241331e-006,4.1808823e-010,
       -4.88237627e-014,2.295419936e-018},
  bhigh={31962.2731,-0.0543435069},
  R=29.50384775081279);

constant IdealGases.Common.DataRecord Cs2O2 (
  name="Cs2O2",
  MM=0.2978097,
  Hf=-829620.7410302619,
  H0=58515.54197193711,
  Tlimit=1000,
  alow={22745.0518,-566.7912570000001,10.01728332,0.002547081016,-4.90921669e-
006,
       3.84878891e-009,-1.124566008e-012},
  blow={-29473.3436,-18.41275291},
  ahigh={-119003.2094,-92.6107526,10.06915777,-2.766227147e-005,
       6.11075296e-009,-7.003595480000001e-013,3.24374252e-017},
  bhigh={-32556.1074,-17.24868948},
  R=27.91874139761062);

constant IdealGases.Common.DataRecord Cs2O2H2 (
  name="Cs2O2H2",
  MM=0.29982558,
  Hf=-2177932.916864532,
  H0=79837.59424396011,
  Tlimit=1000,
  alow={-10859.81041,-688.306842,16.43543354,-0.00403922058,2.331117156e-006,
       1.124566008e-012});

```

---

**1050 Modelica.Media.IdealGases.Common.SingleGasesData**

---

```
 8.43965163e-010,-6.85930973e-013},  
blow={-79394.7623,-49.0619681},  
ahigh={1802857.941,-4656.35475,16.93566597,0.0002117402535,-1.272249096e-  
007,  
 2.066555839e-011,-1.152390667e-015},  
bhigh={-51908.1369,-58.4971916},  
R=27.73102948721054);  
  
constant IdealGases.Common.DataRecord Cs2SO4 (  
  name="Cs2SO4",  
  MM=0.3618735,  
  Hf=-3088515.279510658,  
  H0=66288.81639578473,  
  Tlimit=1000,  
  allow={61536.3787,-638.137433,6.89477209,0.0396018291,-5.529862010000001e-  
005,  
  3.73633252e-008,-9.973241829999999e-012},  
  blow={-133976.031,-1.284432588},  
  ahigh={-522339.752,-944.415525,19.70476453,-0.000282122599,  
  6.240792279999999e-008,-7.16346407e-012,3.32266719e-016},  
  bhigh={-136417.9983,-68.20406795},  
  R=22.97618366639171);  
  
constant IdealGases.Common.DataRecord Cu (  
  name="Cu",  
  MM=0.06354600000000001,  
  Hf=5309539.54615554,  
  H0=97526.64211752116,  
  Tlimit=1000,  
  allow={77.1313315,-1.169236206,2.506987803,-2.116434879e-005,3.44171471e-008,  
  -2.862608999e-011,9.559250991000001e-015},  
  blow={39839.8121,5.73081322},  
  ahigh={2308090.411,-8503.261,14.67859102,-0.00846713652,2.887821016e-006,-  
4.27065918e-010,  
  2.304265084e-014},  
  bhigh={92075.3562,-78.5470156},  
  R=130.8417839045731);  
  
constant IdealGases.Common.DataRecord Cuplus (  
  name="Cuplus",  
  MM=0.0635454513999999,  
  Hf=17138594.56508637,  
  H0=97527.48408362082,  
  Tlimit=1000,  
  allow={-0.002452340093,2.606893531e-005,2.49999989,2.351922485e-010,-  
2.669362382e-013,  
  1.510315123e-016,-3.278224814e-020},  
  blow={130240.0621,5.07594077},  
  ahigh={-2181443.016,7217.85819,-6.94115475,0.00620824892,-2.139340497e-006,  
  3.56643144e-010,-2.081198501e-014},  
  bhigh={85164.56659999999,71.1680067000001},  
  R=130.842913486645);  
  
constant IdealGases.Common.DataRecord Cuminus (  
  name="Cuminus",  
  MM=0.0635465486,  
  Hf=3347445.812344244,
```

```

H0=97525.80016595896,
Tlimit=1000,
alow={0,0,2.5,0,0,0,0},
blow={24838.64954,5.07596603},
ahigh={0,0,2.5,0,0,0,0},
bhigh={24838.64954,5.07596603},
R=130.8406543420047);

constant IdealGases.Common.DataRecord CuCL(
  name="CuCL",
  MM=0.09899899999999999,
  Hf=920110.3041444863,
  H0=95669.40070101718,
  Tlimit=1000,
  alow={14698.47736,-339.316725,5.72345959,-0.002417296132,2.858019864e-006,-
1.757900548e-009,
        4.455047290000001e-013},
  blow={11317.14015,-4.527352594},
  ahigh={-25771.2241,-7.10634606,4.50562318,5.39454827e-005,5.38641796e-010,-
6.348482770000001e-014,
        3.006622062e-018},
  bhigh={9566.192870000001,2.680067564},
  R=83.98541399408076);

constant IdealGases.Common.DataRecord CuF(
  name="CuF",
  MM=0.08254440320000001,
  Hf=-152039.3813932136,
  H0=110038.847552053,
  Tlimit=1000,
  alow={37613.8541,-546.8104900000001,5.83235172,-0.001695312068,
        1.21815088e-006,-3.71328528e-010,2.002276589e-014},
  blow={58.6514072,-7.15693009},
  ahigh={509415.483,-1415.00987,5.63234938,-0.000162912841,-1.156611499e-007,
        5.06603408e-011,-4.15320511e-015},
  bhigh={6305.506600000001,-7.40777361},
  R=100.7272652981032);

constant IdealGases.Common.DataRecord CuF2(
  name="CuF2",
  MM=0.1015428064,
  Hf=-2628842.056506328,
  H0=118727.3665897026,
  Tlimit=1000,
  alow={65733.1128,-896.243084,7.91756094,0.001345667904,-4.16240001e-006,
        3.71618109e-009,-1.155537597e-012},
  blow={-29168.87325,-15.87076153},
  ahigh={-1650355.082,3775.36212,3.88911655,0.000682191264,1.875979026e-007,-
5.67736811e-011,
        3.82324873e-015},
  bhigh={-59533.717,15.22659664},
  R=81.88144778318831);

constant IdealGases.Common.DataRecord CuO(
  name="CuO",
  MM=0.0795454,
  Hf=3850254.068745647,

```

---

**1052 Modelica.Media.IdealGases.Common.SingleGasesData**

---

```
H0=122583.317199989,
Tlimit=1000,
alow={4689.76224,-118.4808464,4.56615524,0.000430905802,-8.309886399999999e-
007,
       6.94663802e-010,-2.108693366e-013},
blow={36151.9068,1.733847344},
ahigh={358228.017,-913.936640999999,5.13689867,6.240577240000001e-005,-
1.558613495e-007,
       5.23680312e-011,-4.02674561999999e-015},
bhigh={41507.9286,-2.63407096},
R=104.5248625313343);

constant IdealGases.Common.DataRecord Cu2 (
  name="Cu2",
  MM=0.127092,
  Hf=3818808.422245302,
  H0=78133.58826676737,
  Tlimit=1000,
  alow={-852.918348,-97.2004493,4.8822337,-0.00073713694,1.000575401e-006,-
6.39198943999999e-010,
       1.668655797e-013},
  blow={57493.0702,1.105391325},
  ahigh={-86993.995,320.910387,3.97380288,0.000508080967,-1.707470385e-007,
       3.21910819e-011,-1.958830868e-015},
  bhigh={55060.9019,6.91450217},
  R=65.42089195228652);

constant IdealGases.Common.DataRecord Cu3CL3 (
  name="Cu3CL3",
  MM=0.296997,
  Hf=-870614.854695502,
  H0=96713.73784920386,
  Tlimit=1000,
  alow={4487.88532,-873.664443999999,19.36189091,-0.00713744786,
       8.56045487e-006,-5.419978740000001e-009,1.40517535e-012},
  blow={-31626.8996,-59.76797945000001},
  ahigh={-91885.5967999999,-15.7193643,16.0129417,-5.57299237e-006,
       1.302748507e-009,-1.560175322e-013,7.48082757e-018},
  bhigh={-36087.6074,-40.13211395},
  R=27.99513799802692);

constant IdealGases.Common.DataRecord D (
  name="D",
  MM=0.002014102,
  Hf=110083912.3341321,
  H0=3077017.946459514,
  Tlimit=1000,
  alow={0,0,2.5,0,0,0,0},
  blow={25921.287,0.591714338},
  ahigh={60.50019210000001,-0.1810766064,2.500210817,-1.220711706e-007,
       3.71517217e-011,-5.66068021e-015,3.393920393e-019},
  bhigh={25922.43752,0.590212537},
  R=4128.128565484767);

constant IdealGases.Common.DataRecord Dplus (
  name="Dplus",
  MM=0.0020135534,
```

```

Hf=764978136.6612874,
H0=3077856.291270944,
Tlimit=1000,
allow={0,0,2.5,0,0,0,0},
blow={184512.0037,-0.1018414521},
ahigh={0,0,2.5,0,0,0,0},
bhigh={184512.0037,-0.1018414521},
R=4129.253289234842);

constant IdealGases.Common.DataRecord Dminus(
  name="Dminus",
  MM=0.0020146506,
  Hf=70857312.92562591,
  H0=3076180.058219525,
  Tlimit=1000,
  allow={0,0,2.5,0,0,0,0},
  blow={16423.73393,-0.1010243437},
  ahigh={0,0,2.5,0,0,0,0},
  bhigh={16423.73393,-0.1010243437},
  R=4127.004454271128);

constant IdealGases.Common.DataRecord DBr(
  name="DBr",
  MM=0.08191810200000001,
  Hf=-452116.1391166021,
  H0=105814.2680112388,
  Tlimit=1000,
  allow={-19182.82458,202.0175399,3.044629038,-0.00148815745,6.8708978e-006,-
6.59116028e-009,
         2.093769216e-012},
  blow={-6560.0744,8.00869003999999},
  ahigh={665400.044,-2594.092228,6.8858794,-0.001103284901,2.894201105e-007,-
3.152037514e-011,
         1.011011776e-015},
  bhigh={10378.1314,-19.73703653},
  R=101.4973711182908);

constant IdealGases.Common.DataRecord DCL(
  name="DCL",
  MM=0.037467102,
  Hf=-2496777.119297884,
  H0=231165.5702648153,
  Tlimit=1000,
  allow={10464.21033,-231.3813446,5.42069326,-0.007508012650000001,
         1.398672495e-005,-1.066249569e-008,3.011167635e-012},
  blow={-11284.03374,-6.711842349},
  ahigh={411728.489999999,-1764.535217,5.6688771,-0.000347596417,
         5.88268803e-008,3.76025313e-013,-5.16460056e-016},
  bhigh={-1541.258092,-12.75659404},
  R=221.9139339893436);

constant IdealGases.Common.DataRecord DF(
  name="DF",
  MM=0.0210125052,
  Hf=-13145866.87169505,
  H0=411093.4854164842,
  Tlimit=1000,

```

## 1054 Modelica.Media.IdealGases.Common.SingleGasesData

---

```
alow={57213.7075,-731.17338,7.22087087,-0.00942396935,1.208025139e-005,-  
6.94181269e-009,  
    1.538525476e-012},  
blow={-30692.30024,-19.32760992},  
ahigh={800117.2800000001,-2438.386832,5.62066445,-0.0002020416838,  
    1.714418979e-008,2.697462563e-012,-2.88829741e-016},  
bhigh={-18574.47029,-14.73004444},  
R=395.6916093945809);  
  
constant IdealGases.Common.DataRecord DO1 (  
    name="DO1",  
    MM=0.053466502,  
    Hf=-1487635.865910959,  
    H0=193108.0884999733,  
    Tlimit=1000,  
    alow={68528.3429,-767.344455,5.93397014,0.002868820667,-5.207166370000001e-  
006,  
        4.927572739999999e-009,-1.709526276e-012},  
    blow={-6824.03505,-7.757833983},  
    ahigh={604306.633,-2646.500381,8.61247421,-0.000553226823999999,  
        1.086845232e-007,-1.13738129e-011,4.90429552e-016},  
    bhigh={4845.79518,-25.88861801},  
    R=155.5080599811822);  
  
constant IdealGases.Common.DataRecord DO2 (  
    name="DO2",  
    MM=0.034012902,  
    Hf=190731.8581637051,  
    H0=295927.2337303062,  
    Tlimit=1000,  
    alow={-21114.79735,602.337175,-1.294877674,0.0181294797,-2.161807666e-005,  
        1.391729127e-008,-3.69774028e-012},  
    blow={-2976.943856,32.7283448},  
    ahigh={-1267224.927,2799.947016,2.325174609,0.00272632507,-6.31450732e-007,  
        6.7292892e-011,-2.765192818e-015},  
    bhigh={-19594.11733,17.89922833},  
    R=244.4505323303493);  
  
constant IdealGases.Common.DataRecord DO2minus (  
    name="DO2minus",  
    MM=0.0340134506,  
    Hf=-3081002.019830355,  
    H0=296350.11509241,  
    Tlimit=1000,  
    alow={104870.5051,-890.623676999999,5.66655356,0.001904603784,-1.21526321e-  
006,  
        6.57751333e-010,-2.147376147e-013},  
    blow={-8942.37725,-7.7972192},  
    ahigh={552766.192,-2796.895783,8.77220666999999,-0.000629111663000001,  
        1.272331622e-007,-1.364440687e-011,6.00519944e-016},  
    bhigh={2512.066767,-28.90020381},  
    R=244.4465896088767);  
  
constant IdealGases.Common.DataRecord D2 (  
    name="D2",  
    MM=0.004028204,  
    Hf=0,
```

```

H0=2127276.324634999,
Tlimit=1000,
alow={21257.90482,-299.6945907,5.13031498,-0.004172970890000001,
      5.01434571999999e-006,-2.126389969e-009,2.386536969e-013},
blow={394.49859,-11.64191209},
ahigh={821516.856,-2365.623159,5.34297451,6.92814599e-005,-8.52367102e-008,
       2.456447415e-011,-1.960597698e-015},
bhigh={14342.14587,-17.12600356},
R=2064.064282742384);

constant IdealGases.Common.DataRecord D2plus(
  name="D2plus",
  MM=0.004027655400000001,
  Hf=372069647.517511,
  H0=2147899.743359374,
  Tlimit=1000,
  alow={-96409.59090000001,1243.052385,-2.557714366,0.01343064234,-
1.285600289e-005,
       6.46342167e-009,-1.337616868e-012},
  blow={173096.6171,33.5631492},
  ahigh={925595.135,-4505.21994,11.03203365,-0.00470608903,1.83806846e-006,-
3.135924623e-010,
       1.857684975e-014},
  bhigh={205829.889,-52.8391224},
  R=2064.345425380731);

constant IdealGases.Common.DataRecord D2minus(
  name="D2minus",
  MM=0.004028752599999999,
  Hf=58370578.77418436,
  H0=2162915.389741232,
  Tlimit=1000,
  alow={-4365.33206,311.2987057,0.548195003,0.009956988109999999,-
1.167000612e-005,
       6.97565641e-009,-1.682718179e-012},
  blow={25978.97625,14.42223161},
  ahigh={-57988.05190000001,-312.2959355,4.73038828,5.6959008e-005,
       2.01897543e-008,-2.311492448e-012,1.070266527e-016},
  bhigh={28512.00896,-9.15752792},
  R=2063.783216671086);

constant IdealGases.Common.DataRecord D2O(
  name="D2O",
  MM=0.020027604,
  Hf=-12443325.72183872,
  H0=497314.4565870186,
  Tlimit=1000,
  alow={6958.27847,-12.80889437,3.59587887,0.001502093683,3.59467505e-007,
       5.3404172e-010,-5.18194127e-013},
  blow={-31019.44566,2.895556576},
  ahigh={1544193.253,-5474.238899999999,10.17542424,-0.0009619415540000001,
       2.036545675e-007,-2.050566442e-011,8.510770689999999e-016},
  bhigh={2983.24898,-44.6501157},
  R=415.1506091292798);

constant IdealGases.Common.DataRecord D2O2(
  name="D2O2",

```

---

**1056 Modelica.Media.IdealGases.Common.SingleGasesData**

---

```
MM=0.036027004,
Hf=-4005328.891628069,
H0=321026.9441222478,
Tlimit=1000,
alow={29577.11324,-68.9303757,2.043905473,0.01570281822,-1.935478714e-005,
      1.384941336e-008,-4.1041849e-012},
blow={-18025.02679,13.47156482},
ahigh={1147867.936,-5225.76093,13.11088701,-0.001179811896,2.729336904e-007,
       -2.961433535e-011,1.310306129e-015},
bhigh={12195.80532,-56.905382},
R=230.7844415816536);

constant IdealGases.Common.DataRecord D2S (
  name="D2S",
  MM=0.036093204,
  Hf=-665126.5983479882,
  H0=279513.9217898196,
  Tlimit=1000,
  alow={3988.38648,-66.2079256,4.39335445,-0.002325113084,1.189503465e-005,-
1.146524521e-008,
      3.62619045e-012},
  blow={-3787.38555,0.9238754374999999},
  ahigh={423581.463,-2823.776231,8.96282184,-0.000646923123,1.634615644e-007,
       -1.797991376e-011,8.16093957e-016},
  bhigh={12046.03509,-31.91047376},
  R=230.3611505368158);

constant IdealGases.Common.DataRecord eminus (
  name="eminus",
  MM=5.48579903e-007,
  Hf=0,
  H0=11297220270.20738,
  Tlimit=1000,
  alow={0,0,2.5,0,0,0,0},
  blow={-745.375,-11.72081224},
  ahigh={0,0,2.5,0,0,0,0},
  bhigh={-745.375,-11.72081224},
  R=15156355.44527048);

constant IdealGases.Common.DataRecord F (
  name="F",
  MM=0.0189984032,
  Hf=4178245.885422623,
  H0=343105.677428722,
  Tlimit=1000,
  alow={1137.409088,-145.3392797,4.07740361,-0.004303360139999999,
      5.72889774e-006,-3.8193129e-009,1.018322509e-012},
  blow={9311.110120000001,-3.55898265},
  ahigh={14735.06226,81.4992736,2.444371819,2.120210026e-005,-4.54691862e-009,
       5.10952873e-013,-2.333894647e-017},
  bhigh={8388.37465,5.47871064},
  R=437.6405697085111);

constant IdealGases.Common.DataRecord Fplus (
  name="Fplus",
  MM=0.0189978546,
  Hf=93000834.52581009,
```

```

H0=353236.3070091083,
Tlimit=1000,
alow={-38716.8019,321.881566,2.200920452,-0.0002455492688,7.85835506e-007,-
6.43598792e-010,
1.839793564e-013},
blow={209883.0937,7.81699924},
ahigh={16496.35664,133.7351478,2.332522942,0.0001215277877,-4.8010377e-008,
9.02722514999999e-012,-5.47066494e-016},
bhigh={211074.5327,6.62581709},
R=437.653207431117);

constant IdealGases.Common.DataRecord Fminus(
  name="Fminus",
  MM=0.0189989518,
  Hf=-13426639.25280341,
  H0=326198.4169042421,
  Tlimit=1000,
  alow={0,0,2.5,0,0,0,0},
  blow={-31425.72443,3.26488271},
  ahigh={0,0,2.5,0,0,0,0},
  bhigh={-31425.72443,3.26488271},
  R=437.6279327157407);

constant IdealGases.Common.DataRecord FCN(
  name="FCN",
  MM=0.0450158032,
  Hf=762573.3977795602,
  H0=225008.9808460865,
  Tlimit=1000,
  alow={39844.5412,-698.865536,7.05883966,-0.001404863382,3.96465251e-006,-
3.045609294e-009,
7.9230459e-013},
  blow={6172.627570000001,-15.05643174},
  ahigh={398187.355,-2302.971079,9.02708138999999,-0.000522871026,
1.191059272e-007,-1.280287231e-011,5.73380719e-016},
  bhigh={15884.49451,-29.84971982},
  R=184.7011806733685);

constant IdealGases.Common.DataRecord FCO(
  name="FCO",
  MM=0.0470085032,
  Hf=-3816716.929629871,
  H0=220980.3395739688,
  Tlimit=1000,
  alow={11326.62744,-53.979185,2.966927601,0.00755995935,-6.21177358e-006,
2.40378155e-009,-3.36950775e-013},
  blow={-22403.69579,10.92652142},
  ahigh={-60858.5158,-1022.397533,7.52732256,-0.0001057942328,-1.365311093e-
009,
2.484612871e-012,-9.9979168e-017},
  bhigh={-18050.98416,-16.30331278},
  R=176.8716601042511);

constant IdealGases.Common.DataRecord FO(
  name="FO",
  MM=0.0349978032,
  Hf=3114826.932908749,

```

---

**1058 Modelica.Media.IdealGases.Common.SingleGasesData**

---

```
H0=268245.4080432112,
Tlimit=1000,
alow={-39121.8244,796.703694,-1.63477767,0.01601810071,-2.095210771e-005,
      1.382740666e-008,-3.66452495e-012},
blow={8375.525229999999,33.83325720000001},
ahigh={-1597940.503,4377.376380000001,-0.489750764,0.002682336321,-
6.90080485e-007,
      7.24647968e-011,-2.726912632e-015},
bhigh={-16442.48211,35.5992361},
R=237.5712541866057);
```

```
constant IdealGases.Common.DataRecord FO2_FOO(
  name="FO2_FOO",
  MM=0.0509972032,
  Hf=498066.529264099,
  H0=220714.2998775274,
  Tlimit=1000,
  alow={5821.564,-234.7363967,5.43733876,0.002165855252,3.67147219e-007,-
2.071530827e-009,
      9.43106685e-013},
  blow={2694.856027,-1.168202057},
  ahigh={-1213166.895,2493.397189,4.46506574,0.0009416104040000001,-
6.42647225999999e-008,
      -1.085643277e-011,1.216995394e-015},
  bhigh={-15968.00286,8.65519318},
  R=163.0378036103753);
```

```
constant IdealGases.Common.DataRecord FO2_OFO(
  name="FO2_OFO",
  MM=0.0509972032,
  Hf=7423936.534621569,
  H0=206646.2734960336,
  Tlimit=1000,
  alow={-77341.064,1663.800209,-9.61281516,0.04989156870000001,-6.94621713e-
005,
      4.7077878e-008,-1.258890729e-011},
  blow={36970.9273,77.9594304},
  ahigh={-621186.7340000001,936.947459,6.40155875,0.0002136110846,-
4.33797765e-008,
      4.66731224e-012,-2.059767432e-016},
  bhigh={36383.5727,-6.15026433},
  R=163.0378036103753);
```

```
constant IdealGases.Common.DataRecord F2 (
  name="F2",
  MM=0.0379968064,
  Hf=0,
  H0=232259.1511269747,
  Tlimit=1000,
  alow={10181.76308,22.74241183,1.97135304,0.008151604010000001,-1.14896009e-
005,
      7.95865253e-009,-2.167079526e-012},
  blow={-958.6943,11.30600296},
  ahigh={-2941167.79,9456.5977,-7.73861615,0.00764471299,-2.241007605e-006,
      2.915845236e-010,-1.425033974e-014},
  bhigh={-60710.0561,84.23835080000001},
  R=218.8202848542556);
```

```

constant IdealGases.Common.DataRecord F2O (
  name="F2O",
  MM=0.0539962064,
  Hf=453735.5794684124,
  H0=202079.4001557858,
  Tlimit=1000,
  allow={30829.19995,-229.9506259,2.603805825,0.01586111264,-2.345633734e-005,
    1.679619314e-008,-4.70356033e-012},
  blow={3055.185332,10.50933022},
  ahighe={-188537.4518,-210.0729689,7.15123916,0.0001327687906,
    1.804705706e-008,-1.416973671e-012,6.489389390000001e-017},
  bhigh={1449.129965,-12.57858336},
  R=153.9825212609751);

constant IdealGases.Common.DataRecord F2O2 (
  name="F2O2",
  MM=0.06999560639999999,
  Hf=274302.9311051158,
  H0=196847.3981246915,
  Tlimit=1000,
  allow={56122.70460000001,-998.595223,8.98064939,0.00642190087,-9.88851602e-
006,
    6.85806101e-009,-1.842137493e-012},
  blow={5298.71259,-22.39292195},
  ahighe={-219056.3532,-391.092081,10.29107144,-0.0001163345202,
    2.570922521e-008,-2.949199983e-012,1.367386916e-016},
  bhigh={827.8020939999999,-27.56752555},
  R=118.7856270933028);

constant IdealGases.Common.DataRecord FS2F (
  name="FS2F",
  MM=0.1021268064,
  Hf=-3294286.895472725,
  H0=142913.5063994325,
  Tlimit=1000,
  allow={114446.228,-1853.247564,13.94098518,-0.00404368937,1.370970316e-006,
    6.7119031e-010,-4.624954960000001e-013},
  blow={-33510.6833,-48.49215357},
  ahighe={-207709.7921,-109.4539831,10.08136946,-3.23953791e-005,
    7.12522527e-009,-8.13478741e-013,3.75502241e-017},
  bhigh={-43494.4606,-23.46916058},
  R=81.41321845936034);

constant IdealGases.Common.DataRecord Fe (
  name="Fe",
  MM=0.055845,
  Hf=7439717.07404423,
  H0=122666.7920136091,
  Tlimit=1000,
  allow={67908.2266,-1197.218407,9.84339331,-0.01652324828,1.917939959e-005,-
    1.149825371e-008,
    2.832773807e-012},
  blow={54669.9594,-33.8394626},
  ahighe={-1954923.682,6737.16109999999,-5.48641097,0.004378803450000001,-
    1.116286672e-006,
    1.544348856e-010,-8.023578182e-015},
  bhigh={7137.37006,65.0497986},
  R=148.8848061599069);

```

---

**1060 Modelica.Media.IdealGases.Common.SingleGasesData**

---

```
constant IdealGases.Common.DataRecord Feplus(
  name="Feplus",
  MM=0.0558444514,
  Hf=21205649.25094779,
  H0=124197.6387290645,
  Tlimit=1000,
  allow={-56912.3162,184.713439,4.19697212,-0.00597827597,1.054267912e-005,-
8.05980431999999e-009,
  2.256925874e-012},
  blow={140120.6571,-0.360254258},
  ahighelement{140120.6571,-0.360254258}=-817645.009,1925.359408,1.717387154,0.000338533898,-
9.813533120000001e-008,
  2.228179208e-011,-1.483964439e-015},
  bhigh={128635.2466,15.00256262},
  R=148.8862687618774);

constant IdealGases.Common.DataRecord Fe_CO_5 (
  name="Fe_CO_5",
  MM=0.1958955,
  Hf=-3715501.377009681,
  H0=169199.3894704064,
  Tlimit=1000,
  allow={379780.571,-7285.92816,54.0023618,-0.06935400750000001,
  0.0001026705717,-7.20737313e-008,1.958981996e-011},
  blow={-58545.8048,-260.4377836},
  ahighelement{1116600.852,-8067.07473,36.5294241,-0.0020471903,4.44196775e-007,-
4.93250713e-011,
  2.235704865e-015},
  bhigh={-48606.6125,-175.4566113},
  R=42.44340477448436);

constant IdealGases.Common.DataRecord FeCL (
  name="FeCL",
  MM=0.091298,
  Hf=2749676.882297531,
  H0=113662.435102631,
  Tlimit=1000,
  allow={11173.40353,-54.0214429,3.48605792,0.00687983714,-1.273679557e-005,
  1.025321859e-008,-3.051544011e-012},
  blow={29286.8199,9.428962979},
  ahighelement{29286.8199,9.428962979}=-528870.022,-1282.897413,5.79844169,-0.0002896589776,3.34390381e-008,-
1.469606582e-013,-1.213444602e-016},
  bhigh={37261.9673,-4.075134191},
  R=91.06959626716905);

constant IdealGases.Common.DataRecord FeCL2 (
  name="FeCL2",
  MM=0.126751,
  Hf=-1112423.570622717,
  H0=112640.8312360455,
  Tlimit=1000,
  allow={23011.21607,-585.804406,9.322903480000001,-0.003006298824,
  2.590788666e-006,-1.080178662e-009,2.341239608e-013},
  blow={-16210.41276,-18.16752393},
  ahighelement{164412.3697,692.800269,4.63827014,0.002754339782,-8.624218250000001e-
007,
  1.170827576e-010,-5.93806195e-015},
  bhigh={-22197.21855,11.16897701},
```

```

R=65.59689469905563);

constant IdealGases.Common.DataRecord FeCL3 (
  name="FeCL3",
  MM=0.162204,
  Hf=-6529458.509038001,
  H0=112291.1457177382,
  Tlimit=1000,
  allow={4284.772099999999,-574.170537,12.20667256,-0.00468043412,
    5.609500470000001e-006,-3.5495995e-009,9.198543490000001e-013},
  blow={-127568.8456,-28.87584081},
  ahight={-59157.40590000001,-10.33382228,10.00849967,-3.65765187e-006,
    8.545873639999999e-010,-1.023070527e-013,4.90406364e-018},
  bhigh={-130501.205,-15.98494171},
  R=51.25935242040887);

constant IdealGases.Common.DataRecord FeO (
  name="FeO",
  MM=0.0718443999999999,
  Hf=3494218.060141083,
  H0=123002.363440992,
  Tlimit=1000,
  allow={15692.82213,-64.6018888,2.45892547,0.00701604736,-1.021405947e-005,
    7.179297870000001e-009,-1.978966365e-012},
  blow={29645.72665,13.26115545},
  ahight={-119597.148,-362.486478,5.51888075,-0.000997885688999999,
    4.37691383e-007,-6.79062946e-011,3.63929268e-015},
  bhigh={30379.85806,-3.63365542},
  R=115.7288807478384);

constant IdealGases.Common.DataRecord Fe_OH_2 (
  name="Fe_OH_2",
  MM=0.08985968,
  Hf=-3678357.189787456,
  H0=158129.4413690323,
  Tlimit=1000,
  allow={444302.72,-6795.14089,38.9472621,-0.0597300568,7.046165430000001e-005,
    -4.087859510000001e-008,9.368766340000001e-012},
  blow={-9051.42086,-193.1304058},
  ahight={1612519.19,-6533.24199,18.42922816,-0.002073249635,4.26587436e-007,-
    4.56406313e-011,
    1.990105746e-015},
  bhigh={-2992.568633,-84.45940589999999},
  R=92.52728253650581);

constant IdealGases.Common.DataRecord Fe2CL4 (
  name="Fe2CL4",
  MM=0.253502,
  Hf=-1701644.957436233,
  H0=117747.3984426158,
  Tlimit=1000,
  allow={1501.308814,-661.709651,18.26924882,-0.00418149466,
    4.196535049999999e-006,-2.188670747e-009,5.395194700000001e-013},
  blow={-53400.5758,-49.3564068},
  ahight={140245.0973,693.606441,13.1380041,0.00275433307,-8.623996010000001e-
    007,
    1.170782491e-010,-5.93777808e-015},

```

---

**1062 Modelica.Media.IdealGases.Common.SingleGasesData**

---

```
bhigh={-59741.72730000001,-17.52491511},  
R=32.79844734952781);  
  
constant IdealGases.Common.DataRecord Fe2CL6(  
  name="Fe2CL6",  
  MM=0.324408,  
  Hf=-2017143.843555029,  
  H0=124682.0485314789,  
  Tlimit=1000,  
  alow={-10345.44447,-823.48766,25.20090594,-0.00684579679,8.25582075e-006,-  
5.24896566e-009,  
  1.365242237e-012},  
  blow={-81318.4719,-80.10768938},  
  ahight={-99991.1799999999,-14.57716085,22.0120846,-5.22948113e-006,  
  1.226831608e-009,-1.473195806e-013,7.07827233e-018},  
  bhigh={-85514.9619,-61.43757198},  
  R=25.62967621020443);  
  
constant IdealGases.Common.DataRecord Ga(  
  name="Ga",  
  MM=0.0697229999999999,  
  Hf=3901151.700299758,  
  H0=93959.16698937224,  
  Tlimit=1000,  
  alow={238794.789,-3121.634631,16.30171272,-0.02347922342,1.932327565e-005,-  
7.31631187e-009,  
  8.857387735e-013},  
  blow={47327.1738,-75.4717275999999},  
  ahight={-55441.8652,880.862441999999,1.760717716,0.000299325946,-  
5.61082731e-008,  
  2.682832239e-012,3.132134914e-016},  
  bhigh={26843.51822,12.52212023},  
  R=119.2500609554953);  
  
constant IdealGases.Common.DataRecord Gaplus(  
  name="Gaplus",  
  MM=0.0697224514,  
  Hf=12287124.31645799,  
  H0=88887.12137278639,  
  Tlimit=1000,  
  alow={0.000409834494,-4.34358162e-006,2.500000019,-4.389036810000001e-011,  
  5.5639692e-014,-3.63822826e-017,9.60788199499999e-021},  
  blow={102289.9726,5.21509046},  
  ahight={35666.0714,-110.8845546,2.635517951,-8.33897352e-005,  
  2.734243614e-008,-4.55567374e-012,3.035050101e-016},  
  bhigh={102989.743,4.25707541},  
  R=119.2509992555999);  
  
constant IdealGases.Common.DataRecord GaBr(  
  name="GaBr",  
  MM=0.149627,  
  Hf=-120084.4767321406,  
  H0=66379.18958476746,  
  Tlimit=1000,  
  alow={-2185.275415,-76.7223296999999,4.77080567,-0.000449788106,  
  5.5798536e-007,-3.065156457e-010,7.042946040000001e-014},  
  blow={-3138.004818,2.948144294},
```

```

ahigh={335980.503,-840.683668,5.14508473,-3.265526e-005,-7.67102275e-008,
       3.63512382e-011,-3.35817783e-015},
bhigh={2035.179445,-0.427789408},
R=55.56799240778737);

constant IdealGases.Common.DataRecord GaBr2 (
  name="GaBr2",
  MM=0.229531,
  Hf=-649938.1434316061,
  H0=61102.63101716108,
  Tlimit=1000,
  allow={7081.68162,-370.57661,8.418425360000001,-0.003001930915,
         3.59547767e-006,-2.276883735e-009,5.91322177e-013},
  blow={-18211.16655,-9.512572049999999},
  ahight={-330229.436,476.672265,7.1047254,-0.0005522771210000001,
           3.159876015e-007,-5.51363715e-011,3.17329323e-015},
  bhigh={-23656.8362,-1.162714892},
  R=36.22374319808653);

constant IdealGases.Common.DataRecord GaBr3 (
  name="GaBr3",
  MM=0.309435,
  Hf=-946766.4065151,
  H0=61522.89495370594,
  Tlimit=1000,
  allow={6363.80734,-514.204045,11.96065138,-0.00413463765,
         4.933978330000001e-006,-3.111868526e-009,8.04363148e-013},
  blow={-35704.155,-25.56896053},
  ahight={-51030.6375,-9.39520969,10.00768765,-3.2959543e-006,
           7.679838140000001e-010,-9.175147259999999e-014,4.39118694e-018},
  bhigh={-38334.0215,-14.10315914},
  R=26.86984988769855);

constant IdealGases.Common.DataRecord GaCL (
  name="GaCL",
  MM=0.105176,
  Hf=-661951.7760705864,
  H0=91362.24994295275,
  Tlimit=1000,
  allow={6417.143,-225.7949147,5.31167369,-0.001562280975,1.836579731e-006,-
1.100421873e-009,
         2.725812445e-013},
  blow={-8593.892329999999,-1.695911735},
  ahight={-486484.224,1452.870183,2.719542959,0.001144285628,-3.43145286e-007,
           5.47044048e-011,-3.019622103e-015},
  bhigh={-18950.84035,15.70926145},
  R=79.05293983418271);

constant IdealGases.Common.DataRecord GaCL2 (
  name="GaCL2",
  MM=0.140629,
  Hf=-1571360.878623897,
  H0=96092.49159135029,
  Tlimit=1000,
  allow={22254.81851,-604.621844,9.220388910000001,-0.00455440079,
         5.32485518e-006,-3.30860557e-009,8.462235559999999e-013},
  blow={-25645.50386,-16.85409262},

```

---

**1064 Modelica.Media.IdealGases.Common.SingleGasesData**

---

```
ahigh={-344540.53, 471.443716, 7.10877548, -0.000553943356, 3.16363824e-007, -  
5.51802393e-011,  
3.17535303e-015},  
bhigh={-32309.3616, -3.74484074},  
R=59.12345248846255);  
  
constant IdealGases.Common.DataRecord GaCL3 (  
name="GaCL3",  
MM=0.176082,  
Hf=-2456950.47193921,  
H0=98677.01979759432,  
Tlimit=1000,  
alow={47849.484, -1162.678125, 14.08954213, -0.008106134290000001,  
9.21943016e-006, -5.59971055e-009, 1.4048526e-012},  
blow={-49159.0973, -42.82179383},  
ahigh={-94669.1731, -25.61468254, 10.02010048, -8.35407307e-006,  
1.901439055e-009, -2.231168493e-013, 1.052949609e-017},  
bhigh={-55182.0143, -18.64958383},  
R=47.21931827216866);  
  
constant IdealGases.Common.DataRecord GaF (  
name="GaF",  
MM=0.08872140319999999,  
Hf=-2621785.528748265,  
H0=102355.3356063241,  
Tlimit=1000,  
alow={36703.9801, -533.6572219999999, 5.75929192, -0.001506128492,  
9.509038450000001e-007, -1.83094113e-010, -3.135959193e-014},  
blow={-26470.7569, -6.58932188},  
ahigh={-298339.2279, 722.5157870000001, 3.63955743, 0.000554673676, -  
1.46490145e-007,  
2.177385396e-011, -1.031058874e-015},  
bhigh={-34085.2871, 7.55396},  
R=93.71438796179906);  
  
constant IdealGases.Common.DataRecord GaF2 (  
name="GaF2",  
MM=0.1077198064,  
Hf=-4796819.250503221,  
H0=112473.8282114105,  
Tlimit=1000,  
alow={68425.3953, -889.761774, 7.70745549, 0.00207215089, -5.28998193e-006,  
4.56412742e-009, -1.406942753e-012},  
blow={-59198.5615, -13.41142116},  
ahigh={410025.1359999999, -1723.259472, 8.95132922, -0.001059205682,  
2.745755154e-007, -2.907681311e-011, 1.084834142e-015},  
bhigh={-53765.716, -21.13835528},  
R=77.18610233224483);  
  
constant IdealGases.Common.DataRecord GaF3 (  
name="GaF3",  
MM=0.1267182096,  
Hf=-7271859.237190485,  
H0=120132.8289600455,  
Tlimit=1000,  
alow={95932.66699999999, -1355.507494, 10.97760197, 0.00346157172, -  
8.46955194999999e-006,
```

```

    7.22246893e-009,-2.213328653e-012},
blow={-106147.9087,-32.1265185},
ahigh={-209696.176,-143.1211274,10.10568378,-4.18493682e-005,
9.16475945e-009,-1.042661957e-012,4.79916345e-017},
bhigh={-113667.8134,-23.9253933},
R=65.61386896362842);

constant IdealGases.Common.DataRecord GaH(
  name="GaH",
  MM=0.07073094000000001,
  Hf=3030122.376431021,
  H0=122621.0764341602,
  Tlimit=1000,
  allow={-43918.762,625.445712,0.327686289,0.0064966415,-3.80297001e-006,
2.852103704e-010,3.62680913e-013},
blow={21712.60108,22.24032244},
ahigh={3257993.99,-11090.99502,18.01126796,-0.008167719110000001,
2.601293566e-006,-3.81345935e-010,2.028049649e-014},
bhigh={93697.7824000001,-98.2090022},
R=117.5507069466347);

constant IdealGases.Common.DataRecord GaI(
  name="GaI",
  MM=0.19662747,
  Hf=228203.7296212986,
  H0=51412.56203927151,
  Tlimit=1000,
  allow={-3661.85503,-32.0692418,4.59996192,-8.899475459999999e-005,
1.162792633e-007,-1.812351457e-011,-7.584619639999999e-015},
blow={4198.64937,4.92896465},
ahigh={1498356.654,-4551.96724,9.80300534999999,-0.002943735609,
8.658040509999999e-007,-1.106638204e-010,5.00783686e-015},
bhigh={32918.1191,-32.2735977},
R=42.28540396720764);

constant IdealGases.Common.DataRecord GaI2(
  name="GaI2",
  MM=0.32353194,
  Hf=-89496.46517125944,
  H0=44727.99501650439,
  Tlimit=1000,
  allow={-983.13959,-215.2621259,7.84431876,-0.001819797849,2.210706958e-006,-
1.416110855e-009,
3.71376293e-013},
blow={-4534.090230000001,-4.25051681},
ahigh={-320287.975,479.650233,7.10233396,-0.000551265783,3.157543452e-007,-
5.51087194e-011,
3.17197773e-015},
bhigh={-9180.919610000001,0.727520488},
R=25.69907626430948);

constant IdealGases.Common.DataRecord GaI3(
  name="GaI3",
  MM=0.45043641,
  Hf=-257255.4914022159,
  H0=44673.91967714155,
  Tlimit=1000,

```

## 1066 Modelica.Media.IdealGases.Common.SingleGasesData

---

```
alow={-5663.47265,-242.5937116,10.95451119,-0.002059175489,2.499117622e-006,
      -1.596502506e-009,4.16764406e-013},
blow={-15767.22832,-16.46129942},
ahigh={-31640.4013,-4.20866283,10.00352001,-1.53265018e-006,3.61153154e-010,
      -4.35097887e-014,2.095704875e-018},
bhigh={-17000.58453,-10.90291552},
R=18.45870319408682);

constant IdealGases.Common.DataRecord GaO (
  name="GaO",
  MM=0.08572239999999999,
  Hf=1712779.938499156,
  H0=104116.3919815591,
  Tlimit=1000,
  alow={73255.4391,-980.4269509999999,7.84412276,-0.00717107285,
        7.87278051e-006,-2.401464112e-009,-2.594919458e-013},
  blow={21405.81724,-17.99742951},
  ahigh={2937313.804,-11183.14112,18.59366773,-0.00685475345,1.665399251e-006,
        -1.887560878e-010,7.97268692e-015},
  bhigh={85302.48080000001,-99.04040480000001},
  R=96.9929913301541);

constant IdealGases.Common.DataRecord GaOH (
  name="GaOH",
  MM=0.08673034,
  Hf=-1656058.249051024,
  H0=122028.9001518961,
  Tlimit=1000,
  alow={69631.0187,-1247.828695,10.22001611,-0.00592253439,4.6433874e-006,-
1.215936978e-009,
        -2.945243614e-014},
  blow={-12753.97794,-32.17063},
  ahigh={842905.101,-2372.324595,8.009071090000001,8.77723071e-005,-
5.94306891e-008,
        9.841943439999999e-012,-5.53155279e-016},
  bhigh={-3751.78825,-21.71053719},
  R=95.86578353088434);

constant IdealGases.Common.DataRecord Ga2Br2 (
  name="Ga2Br2",
  MM=0.299254,
  Hf=-457683.7469173344,
  H0=69483.80974022068,
  Tlimit=1000,
  alow={-11171.83241,-74.7147162,10.30246084,-0.0006657929850000001,
        8.20051585e-007,-5.29702152e-010,1.394553192e-013},
  blow={-19132.97856,-13.17110316},
  ahigh={-18853.73131,-1.263621802,10.00108034,-4.774975660000001e-007,
        1.137227648e-010,-1.380820486e-014,6.69024243e-019},
  bhigh={-19510.53843,-11.41667594},
  R=27.78399620389368);

constant IdealGases.Common.DataRecord Ga2Br4 (
  name="Ga2Br4",
  MM=0.459062,
  Hf=-905803.647437601,
  H0=66639.72622434443,
```

```

Tlimit=1000,
alow={-13155.45939,-413.655302,17.62849229,-0.00351465216,4.26693773e-006,-
2.726515882e-009,
    7.11892449e-013},
blow={-52831.0181,-44.2752069},
ahigh={-57421.6721,-7.16013737,16.00598947,-2.60818227e-006,6.14645595e-010,
    -7.40543865e-014,3.56712663e-018},
bhigh={-54933.8976,-34.79271660000001},
R=18.11187159904327);

constant IdealGases.Common.DataRecord Ga2Br6(
  name="Ga2Br6",
  MM=0.61887,
  Hf=-1088578.75159565,
  H0=66494.20072066832,
  Tlimit=1000,
  alow={-6725.11689,-806.752655,25.11671424,-0.00663651457,7.9775557e-006,-
5.05968521e-009,
    1.313546388e-012},
  blow={-83706.352,-75.8527872},
  ahigh={-95261.73,-14.4468188,22.01192546,-5.14500674e-006,1.204348449e-009,
    -1.443806473e-013,6.928270740000001e-018},
  bhigh={-87822.23300000001,-57.6589512},
  R=13.43492494384927);

constant IdealGases.Common.DataRecord Ga2CL2(
  name="Ga2CL2",
  MM=0.210352,
  Hf=-1050490.064273218,
  H0=92646.73024264089,
  Tlimit=1000,
  alow={-9246.1337,-259.2159934,11.02847129,-0.002231983927,2.720723456e-006,
    -1.743798624e-009,4.56366604e-013},
  blow={-28340.59821,-20.67773272},
  ahigh={-36689.542,-4.43458068,10.00373051,-1.630793333e-006,3.85375332e-010,
    -4.65252773e-014,2.244498791e-018},
  bhigh={-29656.33085,-14.69535817},
  R=39.52646991709135);

constant IdealGases.Common.DataRecord Ga2CL4(
  name="Ga2CL4",
  MM=0.281258,
  Hf=-2141545.584481152,
  H0=97118.63129226546,
  Tlimit=1000,
  alow={16373.43702,-1140.637354,20.30139827,-0.0089977462,1.067254867e-005,-
6.70029061999999e-009,
    1.725749144e-012},
  blow={-71623.85010000001,-66.663619},
  ahigh={-112711.0409,-21.38517542,16.01738077,-7.41556464e-006,
    1.721694751e-009,-2.051361953e-013,9.79731149999999e-018},
  bhigh={-77468.53380000001,-41.4728087},
  R=29.56172624423128);

constant IdealGases.Common.DataRecord Ga2CL6(
  name="Ga2CL6",
  MM=0.352164,

```

---

**1068 Modelica.Media.IdealGases.Common.SingleGasesData**

---

```
Hf=-2732998.069081451,
H0=103246.7799093604,
Tlimit=1000,
alow={60894.6983999999,-2104.456302,29.59653878,-0.01537809796,
      1.779096808e-005,-1.095619429e-008,2.779706493e-012},
blow={-111840.0535,-112.1945814},
ahigh={-189844.5426,-44.1365524,22.03505047,-1.470071128e-005,
      3.36951281e-009,-3.9754628e-013,1.884231868e-017},
bhigh={-122696.9998,-67.45332883},
R=23.60965913608433);

constant IdealGases.Common.DataRecord Ga2F2 (
  name="Ga2F2",
  MM=0.1774428064,
  Hf=-3416488.705850406,
  H0=96638.77813871186,
  Tlimit=1000,
  alow={29781.3065,-1013.97552,13.74933053,-0.00772805096,9.06400733e-006,-
5.64119322e-009,
      1.443106716e-012},
  blow={-70860.8337,-41.09833949999999},
  ahigh={-87732.34929999999,-19.8377587,10.0159403,-6.744402830000001e-006,
      1.556129063e-009,-1.845316975e-013,8.780884020000001e-018},
  bhigh={-76073.6863,-19.08325521},
  R=46.85719398089953);

constant IdealGases.Common.DataRecord Ga2F4 (
  name="Ga2F4",
  MM=0.2154396128,
  Hf=-6150229.016750257,
  H0=105148.9171632971,
  Tlimit=1000,
  alow={142207.9998,-2798.829941,23.95904235,-0.01293904774,1.223101326e-005,
      -6.25120982e-009,1.334919195e-012},
  blow={-149601.371,-97.34153640000001},
  ahigh={-270436.609,-104.9957196,16.07903965,-3.17889592e-005,
      7.04958272e-009,-8.10237952e-013,3.76054053e-017},
  bhigh={-164416.9589,-48.9659173},
  R=38.59305116612241);

constant IdealGases.Common.DataRecord Ga2F6 (
  name="Ga2F6",
  MM=0.2534364192,
  Hf=-7961065.143552976,
  H0=118891.6576990526,
  Tlimit=1000,
  alow={240904.5572,-4142.3798,31.9577372,-0.0129791429,8.95928219e-006,-
2.733621068e-009,
      1.554030994e-013},
  blow={-227278.524,-139.8403732},
  ahigh={-436866.48,-212.8208368,22.15886373,-6.34628791e-005,
      1.399738556e-008,-1.601732039e-012,7.407584420000001e-017},
  bhigh={-249429.8707,-77.87635830000001},
  R=32.80693448181421);

constant IdealGases.Common.DataRecord Ga2I2 (
  name="Ga2I2",
```

```

MM=0.39325494,
Hf=34382.14151867996,
H0=54666.69280746988,
Tlimit=1000,
alow={-8960.25267,-31.7205721,10.12930491,-0.0002860191165,3.53533301e-007,
      -2.289632389e-010,6.04008536e-014},
blow={-1233.174068,-9.80196838},
ahigh={-12190.56501,-0.528755447,10.00045392,-2.011599344e-007,
       4.79946781e-011,-5.83478108e-015,2.829565069e-019},
bhigh={-1393.258606,-9.052614910000001},
R=21.14270198360382);

```

```

constant IdealGases.Common.DataRecord Ga2I4(
  name="Ga2I4",
  MM=0.64706388,
  Hf=-246139.5109861487,
  H0=50356.40839664856,
  Tlimit=1000,
  alow={-17694.95991,-135.4871509,16.54625333,-0.001199014713,
        1.473754848e-006,-9.504796569999999e-010,2.499380837e-013},
  blow={-23334.05795,-33.2571289},
  ahigh={-31712.9976,-2.287068836,16.00194812,-8.58896438e-007,
         2.041966602e-010,-2.476145769e-014,1.198551944e-018},
  bhigh={-24019.40505,-30.08672521},
  R=12.84953813215474);

```

```

constant IdealGases.Common.DataRecord Ga2I6(
  name="Ga2I6",
  MM=0.90087282,
  Hf=-352208.0097832234,
  H0=48367.95719955232,
  Tlimit=1000,
  alow={-21108.91458,-369.540416,23.46811836,-0.00318917821,3.89033904e-006,-
2.494830378e-009,
        6.53200728e-013},
  blow={-43011.8809,-60.148115},
  ahigh={-60156.9815,-6.32920923,22.0053304,-2.332008629e-006,5.51386848e-010,
        -6.659445350000001e-014,3.21367043e-018},
  bhigh={-44886.9938,-51.6100875},
  R=9.22935159704341);

```

```

constant IdealGases.Common.DataRecord Ga2O(
  name="Ga2O",
  MM=0.1554454,
  Hf=-639822.4521278854,
  H0=78243.51830288963,
  Tlimit=1000,
  alow={56971.9718,-807.4085990000001,7.73306154,0.001478450874,-
4.097743430000001e-006,
        3.58115892e-009,-1.107073202e-012},
  blow={-9512.216820000002,-12.56149299},
  ahigh={-119387.4986,-87.19341610000001,7.0646782,-2.571524542e-005,
        5.65110849e-009,-6.44819054e-013,2.975374645e-017},
  bhigh={-13936.94796,-6.9428013},
  R=53.4880543264709);

```

```
constant IdealGases.Common.DataRecord Ge (
```

---

**1070 Modelica.Media.IdealGases.Common.SingleGasesData**

---

```
name="Ge",
MM=0.07264,
Hf=5063325.991189428,
H0=101852.7533039648,
Tlimit=1000,
alow={-20592.15242,-143.2022103,4.50600233,0.00154718784,-
8.518296550000001e-006,
     8.24382446e-009,-2.566167305e-012},
blow={43630.7086,-6.224648889},
ahigh={-856541.384,3917.95866,-1.809888212,0.002276482224,-5.36562755e-007,
      5.984958090000001e-011,-2.541700646e-015},
bhigh={19565.18798,38.41408752},
R=114.4613436123348);

constant IdealGases.Common.DataRecord Geplus(
name="Geplus",
MM=0.0726394514,
Hf=15624903.03994779,
H0=85432.70743919742,
Tlimit=1000,
alow={-345366.643,4008.44971,-15.22395737,0.0362603002,-3.35136726e-005,
      1.431059219e-008,-2.236976459e-012},
blow={115705.8042,109.0158182},
ahigh={-2244860.57,5165.53114,-0.386751651,0.000852873094,-1.386040003e-007,
      1.155945775e-011,-3.784413134e-016},
bhigh={100682.3985,29.65845411},
R=114.4622080667311);

constant IdealGases.Common.DataRecord Geminus(
name="Geminus",
MM=0.07264054860000001,
Hf=3378313.417638478,
H0=96102.52310236543,
Tlimit=1000,
alow={2989.142308,86.5741723,2.191868507,0.0005900795269999999,-6.37580058e-
007,
      3.66113215e-010,-8.689778450000001e-014},
blow={28356.9519,9.417623637},
ahigh={13520.40246,-0.504851878,2.500140247,-3.65294416e-009,-6.7167222e-
012,
      1.378119925e-015,-8.475212940000001e-020},
bhigh={28817.39757,7.577953867},
R=114.4604791709957);

constant IdealGases.Common.DataRecord GeBr(
name="GeBr",
MM=0.152544,
Hf=900973.6141703377,
H0=64664.09036081393,
Tlimit=1000,
alow={-56807.7463,917.336126,-2.224809366,0.02171239242,-3.035490868e-005,
      2.011776305e-008,-5.20291504e-012},
blow={11041.92901,41.07865983},
ahigh={225675.0476,-731.071546,6.26328381,-0.001198085447,4.00037777e-007,-
5.42993788e-011,
      2.510481953e-015},
bhigh={19428.63606,-6.106507844},
R=54.50540172015943);
```

```

constant IdealGases.Common.DataRecord GeBr2 (
  name="GeBr2",
  MM=0.232448,
  Hf=-262264.6656456497,
  H0=61060.80929928413,
  Tlimit=1000,
  allow={-1649.285325,-229.6262116,7.90014468,-0.001936685504,2.345776175e-006,
         -1.496279368e-009,3.9014535e-013},
  blow={-8316.62421,-6.918156136},
  ahighe={-26366.55245,-3.99050902,7.00332744,-1.445749787e-006,
          3.40159136e-010,-4.09345703e-014,1.969987842e-018},
  bhigh={-9484.999249999999,-1.673617098},
  R=35.76916987885463);

constant IdealGases.Common.DataRecord GeBr3 (
  name="GeBr3",
  MM=0.312352,
  Hf=-381080.665403135,
  H0=59386.49984632723,
  Tlimit=1000,
  allow={2725.568677,-518.621326,11.99957754,-0.00425102453,5.10362972e-006,-
3.23370625e-009,
         8.38835886e-013},
  blow={-14779.88251,-25.34555678},
  ahighe={-54339.267,-9.28032795,10.00765019,-3.29732215e-006,7.71292926e-010,
          -9.24157185e-014,4.43287623e-018},
  bhigh={-17426.97092,-13.66946296},
  R=26.61891711914763);

constant IdealGases.Common.DataRecord GeBr4 (
  name="GeBr4",
  MM=0.392256,
  Hf=-741862.4571708272,
  H0=61090.76215532714,
  Tlimit=1000,
  allow={5431.940390000001,-677.774880000001,15.60015305,-0.00550778888,
         6.59476429e-006,-4.17014614e-009,1.080116643e-012},
  blow={-35576.064,-42.09122927},
  ahighe={-69777.5791,-11.79873115,13.00953955,-4.02493875e-006,
          9.203567209999999e-010,-1.076853369e-013,5.0409548e-018},
  bhigh={-39041.4816,-26.8943607},
  R=21.19654511339533);

constant IdealGases.Common.DataRecord GeCL (
  name="GeCL",
  MM=0.108093,
  Hf=638615.960330456,
  H0=88804.2241403236,
  Tlimit=1000,
  allow={10697.24739,-22.41238266,1.98737107,0.01368270905,-2.358294414e-005,
         1.78675602e-008,-5.10250325e-012},
  blow={7440.68339,15.05837189},
  ahighe={-378419.325,1429.309204,3.19485464,0.0008119939119999999,-
2.652459954e-007,
         5.15195788e-011,-3.53421457e-015},
  bhigh={-2087.046335,13.8503382},
  R=76.91961551626839);

```

---

**1072 Modelica.Media.IdealGases.Common.SingleGasesData**

---

```
constant IdealGases.Common.DataRecord GeCL2 (
  name="GeCL2",
  MM=0.143546,
  Hf=-1191255.764702604,
  H0=92491.54278071142,
  Tlimit=1000,
  allow={18385.36673,-582.720534,9.161404660000001,-0.00446478603,
         5.24487355e-006,-3.26806721e-009,8.36757542e-013},
  blow={-19757.98598,-17.3432163},
  ahigh={-48901.28920000001,-11.26210418,7.00906353,-3.83918073e-006,
         8.86563357e-010,-1.05200054e-013,5.00842222e-018},
  bhigh={-22752.54424,-4.656641298},
  R=57.9220040962479);

constant IdealGases.Common.DataRecord GeCL3 (
  name="GeCL3",
  MM=0.178999,
  Hf=-1491352.404203376,
  H0=98097.96144112539,
  Tlimit=1000,
  allow={19358.59579,-834.202,13.1270253,-0.006511390020000001,7.69627748e-006,
         -4.81854173e-009,1.238399959e-012},
  blow={-30972.18753,-35.52547183999999},
  ahigh={-75744.9982,-15.7921342,10.01278835,-5.44173831e-006,
         1.260936724e-009,-1.500137106e-013,7.15640081e-018},
  bhigh={-35251.3172,-17.19719944},
  R=46.44982374203208);

constant IdealGases.Common.DataRecord GeCL4 (
  name="GeCL4",
  MM=0.214452,
  Hf=-2331524.070654505,
  H0=98561.09991979557,
  Tlimit=1000,
  allow={65028.7041,-1652.48276,18.81327,-0.0115242681,1.31080998e-005,-
7.96195611e-009,
         1.99750504e-012},
  blow={-55700.5239,-67.54438135},
  ahigh={-137626.121,-36.037695,13.0281667,-1.16495696e-005,2.63619338e-009,-
3.07292672e-013,
         1.43980875e-017},
  bhigh={-64262.8178,-33.18158305},
  R=38.77078320556581);

constant IdealGases.Common.DataRecord GeF(
  name="GeF",
  MM=0.0916384032,
  Hf=-770342.8642894556,
  H0=99784.69376035572,
  Tlimit=1000,
  allow={50170.9782,-439.555252,2.990864513,0.01252766738,-2.337593226e-005,
         1.841631179e-008,-5.398492260000001e-012},
  blow={-7093.5164,7.064248931},
  ahigh={-515048.058,1659.711777,2.999610299,0.000819851167999999,-
2.271982948e-007,
         3.62418386e-011,-2.133840473e-015},
  bhigh={-20515.29487,13.7315423},
  R=90.73130597718665);
```

```

constant IdealGases.Common.DataRecord GeF2 (
  name="GeF2",
  MM=0.1106368064,
  Hf=-5188146.862489335,
  H0=106542.1660616552,
  Tlimit=1000,
  allow={75430.9708,-1108.456469,9.07889827,-0.001445630636,-6.57532232e-007,
         1.475344103e-009,-5.81122027e-013},
  blow={-65106.8555,-22.00457134},
  ahighe={-127532.7716,-71.6259135,7.05300237,-2.101910441e-005,4.6078937e-009,
          -5.24639467e-013,2.416218951e-017},
  bhigh={-71126.3763,-8.483841439000001},
  R=75.15104846699552);

constant IdealGases.Common.DataRecord GeF3 (
  name="GeF3",
  MM=0.1296352096,
  Hf=-6220014.111042869,
  H0=113226.8543807716,
  Tlimit=1000,
  allow={111412.1569,-1776.90803,13.52166477,-0.002983622949,-3.61274975e-008,
         1.621528436e-009,-7.19901178e-013},
  blow={-90382.62960000002,-45.68760023},
  ahighe={-206958.0129,-111.0233803,10.08234122,-3.27167626e-005,
          7.1838967e-009,-8.19038812e-013,3.77632244e-017},
  bhigh={-99997.4909,-23.0346594},
  R=64.13745174366579);

constant IdealGases.Common.DataRecord GeF4 (
  name="GeF4",
  MM=0.1486336128,
  Hf=-8007273.574123873,
  H0=116346.6639492194,
  Tlimit=1000,
  allow={111981.8968,-1636.85963,12.31371005,0.01025446325,-1.893783072e-005,
         1.476849261e-008,-4.32517411e-012},
  blow={-137426.5128,-41.04202391000001},
  ahighe={-325303.268,-253.8279448,13.18792173,-7.46005855e-005,
          1.637384721e-008,-1.866482733e-012,8.605530500000001e-017},
  bhigh={-146617.6526,-41.22237561},
  R=55.93937900969867);

constant IdealGases.Common.DataRecord GeH4 (
  name="GeH4",
  MM=0.07667176000000001,
  Hf=1184177.85114102,
  H0=0,
  Tlimit=1000,
  allow={-256839.8191,-41.3281145,7.75511628,0.002129640783,
         7.25505361000001e-007,-5.09631401e-010,1.431608261e-013},
  blow={7881.53804,-20.30107999},
  ahighe={-3809805.88,10931.67093,-5.14967541,0.00948593136999999,-
          1.649459859e-006,
          1.419674126e-010,-5.00463975e-015},
  bhigh={-61585.1852,71.68961272999999},
  R=108.4424304333173);

```

---

**1074 Modelica.Media.IdealGases.Common.SingleGasesData**

---

```
constant IdealGases.Common.DataRecord GeI (
  name="GeI",
  MM=0.19954447,
  Hf=1057253.653784542,
  H0=50139.80091755988,
  Tlimit=1000,
  allow={-74383.08219999999,1033.759595,-1.520491112,0.01555125152,-
1.683415215e-005,
  8.383557809999999e-009,-1.550875391e-012},
  blow={19129.20413,39.59395473},
  ahighe={-213314.8437,-295.3079008,6.87660736,-0.002102579132,8.59629938e-007,
  -1.44769276e-010,8.401042789999999e-015},
  bhigh={24523.26136,-8.549783749},
  R=41.66726344257999);
constant IdealGases.Common.DataRecord GeO (
  name="GeO",
  MM=0.08863939999999999,
  Hf=-425254.5369215045,
  H0=99076.77624171645,
  Tlimit=1000,
  allow={-6781.22563,279.1946972,0.619895286,0.01111378013,-1.501027092e-005,
  1.006783003e-008,-2.687101932e-012},
  blow={-6711.84743,21.56441027},
  ahighe={-1044508.485,2734.866048,1.298071298,0.001832638634,-5.06098635e-007,
  6.37495802e-011,-2.172371872e-015},
  bhigh={-23621.04658,23.50906051},
  R=93.80108619868818);
constant IdealGases.Common.DataRecord GeO2 (
  name="GeO2",
  MM=0.1046388,
  Hf=-1014652.289590477,
  H0=107589.1734232426,
  Tlimit=1000,
  allow={-3824.17383,156.3451633,1.694198285,0.01756205059,-2.418270032e-005,
  1.632388154e-008,-4.36895786e-012},
  blow={-14775.3488,15.56926515},
  ahighe={-172734.7526,-296.1154443,7.72072625,-8.82493339e-005,
  1.949874702e-008,-2.235834829e-012,1.036128507e-016},
  bhigh={-13879.83288,-16.69845695},
  R=79.45878584234529);
constant IdealGases.Common.DataRecord GeS (
  name="GeS",
  MM=0.104705,
  Hf=883674.2275918056,
  H0=87303.47165846905,
  Tlimit=1000,
  allow={36610.1843,-565.9653500000001,6.15293641,-0.002746908039,
  2.733435196e-006,-1.460209187e-009,3.27870947e-013},
  blow={12741.75754,-7.70565946},
  ahighe={-2351367.441,7211.58186,-4.36252631,0.00545173775,-1.721002774e-006,
  2.632908084e-010,-1.39437404e-014},
  bhigh={-35833.1725,65.05566890999999},
  R=79.40854782484124);
```

```

constant IdealGases.Common.DataRecord GeS2 (
  name="GeS2",
  MM=0.13677,
  Hf=868741.9097755356,
  H0=95546.03348687578,
  Tlimit=1000,
  allow={55436.836,-1024.826697,10.57594552,-0.00528788391,5.29580308e-006,-
2.87252953e-009,
  6.51902755e-013},
  blow={17355.79668,-29.91719536},
  ahighe={-89709.8036,-33.64377,7.52543372,-1.026368137e-005,2.282230718e-009,
  -2.62874877e-013,1.222218614e-017},
  bhigh={11952.32103,-11.35170616},
  R=60.79163559260072);

constant IdealGases.Common.DataRecord Ge2 (
  name="Ge2",
  MM=0.14528,
  Hf=3245449.676486785,
  H0=73624.23595814979,
  Tlimit=1000,
  allow={216197.8929,-3079.621733,18.88047728,-0.02604825861,2.27915135e-005,-
9.02835127e-009,
  1.146806387e-012},
  blow={70324.0254999999,-79.01095968},
  ahighe={1969461.258,-5248.6871,10.58277303,-0.00333655151,1.052745838e-006,-
1.484534793e-010,
  7.475431760000001e-015},
  bhigh={89256.65990000001,-37.50233948},
  R=57.23067180616741);

constant IdealGases.Common.DataRecord H (
  name="H",
  MM=0.00100794,
  Hf=216281552.4733615,
  H0=6148608.052066591,
  Tlimit=1000,
  allow={0,0,2.5,0,0,0,0},
  blow={25473.70801,-0.446682853},
  ahighe={60.78774250000001,-0.1819354417,2.500211817,-1.226512864e-007,
  3.73287633e-011,-5.68774456e-015,3.410210197e-019},
  bhigh={25474.86398,-0.448191777},
  R=8248.975137408972);

constant IdealGases.Common.DataRecord Hplus (
  name="Hplus",
  MM=0.0010073914,
  Hf=1524974233.450872,
  H0=6151956.429248851,
  Tlimit=1000,
  allow={0,0,2.5,0,0,0,0},
  blow={184021.4877,-1.140646644},
  ahighe={0,0,2.5,0,0,0,0},
  bhigh={184021.4877,-1.140646644},
  R=8253.46732163884);

constant IdealGases.Common.DataRecord Hminus (

```

## 1076 Modelica.Media.IdealGases.Common.SingleGasesData

---

```
name="Hminus",
MM=0.0010084886,
Hf=137861080.4326395,
H0=6145263.317800519,
Tlimit=1000,
allow={0,0,2.5,0,0,0,0},
blow={15976.15494,-1.139013868},
ahigh={0,0,2.5,0,0,0,0},
bhigh={15976.15494,-1.139013868},
R=8244.487840516989);

constant IdealGases.Common.DataRecord HALO(
  name="HALO",
  MM=0.043988878,
  Hf=41399.41918955059,
  H0=225857.476974066,
  Tlimit=1000,
  allow={26686.71568,-466.321459,5.38380021,0.002902425002,-2.24367761e-007,-
1.516294323e-009,
  6.95173766e-013},
  blow={1235.914473,-6.50883752},
  ahigh={78474.67049999999,-1515.62885,8.572776230000001,-
0.0004142611439999999,
  8.9350817e-008,-1.007036446e-011,4.60801725e-016},
  bhigh={6663.24189,-26.72948642},
  R=189.0130500714294);

constant IdealGases.Common.DataRecord HALO2 (
  name="HALO2",
  MM=0.059988278,
  Hf=-5925716.870886008,
  H0=199715.7511339132,
  Tlimit=1000,
  allow={3544.59891,115.9819104,0.863797069,0.02460875785,-3.42316012e-005,
  2.373182987e-008,-6.46760883e-012},
  blow={-44495.06140000001,20.12317065},
  ahigh={698570.544,-2864.935042,10.86721494,-5.3683127e-005,-2.836877559e-
008,
  6.29164119e-012,-3.88903311e-016},
  bhigh={-27643.92495,-37.7588634},
  R=138.6016114681605);

constant IdealGases.Common.DataRecord HBO (
  name="HBO",
  MM=0.02781834,
  Hf=-7571307.454003366,
  H0=329527.139290123,
  Tlimit=1000,
  allow={63609.7503,-800.1557590000001,6.21881613,-0.000780167998,
  3.141286759e-006,-2.031853478e-009,3.73855289e-013},
  blow={-22402.8291,-13.26952393},
  ahigh={886186.0229999999,-3913.06805,9.8886448,-0.000822269139999999,
  1.621530554e-007,-1.703550237e-011,7.372740020000001e-016},
  bhigh={-3161.852029,-40.3681272},
  R=298.8845488264217);

constant IdealGases.Common.DataRecord HBOplus (
```

```

name="HBOplus",
MM=0.0278177914,
Hf=42247053.76861803,
H0=326935.6962681085,
Tlimit=1000,
alow={65244.7147,-704.733736,5.29427187,0.001308494598,2.088343622e-006,-
2.564100339e-009,
8.198981249999999e-013},
blow={143929.3436,-6.81674923},
ahigh={18360.83606,-1212.975446,6.82290196,0.0009142272500000001,-
2.665355457e-007,
3.29704981e-011,-1.530684082e-015},
bhigh={146361.7504,-16.99646159},
R=298.8904431859389);

constant IdealGases.Common.DataRecord HBO2 (
  name="HBO2",
  MM=0.04381774,
  Hf=-12785005.63926848,
  H0=249170.8152907932,
  Tlimit=1000,
  alow={6225.08747,75.6615369,1.253406833,0.01748006535,-1.982688351e-005,
  1.22965646e-008,-3.153609847e-012},
  blow={-68785.8878,17.67793507},
  ahigh={1049369.185,-4479.145479999999,11.97755861,-0.00047357434,
  6.08020714e-008,-3.64156544e-012,6.15597317e-017},
  bhigh={-42211.4947,-49.11366819999999},
  R=189.7512742555869);

constant IdealGases.Common.DataRecord HBS (
  name="HBS",
  MM=0.04388394,
  Hf=1144108.755959469,
  H0=211609.5090823659,
  Tlimit=1000,
  alow={56499.66209999999,-546.640995,3.63645839,0.009963657670000001,-
  1.395090386e-005,
  1.032834167e-008,-3.047186722e-012},
  blow={7919.86053,1.182455314},
  ahigh={-202183.6183,-505.202662,6.41521828,0.001049432932,-3.68314669e-007,
  5.34296831e-011,-2.154140451e-015},
  bhigh={6485.47562,-13.31379864},
  R=189.4650298036138);

constant IdealGases.Common.DataRecord HBSpplus (
  name="HBSpplus",
  MM=0.0438833914,
  Hf=25737731.99762314,
  H0=230975.5394155794,
  Tlimit=1000,
  alow={148950.9163,-1847.661736,11.4745802,-0.009476828369999999,
  1.110746739e-005,-6.175558610000001e-009,1.336853512e-012},
  blow={143782.2395,-41.13512498},
  ahigh={661065.836,-2791.498527,9.131267640000001,-0.000506379529,
  7.751213140000001e-008,-4.19113293e-012,2.687149095e-017},
  bhigh={151073.4114,-30.81426166},
  R=189.4673983652048);

```

---

**1078 Modelica.Media.IdealGases.Common.SingleGasesData**

---

```
constant IdealGases.Common.DataRecord HCN (
  name="HCN",
  MM=0.02702534,
  Hf=4924358.398451231,
  H0=341733.4620026983,
  Tlimit=1000,
  allow={90982.86930000001,-1238.657512,8.72130787,-0.00652824294,
         8.87270083e-006,-4.808886669999999e-009,9.31789849999999e-013},
  blow={20989.1545,-27.46678076},
  ahighelement1={1236889.278,-4446.732410000001,9.73887485,-0.000585518264,
                 1.07279144e-007,-1.013313244e-011,3.34824798e-016},
  bhighelement1={42215.1377,-40.05774072000001},
  R=307.6546678043644);

constant IdealGases.Common.DataRecord HCO (
  name="HCO",
  MM=0.02901804,
  Hf=1461085.931372346,
  H0=344249.6460822303,
  Tlimit=1000,
  allow={-11898.51887,215.1536111,2.730224028,0.001806516108,4.98430057e-006,-
5.81456792e-009,
         1.869689894e-012},
  blow={2905.75564,11.3677254},
  ahighelement1={694960.6120000001,-3656.22338,9.604731170000001,-0.001117129278,
                 2.875328019e-007,-3.62624774e-011,1.808329595e-015},
  bhighelement1={25437.0444,-35.8247372},
  R=286.5276910501192);

constant IdealGases.Common.DataRecord HCOpplus (
  name="HCOpplus",
  MM=0.0290174914,
  Hf=28707995.06810571,
  H0=311742.7304553281,
  Tlimit=1000,
  allow={157344.2506,-1867.692159,10.99235423,-0.01211637888,1.659091514e-005,
         -1.016592642e-008,2.391234771e-012},
  blow={108493.028,-40.7826162},
  ahighelement1={1219060.653,-4714.29489,10.21192493,-0.000885451707,1.667408026e-007,
                 -1.683285548e-011,7.04005178e-016},
  bhighelement1={127798.9027,-43.5115846},
  R=286.5331080963119);

constant IdealGases.Common.DataRecord HCCN (
  name="HCCN",
  MM=0.03903604,
  Hf=15637616.62299762,
  H0=303968.6146443133,
  Tlimit=1000,
  allow={2994.114377,-440.466068,6.94003641,0.00384630496,-1.264728529e-006,-
3.63923513e-010,
         2.748929104e-013},
  blow={73708.7864,-13.15302591},
  ahighelement1={939562.031,-4091.06775,12.72233302,-0.000687440531,1.231169968e-007,
                 -1.186956238e-011,4.76061035e-016},
  bhighelement1={95851.6482,-52.48695794},
  R=212.9947607390504);
```

```

constant IdealGases.Common.DataRecord HCCO (
  name="HCCO",
  MM=0.04102874,
  Hf=4303522.360179718,
  H0=284312.7524754599,
  Tlimit=1000,
  allow={69596.12700000001,-1164.594402,9.456616260000001,-0.002331240632,
         5.1618736e-006,-3.52616997e-009,8.59914323e-013},
  blow={25350.03992,-27.26355351},
  ahigh={1093922.002,-4498.228209999999,12.46446433,-0.00063433174,
         1.108549019e-007,-1.125488678e-011,5.68915194e-016},
  bhigh={46522.803,-50.9907043},
  R=202.6499473296036);

constant IdealGases.Common.DataRecord HCL (
  name="HCL",
  MM=0.03646094,
  Hf=-2531750.415650282,
  H0=236968.7671244899,
  Tlimit=1000,
  allow={20625.88287,-309.3368855,5.27541885,-0.00482887422,6.1957946e-006,-
3.040023782e-009,
         4.916790029999999e-013},
  blow={-10677.82299,-7.309305408},
  ahigh={915774.951,-2770.550211,5.97353979,-0.000362981006,4.73552919e-008,
         2.810262054e-012,-6.65610422e-016},
  bhigh={5674.95805,-16.42825822},
  R=228.0377850927595);

constant IdealGases.Common.DataRecord HD (
  name="HD",
  MM=0.003022042,
  Hf=106981.3060175868,
  H0=2815679.596775955,
  Tlimit=1000,
  allow={25191.20338,-276.1004999,4.64444129,-0.002082376844,1.418070803e-006,
         2.839893835e-010,-3.20233103e-013},
  blow={391.361643,-9.395396119999999},
  ahigh={845583.000000001,-1956.578537,4.40437387,0.000575168109,-
2.131983152e-007,
         4.03612668e-011,-2.727170705e-015},
  bhigh={12272.54163,-10.84742878},
  R=2751.276123892388);

constant IdealGases.Common.DataRecord HDplus (
  name="HDplus",
  MM=0.0030214934,
  Hf=495381726.7977484,
  H0=2850908.097300494,
  Tlimit=1000,
  allow={-80700.73049999999,879.643258,0.0596893216,0.005418181009999999,-
2.021515155e-006,
         -4.94526165e-010,4.09293565e-013},
  blow={174499.2497,19.34281193},
  ahigh={1340083.03,-5730.069570000001,12.13836576,-0.00524333812,
         1.976302344e-006,-3.30657353e-010,1.937902763e-014},
  bhigh={213534.8051,-61.2875839},
  R=2751.775661664527);

```

---

**1080 Modelica.Media.IdealGases.Common.SingleGasesData**

---

```
constant IdealGases.Common.DataRecord HDO (
  name="HDO",
  MM=0.019021442,
  Hf=-12894946.50300435,
  H0=521817.5888032044,
  Tlimit=1000,
  allow={-27377.95356, 431.0503899999999, 1.558479899, 0.005764969149999999, -4.85512936e-006,
         3.38299017e-009, -1.040863879e-012},
  blow={-32732.2645, 14.87751891},
  ahighe={1711376.798, -5322.8723, 9.124351519999999, -0.000340066415,
          4.152343519999999e-008, -4.78076857e-014, -1.46803517e-016},
  bhigh={3245.22468, -37.7460068},
  R=437.1104987729112);

constant IdealGases.Common.DataRecord HDO2 (
  name="HDO2",
  MM=0.035020842,
  Hf=-4004519.223152887,
  H0=323663.4059226788,
  Tlimit=1000,
  allow={-32164.7665, 749.2334460000001, -1.957676212, 0.02412802791, -2.915946684e-005,
         1.930016324e-008, -5.22698631e-012},
  blow={-21510.59175, 36.7240674},
  ahighe={1313180.619, -5175.63712, 12.16490975, -0.000613669948, 1.227560176e-007,
          -1.079624939e-011, 3.87401372e-016},
  bhigh={13032.0967, -50.85276870000001},
  R=237.4149656367485);

constant IdealGases.Common.DataRecord HF (
  name="HF",
  MM=0.0200063432,
  Hf=-13660667.38273289,
  H0=429818.8286603021,
  Tlimit=1000,
  allow={-3192.09897, 59.8680772, 3.055113902, 0.001684673783, -3.28739483e-006,
         3.095923617e-009, -9.76469161e-013},
  blow={-34184.4316, 3.29490412},
  ahighe={725708.904, -1484.797741, 3.85552747, 0.000713898985, -2.106757333e-007,
          3.050092453e-011, -1.639495583e-015},
  bhigh={-23554.5666, -3.20385683},
  R=415.5917909075958);

constant IdealGases.Common.DataRecord HI (
  name="HI",
  MM=0.12791241,
  Hf=206070.7010367485,
  H0=67675.21618895305,
  Tlimit=1000,
  allow={18728.8173, -343.178884, 5.95671243, -0.008543439599999999,
         1.454780274e-005, -1.049104164e-008, 2.839734003e-012},
  blow={3682.95072, -8.14975609},
  ahighe={472492.145, -1923.465741, 5.75804897, -0.000406626638,
          9.474332049999999e-008, -1.033534431e-011, 4.61161479e-016},
  bhigh={13948.57037, -11.82487652},
  R=65.00129268145288);
```

```

constant IdealGases.Common.DataRecord HNC (
  name="HNC",
  MM=0.02702534,
  Hf=7192439.429069164,
  H0=370049.8865139163,
  Tlimit=1000,
  allow={48706.2064,-989.1456249999999,9.72215389,-0.01113593916,
         1.668862707e-005,-1.113940941e-008,2.893600868e-012},
  blow={26646.79758,-31.04826344},
  ahighe={1198791.66,-3918.94186,9.11802009,-0.00034177611,3.31480968e-008,-
5.70157474e-013,
         -7.789455389999999e-017},
  bhigh={46588.103,-34.68575882},
  R=307.6546678043644);

constant IdealGases.Common.DataRecord HNCO (
  name="HNCO",
  MM=0.04302474,
  Hf=-2743921.962108313,
  H0=254879.5413987394,
  Tlimit=1000,
  allow={75424.6008,-955.0937770000001,6.72570587,0.0047056875,-4.95947551e-
006,
         3.69425512e-009,-1.164859121e-012},
  blow={-10681.49742,-13.65584762},
  ahighe={1253216.926,-5021.091539999999,12.47789314,-0.000689165525,
         1.097738448e-007,-9.3064038e-012,3.24260695e-016},
  bhigh={14531.55559,-53.06419819},
  R=193.2486285797427);

constant IdealGases.Common.DataRecord HNO (
  name="HNO",
  MM=0.03101404,
  Hf=3289888.224816889,
  H0=320560.6235111581,
  Tlimit=1000,
  allow={-68547.6486,955.16272,-0.600072021,0.00799517674999999,-6.54707916e-
007,
         -3.6705134e-009,1.783392519e-012},
  blow={6435.35126,30.48166179},
  ahighe={-5795614.98,19454.57427,-21.52568374,0.01797428992,-4.97604067e-006,
         6.397924169999999e-010,-3.142619368e-014},
  bhigh={-110419.2372,181.8650338},
  R=268.0873565649622);

constant IdealGases.Common.DataRecord HNO2 (
  name="HNO2",
  MM=0.04701344,
  Hf=-1668712.648978675,
  H0=246680.2046393542,
  Tlimit=1000,
  allow={8591.985060000001,120.3644046,0.9412979119999999,0.01942891839,-
2.253174194e-005,
         1.384587594e-008,-3.47355046e-012},
  blow={-11063.37202,20.73967331},
  ahighe={878790.4129999999,-390.45503,11.87349269,-0.000488190061,
         7.13363679e-008,-5.37630334e-012,1.581778986e-016},
  bhigh={12463.43241,-46.08874688}),

```

---

**1082 Modelica.Media.IdealGases.Common.SingleGasesData**

---

```
R=176.8530871172158);

constant IdealGases.Common.DataRecord HNO3 (
  name="HNO3",
  MM=0.06301284,
  Hf=-2125167.965766977,
  H0=188475.7138386399,
  Tlimit=1000,
  allow={9202.86901,109.3774496,-0.452104245,0.02984914503,-3.1906355e-005,
    1.720931528e-008,-3.78264983e-012},
  blow={-17640.48507,27.46644879},
  ahigh={-94978.0964,-2733.024468,14.49426995,-0.000782186805,
    1.702693665e-007,-1.930543961e-011,8.870455120000001e-016},
  bhigh={-4882.51778,-59.28392985000001},
  R=131.9488535987269);

constant IdealGases.Common.DataRecord HOCL (
  name="HOCL",
  MM=0.05246034,
  Hf=-1443757.932182674,
  H0=194902.0155035213,
  Tlimit=1000,
  allow={-9739.307430000001,354.756952,0.1539514254,0.01617051795,-
    2.179693631e-005,
    1.509103049e-008,-4.12538351e-012},
  blow={-11763.23791,24.73257759},
  ahigh={853045.781,-2847.760552,7.94832904,-0.0001048782013,-1.482405043e-
    008,
    4.59167827e-012,-3.060073987e-016},
  bhigh={7250.964950000001,-22.4983169},
  R=158.4906235834537);

constant IdealGases.Common.DataRecord HOF (
  name="HOF",
  MM=0.0360057432,
  Hf=-2691187.915821163,
  H0=280189.6059737493,
  Tlimit=1000,
  allow={-36968.883,780.900666,-2.077685317,0.02038690173,-2.586919999e-005,
    1.713797538e-008,-4.55128187e-012},
  blow={-16317.20064,36.450525},
  ahigh={881201.823,-3120.013169,8.22371006999999,-0.0002298036315,
    1.459115709e-008,1.095883303e-012,-1.404445608e-016},
  bhigh={6300.54671,-25.90150427},
  R=230.9207160039958);

constant IdealGases.Common.DataRecord HO2 (
  name="HO2",
  MM=0.03300674,
  Hf=364168.0456779433,
  H0=0,
  Tlimit=1000,
  allow={-75988.8254,1329.383918,-4.67738824,0.02508308202,-3.006551588e-005,
    1.895600056e-008,-4.82856739e-012},
  blow={0,-5873.35096},
  ahigh={10002.162,-1810669.724,4963.19203,-1.039498992,0.004560148530000001,
    -1.061859447e-006,1.144567878e-010},
```

```

bhigh={-4.76306416e-015,0},
R=251.9022478439252);

constant IdealGases.Common.DataRecord HO2minus(
  name="HO2minus",
  MM=0.0330072886,
  Hf=-2966710.813077812,
  H0=0,
  Tlimit=1000,
  allow={110383.9835,-1047.963653,6.36001399,0.002942520461,-
6.28414133999999e-006,
      5.43825424e-009,-1.64730582e-012},
  blow={0,-7417.741590000001},
  ahigh={10245.32,793330.6,-2503.312417,7.54896233,8.30839014999999e-005,-
5.96973091e-008,
      9.955377000000001e-012},
  bhigh={-5.60647728e-016,0},
  R=251.8980610846054);

constant IdealGases.Common.DataRecord HPO (
  name="HPO",
  MM=0.04798110100000001,
  Hf=-1185232.473093937,
  H0=209777.241251717,
  Tlimit=1000,
  allow={-38163.1412,792.764187,-1.889940652,0.01800907425,-1.857631793e-005,
      1.018768867e-008,-2.355583619e-012},
  blow={-11576.4124,36.9292591},
  ahigh={384245.945,-2434.951707,8.58217392000001,-0.000405844857,-
1.66948874e-008,
      2.253640566e-011,-1.943063011e-015},
  bhigh={5761.63212,-26.49097544},
  R=173.2863945743971);

constant IdealGases.Common.DataRecord HSO3F (
  name="HSO3F",
  MM=0.1000695432,
  Hf=-7525966.202272021,
  H0=150037.3991914055,
  Tlimit=1000,
  allow={6896.24213,-93.0581074999999,1.570331788,0.0378082075,-4.87272078e-
005,
      3.160796011e-008,-8.18978292999999e-012},
  blow={-91802.40820000001,17.16315977},
  ahigh={554404.708,-3974.42432,17.78869264,-0.000440932203,5.94441139e-008,-
3.92701581e-012,
      8.891284000000001e-017},
  bhigh={-71534.3827,-76.13127849},
  R=83.08693868405707);

constant IdealGases.Common.DataRecord H2 (
  name="H2",
  MM=0.00201588,
  Hf=0,
  H0=4200697.462150524,
  Tlimit=1000,
  allow={40783.2321,-800.918604,8.21470201,-0.01269714457,1.753605076e-005,-

```

---

**1084 Modelica.Media.IdealGases.Common.SingleGasesData**

---

```
1.20286027e-008,
    3.36809349e-012},
blow={2682.484665,-30.43788844},
ahigh={560812.801,-837.150474,2.975364532,0.001252249124,-3.74071619e-007,
    5.936625200000001e-011,-3.6069941e-015},
bhigh={5339.82441,-2.202774769},
R=4124.487568704486);

constant IdealGases.Common.DataRecord H2plus(
    name="H2plus",
    MM=0.0020153314,
    Hf=741650941.3786734,
    H0=4258904.01945804,
    Tlimit=1000,
    alow={-31208.8606,230.4622909,3.33556442,-0.002419056763,
        7.006022340000001e-006,-5.61001066e-009,1.564169746e-012},
    blow={177410.4638,-0.8278523760000001},
    ahigh={1672225.964,-6595.18499,12.79321925,-0.00550934526,2.030669412e-006,
        -3.35102748e-010,1.946089104e-014},
    bhigh={218999.9548,-67.9271078},
    R=4125.610309053885);

constant IdealGases.Common.DataRecord H2minus(
    name="H2minus",
    MM=0.0020164286,
    Hf=116626178.5812798,
    H0=4275178.402052024,
    Tlimit=1000,
    alow={-97535.65670000001,1221.166236,-2.264588838,0.01237202227,-
1.12710002e-005,
        5.36723995e-009,-1.04942016e-012},
    blow={21213.99948,30.50556136},
    ahigh={95992.7562,-914.468287999999,5.14941881,-0.0001016559478,
        5.446919560000001e-008,-6.15517444999999e-012,2.822451181e-016},
    bhigh={32341.0518,-14.4078098},
    R=4123.365439272186);

constant IdealGases.Common.DataRecord HBOH (
    name="HBOH",
    MM=0.02882628,
    Hf=-1690275.817760738,
    H0=379533.3979965504,
    Tlimit=1000,
    alow={-61596.4419,1223.483035,-5.46037051,0.0355538868,-4.67502014e-005,
        3.23600885e-008,-8.98605332e-012},
    blow={-12636.59051,54.2235229},
    ahigh={1534611.049,-5753.66643,12.70893665,-0.000705962825,1.027679375e-007,
        -7.67092674e-012,2.211854768e-016},
    bhigh={27793.70419,-56.0886008},
    R=288.4337486488024);

constant IdealGases.Common.DataRecord HCOOH (
    name="HCOOH",
    MM=0.04602538,
    Hf=-8225244.419492028,
    H0=237426.589416535,
    Tlimit=1000,
```

```

alow={-29062.79097, 765.837888, -3.32841413, 0.02817542991, -2.370050804e-005,
      1.166063663e-008, -2.79137317e-012},
blow={-50064.4347, 43.8709423},
ahigh={487233.645, -7632.238079999999, 21.32788153, -0.004402546540000001,
       1.102001695e-006, -1.364343517e-010, 6.64842975e-015},
bhigh={-5781.43191, -111.1790688},
R=180.6497197850404);

constant IdealGases.Common.DataRecord H2F2 (
  name="H2F2",
  MM=0.0400126864,
  Hf=-14243576.95713228,
  H0=346625.8141567821,
  Tlimit=1000,
  alow={52592.1471, -991.3544890000001, 10.43577115, -0.002407796033, -
6.376956159999999e-007,
      2.7357849e-009, -1.10434859e-012},
  blow={-65724.60829999999, -30.38432132},
  ahigh={1464995.601, -3335.07492, 9.187487040000001, 0.001051127249, -
3.27860557e-007,
      4.45604623e-011, -2.281370136e-015},
  bhigh={-48254.42090000001, -26.39128168},
  R=207.7958954537979);

constant IdealGases.Common.DataRecord H2O (
  name="H2O",
  MM=0.01801528,
  Hf=-13423382.81725291,
  H0=549760.6476280135,
  Tlimit=1000,
  alow={-39479.6083, 575.573102, 0.931782653, 0.00722271286, -7.34255737e-006,
      4.95504349e-009, -1.336933246e-012},
  blow={-33039.7431, 17.24205775},
  ahigh={1034972.096, -2412.698562, 4.64611078, 0.002291998307, -
6.836830479999999e-007,
      9.426468930000001e-011, -4.82238053e-015},
  bhigh={-13842.86509, -7.97814851},
  R=461.5233290850878);

constant IdealGases.Common.DataRecord H2Oplus (
  name="H2Oplus",
  MM=0.0180147314,
  Hf=54488837.17466918,
  H0=551463.4539596855,
  Tlimit=1000,
  alow={-1753.89272, 224.9850054, 1.989400675, 0.00611789516, -7.09543664e-006,
      5.54765947e-009, -1.704344789e-012},
  blow={115958.5952, 11.35409642},
  ahigh={622871.426, -2864.257487, 7.71756556, -0.000902780167, 6.17743686e-007, -
1.201457479e-010,
      7.407709940000001e-015},
  bhigh={134208.6651, -26.3661792},
  R=461.5373837880259);

constant IdealGases.Common.DataRecord H2O2 (
  name="H2O2",
  MM=0.03401468,

```

---

**1086 Modelica.Media.IdealGases.Common.SingleGasesData**

---

```
Hf=-3994745.79799075,
H0=328059.3849479107,
Tlimit=1000,
alow={-92795.3358,1564.748385,-5.97646014,0.0327074452,-3.93219326e-005,
      2.509255235e-008,-6.46504529e-012},
blow={-24940.04728,58.7717418},
ahigh={1489428.027,-5170.82178,11.2820497,-8.04239779e-005,-1.818383769e-
008,
      6.94726559e-012,-4.8278319e-016},
bhigh={14182.51038,-46.50855660000001},
R=244.4377545224592);

constant IdealGases.Common.DataRecord H2S (
  name="H2S",
  MM=0.03408088,
  Hf=-604444.4861752396,
  H0=292199.6439059085,
  Tlimit=1000,
  alow={9543.80881,-68.7517508,4.05492196,-0.0003014557336,3.76849775e-006,-
2.239358925e-009,
      3.086859108e-013},
  blow={-3278.45728,1.415194691},
  ahigh={1430040.22,-5284.02865,10.16182124,-0.000970384996,2.154003405e-007,
      -2.1696957e-011,9.318163070000001e-016},
  bhigh={29086.96214,-43.49160391},
  R=243.9629493135154);

constant IdealGases.Common.DataRecord H2SO4 (
  name="H2SO4",
  MM=0.09807848,
  Hf=-7470871.500047717,
  H0=168320.1962346888,
  Tlimit=1000,
  alow={-41291.5005,668.158989,-2.632753507,0.0541538248,-7.067502229999999e-
005,
      4.68461142e-008,-1.236791238e-011},
  blow={-93156.60120000001,39.61096201},
  ahigh={1437877.914,-6614.90253,21.57662058,-0.000480625597,3.010775121e-008,
      2.334842469e-012,-2.946330375e-016},
  bhigh={-52590.92950000001,-102.3603724},
  R=84.77366288710837);

constant IdealGases.Common.DataRecord H2BOH (
  name="H2BOH",
  MM=0.02983422,
  Hf=-9708110.284096584,
  H0=356044.3678433691,
  Tlimit=1000,
  alow={-86867.6678,1820.335089,-10.32373881,0.0492280106,-5.97861991e-005,
      3.94606273e-008,-1.068442257e-011},
  blow={-44152.3815,79.86177289999999},
  ahigh={2294795.193,-9382.923559999999,17.98228329,-0.001506116214,
      2.635641095e-007,-2.483163637e-011,9.736759120000001e-016},
  bhigh={20397.38953,-94.48201210000001},
  R=278.689102647899);

constant IdealGases.Common.DataRecord HB_OH_2 (
```

```

name="HB_OH_2",
MM=0.04583362000000001,
Hf=-14060393.30954003,
H0=265253.5845957617,
Tlimit=1000,
alow={-20670.44022, 953.4459160000001, -8.08895652, 0.0597242529, -
8.071336320000001e-005,
      5.61211835e-008, -1.553607566e-011},
blow={-82642.70809999999, 65.3957164},
ahigh={2122674.609, -8615.94893999998, 19.6617952, -0.00081484115,
      8.955082040000001e-008, -3.34483051e-012, -6.91957813999999e-017},
bhigh={-27887.90377, -99.8355961},
R=181.4055272090662);

constant IdealGases.Common.DataRecord H3BO3 (
  name="H3BO3",
  MM=0.06183302,
  Hf=-16243108.32626322,
  H0=219299.6234050998,
  Tlimit=1000,
  alow={25689.01843, 113.8029495, -4.04509658, 0.0592452168, -8.148028410000001e-
005,
      5.65859329e-008, -1.54927705e-011},
  blow={-122170.205, 41.3222014},
  ahigh={2297369.132, -8933.57179, 21.93496552, -0.000309478349, -5.06405299e-008,
      1.482296684e-011, -9.7644209e-016},
  bhigh={-69702.9847, -112.2292829},
  R=134.4665358412059);

constant IdealGases.Common.DataRecord H3B3O3 (
  name="H3B3O3",
  MM=0.08345502000000001,
  Hf=-14424069.24113133,
  H0=186966.8355480593,
  Tlimit=1000,
  alow={-198528.4188, 4104.13209, -28.27215617, 0.1269639253, -0.0001558505781,
      9.96630407e-008, -2.60367765e-011},
  blow={-164849.4375, 176.3917407},
  ahigh={1286220.713, -10932.58784, 32.1351867, -0.002595402269, 5.35376647e-007,
      -5.83356737999999e-011, 2.600757913e-015},
  bhigh={-87528.5689, -177.0847815},
  R=99.62818294214057);

constant IdealGases.Common.DataRecord H3B3O6 (
  name="H3B3O6",
  MM=0.13145322,
  Hf=-17220480.49488632,
  H0=180268.3342408805,
  Tlimit=1000,
  alow={-20670.31503, 708.339154, -7.12382095, 0.1025049979, -0.0001295004602,
      8.32273758999999e-008, -2.145746724e-011},
  blow={-277804.9216, 60.6359763},
  ahigh={1952433.352, -11549.32896, 38.9267611, -0.00111385228, 1.280317726e-007,
      -5.72846482e-012, -2.214529572e-017},
  bhigh={-212092.7136, -207.5566801},
  R=63.25042475186231);

```

---

**1088 Modelica.Media.IdealGases.Common.SingleGasesData**

---

```
constant IdealGases.Common.DataRecord H3F3 (
  name="H3F3",
  MM=0.0600190296,
  Hf=-14723276.86550934,
  H0=254303.0452461697,
  Tlimit=1000,
  allow={98885.62650000001,-1380.21288,8.98045847,0.01871609057,-3.127862508e-005,
         2.589436165e-008,-8.04037227e-012},
  blow={-101366.1491,-25.85302557},
  ahighe={2515790.373,-8789.42332,20.23933445,-0.001131704646,1.692779921e-007,
          -1.308662105e-011,3.978322e-016},
  bhigh={-54717.1365,-99.07787640000001},
  R=138.5305969691986);

constant IdealGases.Common.DataRecord H3Oplus (
  name="H3Oplus",
  MM=0.0190226714,
  Hf=31436173.57549476,
  H0=528129.1354273196,
  Tlimit=1000,
  allow={-64476.4015,1181.817922,-3.80189306,0.02220628313,-2.445343237e-005,
         1.573297747e-008,-4.15883641e-012},
  blow={65306.1332,42.8272313},
  ahighe={2955126.2,-9185.669409999999,13.41398696,-0.000559033921,
          1.138387119e-008,7.25992721e-012,-6.13373436e-016},
  bhigh={129053.4257,-70.2182818},
  R=437.0822491314233);

constant IdealGases.Common.DataRecord H4F4 (
  name="H4F4",
  MM=0.08002537279999999,
  Hf=-14831950.09870669,
  H0=270590.1796186322,
  Tlimit=1000,
  allow={131510.9024,-1840.36156,12.64093152,0.0249540736,-4.17039461e-005,
         3.45252384e-008,-1.07203436e-011},
  blow={-136400.3454,-41.2291353},
  ahighe={3354037.99,-11719.2204,27.65243565,-0.001508934621,2.257027572e-007,
          -1.744867255e-011,5.304351740000001e-016},
  bhigh={-74202.13160000001,-138.8603332},
  R=103.897947726899);

constant IdealGases.Common.DataRecord H5F5 (
  name="H5F5",
  MM=0.100031716,
  Hf=-14897154.03862511,
  H0=280362.4602421096,
  Tlimit=1000,
  allow={164136.1897,-2300.510396,16.30140545,0.03119205422,-5.21292635e-005,
         4.31561124e-008,-1.340031413e-011},
  blow={-171434.5408,-56.8215738},
  ahighe={4192290.73,-14649.03043,35.0655492,-0.001886170471,2.821289954e-007,
          -2.181090984e-011,6.63047416e-016},
  bhigh={-93687.042,-178.8592044},
  R=83.11835818151914);
```

```

constant IdealGases.Common.DataRecord H6F6 (
  name="H6F6",
  MM=0.1200380592,
  Hf=-15041438.88224411,
  H0=286030.0577068977,
  Tlimit=1000,
  alow={195688.6618,-2753.90988,19.75625055,0.0380771701,-6.341218879999999e-
005,
      5.21923329e-008,-1.612185649e-011},
  blow={-207926.941,-71.13607980000001},
  ahighe={5024520.319999999,-17428.17868,42.148705,-0.002084564791,
          2.930051519e-007,-2.052079159e-011,5.2069373e-016},
  bhigh={-115449.7653,-216.3947001},
  R=69.26529848459929);

constant IdealGases.Common.DataRecord H7F7 (
  name="H7F7",
  MM=0.1400444024,
  Hf=-14993094.60440098,
  H0=291530.7809546553,
  Tlimit=1000,
  alow={229386.7477,-3220.80786,23.62235212,0.0436680188,-7.29799034e-005,
      6.041786420000001e-008,-1.876025627e-011},
  blow={-241863.7479,-88.38812259999999},
  ahighe={5868789.83,-20508.63473,49.8917611,-0.002640634895,3.94979633e-007,-
3.053515088e-011,
      9.28260178e-016},
  bhigh={-133017.7809,-259.2385142},
  R=59.37025584394225);

constant IdealGases.Common.DataRecord He (
  name="He",
  MM=0.004002602,
  Hf=0,
  H0=1548349.798456104,
  Tlimit=1000,
  alow={0,0,2.5,0,0,0,0},
  blow={-745.375,0.9287239740000001},
  ahighe={0,0,2.5,0,0,0,0},
  bhigh={-745.375,0.9287239740000001},
  R=2077.26673798694);

constant IdealGases.Common.DataRecord Heplus (
  name="Heplus",
  MM=0.0040020534,
  Hf=594325271.3719412,
  H0=1548562.045673853,
  Tlimit=1000,
  alow={0,0,2.5,0,0,0,0},
  blow={285323.3739,1.621665557},
  ahighe={0,0,2.5,0,0,0,0},
  bhigh={285323.3739,1.621665557},
  R=2077.551488943151);

constant IdealGases.Common.DataRecord Hg (
  name="Hg",
  MM=0.20059,

```

---

**1090 Modelica.Media.IdealGases.Common.SingleGasesData**

---

```
Hf=305997.3079415724,
H0=30895.99680941224,
Tlimit=1000,
allow={0,0,2.5,0,0,0,0},
blow={6636.90008,6.80020154},
ahigh={51465.7351,-168.1269855,2.718343098,-0.0001445026192,5.15897766e-008,
      -9.47248501e-012,7.034797406e-016},
bhigh={7688.68493,5.27123609},
R=41.45008225734085);

constant IdealGases.Common.DataRecord Hgplus(
  name="Hgplus",
  MM=0.2005894514,
  Hf=5357425.928928983,
  H0=30896.08130809215,
  Tlimit=1000,
  allow={0.000409834494,-4.34358162e-006,2.500000019,-4.389036810000001e-011,
         5.5639692e-014,-3.63822826e-017,9.607881994999999e-021},
  blow={128503.7483,7.4933445},
  ahigh={-12299.84728,27.32269908,2.48418216,-4.42679761e-006,
         7.489685859999999e-009,-2.549887287e-012,2.819873366e-016},
  bhigh={128318.8257,7.62524457},
  R=41.45019562080521);

constant IdealGases.Common.DataRecord HgBr2 (
  name="HgBr2",
  MM=0.3603980000000001,
  Hf=-253363.2706063851,
  H0=43447.47751097397,
  Tlimit=1000,
  allow={-1991.826537,-190.2186083,8.246401260000001,-0.001607089392,
         1.947654549e-006,-1.242873546e-009,3.24181831e-013},
  blow={-12307.23096,-8.7166593},
  ahigh={-22436.4929,-3.31265227,7.50276517,-1.202360213e-006,
         2.830526999e-010,-3.4076867e-014,1.640497921e-018},
  bhigh={-13274.83366,-4.3685556},
  R=23.07025011237576);

constant IdealGases.Common.DataRecord I (
  name="I",
  MM=0.12690447,
  Hf=841262.7230545938,
  H0=48835.37987274995,
  Tlimit=1000,
  allow={169.8199675,-2.716437233,2.517385557,-5.73069207e-005,
         1.031716184e-007,-9.670641930000001e-011,3.706471651e-014},
  blow={12107.5009,7.40582313},
  ahigh={-778586.0569999999,2303.279568,0.002886686091,0.001180878463,
         -2.264074866e-007,
         1.963511339e-011,-6.243525940999999e-016},
  bhigh={-2616.792742,25.58922997},
  R=65.51756608730962);

constant IdealGases.Common.DataRecord Iplus (
  name="Iplus",
  MM=0.1269039214,
  Hf=8836220.470016144,
```

```

H0=48835.59098592204,
Tlimit=1000,
alow={-801.496901,8.13905261,2.47017476,4.16139382e-005,
    7.133397859999999e-009,-7.10595014e-011,4.873766102e-014},
blow={134079.4213,7.90342009},
ahigh={-778838.5329999999,2404.962651,-0.1791751142,0.001227311979,-
1.80149403e-007,
    9.923983959999999e-012,-9.775286439000001e-017},
bhigh={118853.1631,27.10544347},
R=65.5178493168297);

constant IdealGases.Common.DataRecord Iminus(
  name="Iminus",
  MM=0.1269050186,
  Hf=-1533395.401905721,
  H0=48835.1687614031,
  Tlimit=1000,
  alow={0,0,2.5,0,0,0,0},
  blow={-24149.70936,6.11346538},
  ahigh={0,0,2.5,0,0,0,0},
  bhigh={-24149.70936,6.11346538},
  R=65.51728286023828);

constant IdealGases.Common.DataRecord IF5(
  name="IF5",
  MM=0.221896486,
  Hf=-3790055.512641151,
  H0=90331.04742361714,
  Tlimit=1000,
  alow={202618.453,-3598.18317,25.30349738,-0.01348602458,1.105687229e-005,-
4.70793951e-009,
    7.86441147e-013},
  blow={-87001.4725,-111.2719941},
  ahigh={-362050.626,-163.9236093,16.12272743,-4.91443155e-005,
    1.086015052e-008,-1.244672691e-012,5.76359362e-017},
  bhigh={-106164.2293,-53.9877336},
  R=37.47004808359156);

constant IdealGases.Common.DataRecord IF7(
  name="IF7",
  MM=0.2598932924,
  Hf=-3699595.288208369,
  H0=92820.14467257561,
  Tlimit=1000,
  alow={299985.3564,-5501.55413,36.2713398,-0.02095765818,1.76197268e-005,-
7.82914233e-009,
    1.405939372e-012},
  blow={-93313.0766,-175.421615},
  ahigh={-561442.264,-264.6288326,22.19879891,-7.984221469999999e-005,
    1.768835964e-008,-2.031523755e-012,9.423767390000001e-017},
  bhigh={-122524.5322,-87.74010730000001},
  R=31.99186836728072);

constant IdealGases.Common.DataRecord I2(
  name="I2",
  MM=0.25380894,
  Hf=245933.0234782116,

```

---

**1092 Modelica.Media.IdealGases.Common.SingleGasesData**

---

```
H0=39857.23276729338,
Tlimit=1000,
alow={-5087.96877,-12.4958521,4.50421909,0.0001370962533,-1.390523014e-007,
      1.174813853e-010,-2.337541043e-014},
blow={6213.469810000001,5.58383694},
ahigh={-5632594.16,17939.6156,-17.23055169,0.0124421408,-3.33276858e-006,
      4.12547794e-010,-1.960461713e-014},
bhigh={-106850.5292,160.0531883},
R=32.75878304365481);

constant IdealGases.Common.DataRecord In(
  name="In",
  MM=0.114818,
  Hf=2096361.197721611,
  H0=53986.73552927241,
  Tlimit=1000,
  alow={51495.1805,-917.799902,8.93868795,-0.02224112602,3.82388194e-005,-
2.890116948e-008,
      8.047481221e-012},
  blow={32390.3162,-27.64567028},
  ahigh={-1683608.899,2210.473186,3.47221937,-0.001082267422,3.47969998e-007,
      -5.15809241e-011,3.183043089e-015},
  bhigh={10959.65206,2.557189088},
  R=72.41436011775157);

constant IdealGases.Common.DataRecord Inplus(
  name="Inplus",
  MM=0.1148174514,
  Hf=60935.2055344437,
  H0=53976.35920701163,
  Tlimit=1000,
  alow={0.2150488908,-0.002523854672,2.500011877,-2.869063919e-008,
      3.7538704e-011,-2.525942419e-014,6.84293266e-018},
  blow={96.1093017,5.9632544},
  ahigh={45369.9514,-144.5142103,2.681812441,-0.000115748006,3.94771481e-008,
      -6.87924362e-012,4.81920492e-016},
  bhigh={1004.26144,4.68375918},
  R=72.4147061149626);

constant IdealGases.Common.DataRecord InBr(
  name="InBr",
  MM=0.194722,
  Hf=-277913.5331395528,
  H0=51787.27108390423,
  Tlimit=1000,
  alow={-2540.851435,-52.94704710000001,4.71743809,-0.000434466949,
      6.51351364e-007,-4.40399515e-010,1.234046209e-013},
  blow={-7607.59428,4.258881},
  ahigh={794837.442,-2318.183711,7.03288958,-0.001245920568,3.27669335e-007,-
3.003771106e-011,
      5.85319277e-016},
  bhigh={6957.627289999999,-12.7350266},
  R=42.69919166812174);

constant IdealGases.Common.DataRecord InBr2(
  name="InBr2",
  MM=0.274626,
```

```

Hf=-545210.7921318448,
H0=52523.58116128844,
Tlimit=1000,
alow={-66.19878900000001,-236.589731,7.92654516,-0.001994547959,
      2.420414153e-006,-1.548924922e-009,4.05814839e-013},
blow={-18953.55588,-5.33886754},
ahigh={-321716.203,479.268210000001,7.10264948,-0.000551401978,
      3.157862421e-007,-5.51125451e-011,3.17216139e-015},
bhigh={-23709.23659,0.1193263347},
R=30.27561847749303);

constant IdealGases.Common.DataRecord InBr3(
  name="InBr3",
  MM=0.35453,
  Hf=-723739.2237610358,
  H0=56166.56700420276,
  Tlimit=1000,
  alow={-4990.1153,-280.5930452,11.1035606,-0.002379995808,2.887825281e-006,-
1.844500707e-009,
        4.81439587e-013},
  blow={-32505.0491,-18.71748987},
  ahigh={-35057.831,-4.85872132,10.00406131,-1.767620958e-006,4.16401606e-010,
        -5.01552887e-014,2.415414123e-018},
  bhigh={-33931.7924,-12.29069868},
  R=23.45209714269597);

constant IdealGases.Common.DataRecord InCL(
  name="InCL",
  MM=0.150271,
  Hf=-480118.4726261222,
  H0=64896.86632816712,
  Tlimit=1000,
  alow={6430.71303,-217.8267726,5.4188309,-0.0020480051,2.702295936e-006,-
1.795319798e-009,
        4.84746909e-013},
  blow={-8959.84107,-1.202999253},
  ahigh={-274575.1636,894.098690999999,3.26811839,0.0009180277780000001,-
3.083262431e-007,
        5.74545881e-011,-3.64527575e-015},
  bhigh={-15616.0343,12.66880374},
  R=55.32985073633636);

constant IdealGases.Common.DataRecord InCL2(
  name="InCL2",
  MM=0.185724,
  Hf=-1084853.955331567,
  H0=75145.82929508302,
  Tlimit=1000,
  alow={9935.164869999999,-417.210186,8.58849491,-0.00334838561,
        3.99786664e-006,-2.525301853e-009,6.54449033e-013},
  blow={-24264.80269,-11.64521725},
  ahigh={-332871.984,475.769161,7.10544369,-0.000552578696,3.160567738e-007,-
5.51445368e-011,
        3.17368038e-015},
  bhigh={-29950.95917,-2.294555511},
  R=44.76789214102647);

```

---

**1094 Modelica.Media.IdealGases.Common.SingleGasesData**

---

```
constant IdealGases.Common.DataRecord InCL3 (
  name="InCL3",
  MM=0.221177,
  Hf=-1671481.333954254,
  H0=82769.78618934158,
  Tlimit=1000,
  allow={26147.43044,-841.460456,13.1178479,-0.00643528247,7.55487475e-006,-
4.70507856e-009,
  1.204213663e-012},
  blow={-43264.6958,-35.0008094},
  ahighe={-71140.4363,-16.28425068,10.01309622,-5.54452888e-006,
  1.279877732e-009,-1.518261505e-013,7.22656277e-018},
  bhigh={-47589.7417,-16.69761011},
  R=37.59193767887258);

constant IdealGases.Common.DataRecord InF(
  name="InF",
  MM=0.1338164032,
  Hf=-1445413.150964142,
  H0=68878.7830160421,
  Tlimit=1000,
  allow={31901.2332,-529.525426,6.17288626,-0.002986035832,3.23377803e-006,-
1.856975575e-009,
  4.46502497e-013},
  blow={-21871.76282,-7.59121662},
  ahighe={-468900.12,1321.019051,2.881329353,0.001041591078,-3.090754867e-007,
  4.92664656e-011,-2.699156425e-015},
  bhigh={-33083.6981,13.96688303},
  R=62.13342909518585);

constant IdealGases.Common.DataRecord InF2(
  name="InF2",
  MM=0.1528148064,
  Hf=-2991770.462367971,
  H0=82379.29489010562,
  Tlimit=1000,
  allow={75224.8021,-1088.616881,9.48170520999999,-0.002893229367,
  1.516224649e-006,-7.98951781e-011,-1.466781026e-013},
  blow={-51243.36150000001,-21.4666609},
  ahighe={431896.034,-1688.47242,8.925863010000001,-0.001049188835,
  2.72393419e-007,-2.882959902e-011,1.073493779e-015},
  bhigh={-46736.91940000001,-19.32464363},
  R=54.40881152731022);

constant IdealGases.Common.DataRecord InF3(
  name="InF3",
  MM=0.1718132096,
  Hf=-5023359.775475611,
  H0=94989.40179277113,
  Tlimit=1000,
  allow={107160.8336,-1660.071901,13.95972813,-0.00501869326,3.22636753e-006,-
7.823385970000001e-010,
  -3.7269998e-014},
  blow={-97952.1499,-46.06353470000001},
  ahighe={-165685.5506,-79.6588929,10.05913347,-2.350816639e-005,
  5.16351293e-009,-5.88804881e-013,2.715090697e-017},
  bhigh={-106868.2627,-21.32968218},
  R=48.39250730113827);
```

```

constant IdealGases.Common.DataRecord InH(
  name="InH",
  MM=0.11582594,
  Hf=1856379.529490544,
  H0=74975.46749890395,
  Tlimit=1000,
  allow={-51528.2872, 781.3792590000001, -0.8433385339999999, 0.01044471274, --
9.76107481999999e-006,
  4.54063519999999e-009, -8.169906900000001e-013},
  blow={21100.48908, 29.39346222},
  ahigh={779740.666, -3646.88208, 9.54296735, -0.00348128172, 1.304805603e-006, --
2.182950701e-010,
  1.274009748e-014},
  bhigh={46432.9546, -36.5932923},
  R=71.78419618265131);

constant IdealGases.Common.DataRecord InI(
  name="InI",
  MM=0.24172247,
  Hf=109287.0017421219,
  H0=42491.39105685955,
  Tlimit=1000,
  allow={-468.634874, -56.8553099, 4.78184969, -0.000661936831, 1.027666742e-006, --
7.27059207e-010,
  2.077991102e-013},
  blow={2095.570254, 4.88551513},
  ahigh={1529221.772, -4690.373680000001, 10.04593462, -0.00314043411,
  9.460834390000001e-007, -1.253862848e-010, 5.94980564e-015},
  bhigh={31524.78143, -32.9769749},
  R=34.39676915431156);

constant IdealGases.Common.DataRecord InI2(
  name="InI2",
  MM=0.36862694,
  Hf=-107047.637375608,
  H0=40454.87559862011,
  Tlimit=1000,
  allow={-4513.8906, -118.372687, 7.47311008, -0.00103447409, 1.271465534e-006, --
8.2278203e-010,
  2.178307019e-013},
  blow={-6278.53604, -0.66672975},
  ahigh={-313067.1665, 481.38107, 7.10091418, -0.000550655992, 3.156120735e-007, --
5.50917054e-011,
  3.17116283e-015},
  bhigh={-10430.1275, 2.141662307},
  R=22.55524786115741);

constant IdealGases.Common.DataRecord InI3(
  name="InI3",
  MM=0.49553141,
  Hf=-212773.4223749813,
  H0=42080.25077562692,
  Tlimit=1000,
  allow={-8728.397169999998, -126.5914859, 10.5065912, -0.001106109314,
  1.354335766e-006, -8.709453460000001e-010, 2.28518635e-013},
  blow={-15082.69063, -12.11411745},
  ahigh={-21967.42567, -2.151167521, 10.001822, -8.00221469999999e-007,
  1.897343164e-010, -2.29625246e-014, 1.109838196e-018},

```

---

**1096 Modelica.Media.IdealGases.Common.SingleGasesData**

---

```
bhigh={-15724.06335,-9.17088869},  
R=16.7789000499484);  
  
constant IdealGases.Common.DataRecord InO(  
  name="InO",  
  MM=0.1308174,  
  Hf=1116006.807962855,  
  H0=69120.07882743428,  
  Tlimit=1000,  
  allow={-115472.7169,1753.668703,-6.96503381,0.02961698958,-3.123332334e-005,  
         1.43724472e-008,-2.235519898e-012},  
  blow={8188.67959,66.0862126},  
  ahigh={-805213.0950000001,1693.064203,4.76086358,-0.00073407184,  
         3.22959776e-007,-4.58015038e-011,1.954611725e-015},  
  bhigh={4005.51981,3.50074246},  
  R=63.55784475153917);  
  
constant IdealGases.Common.DataRecord InOH(  
  name="InOH",  
  MM=0.13182534,  
  Hf=-944032.0351155552,  
  H0=81447.45919107813,  
  Tlimit=1000,  
  allow={61234.6302,-1195.863129,10.41427981,-0.0068700025,6.21412878e-006,-  
         2.408203181e-009,  
         3.17764178e-013},  
  blow={-10798.5217,-31.8989084},  
  ahigh={852512.8929999999,-2350.317172,7.98451514,9.91581562e-005,-  
         6.21238545e-008,  
         1.016169575e-011,-5.68255763999999e-016},  
  bhigh={-1536.21361,-20.43107643},  
  R=63.07187980702344);  
  
constant IdealGases.Common.DataRecord In2Br2(  
  name="In2Br2",  
  MM=0.389444,  
  Hf=-504065.321329896,  
  H0=55039.85938928319,  
  Tlimit=1000,  
  allow={-9221.1883,-36.1667703,10.14728771,-0.00032557752,4.02232748e-007,-  
         2.604080237e-010,  
         6.86768955e-014},  
  blow={-26448.83479,-10.00360073},  
  ahigh={-12907.81804,-0.607540124,10.00052163,-2.312058331e-007,  
         5.51731572e-011,-6.708620190000001e-015,3.25384165e-019},  
  bhigh={-26631.36878,-9.149949469999999},  
  R=21.34959583406087);  
  
constant IdealGases.Common.DataRecord In2Br4(  
  name="In2Br4",  
  MM=0.549252,  
  Hf=-794732.9877724615,  
  H0=57893.49333275072,  
  Tlimit=1000,  
  allow={-14255.55379,-286.6009341,17.13866292,-0.002473506655,  
         3.017273071e-006,-1.93490043e-009,5.06587425e-013},  
  blow={-55937.5306,-38.3710598},
```

```

ahigh={-44534.6371,-4.91152398,16.00413764,-1.810579975e-006,
       4.28171574e-010,-5.17199802e-014,2.496142837e-018},
bhigh={-57391.7541,-31.7490639},
R=15.13780923874652);

constant IdealGases.Common.DataRecord In2Br6(
  name="In2Br6",
  MM=0.7090599999999999,
  Hf=-886643.4857416863,
  H0=60780.08631145462,
  Tlimit=1000,
  allow={-17589.40752,-485.903448,23.91654505,-0.00414206149,5.03384932e-006,-
3.21910365e-009,
        8.410220069999999e-013},
  blow={-79888.4446,-64.3787188},
  ahigh={-69450.3948,-8.402368170000001,22.00703884,-3.068251799e-006,
         7.23596573e-010,-8.722870580000001e-014,4.20346991e-018},
  bhigh={-82357.5985000001,-53.221928},
  R=11.72604857134799);

constant IdealGases.Common.DataRecord In2Cl2(
  name="In2Cl2",
  MM=0.300542,
  Hf=-772528.8811547139,
  H0=67291.41018559803,
  Tlimit=1000,
  allow={-11428.48747,-141.6015827,10.56853314,-0.00124425344,1.526076784e-006,
         -9.82640281e-010,2.58076699e-013},
  blow={-30263.27087,-15.6510496},
  ahigh={-26166.06332,-2.401955489,10.00203984,-8.97519003e-007,
         2.130755553e-010,-2.5811409e-014,1.248404822e-018},
  bhigh={-30980.16735,-12.34946505},
  R=27.66492536816818);

constant IdealGases.Common.DataRecord In2Cl4(
  name="In2Cl4",
  MM=0.371448,
  Hf=-1559103.443819862,
  H0=79301.88074777628,
  Tlimit=1000,
  allow={9814.08308,-848.895901,19.2341036,-0.00681567424,
         8.129220730000001e-006,-5.12507685e-009,1.324324755e-012},
  blow={-70276.8814,-55.82665695},
  ahigh={-85037.7179,-15.52464614,16.01269668,-5.44154987e-006,
         1.267594997e-009,-1.514110365e-013,7.24540454e-018},
  bhigh={-74619.21740000001,-36.91150615},
  R=22.38394607051324);

constant IdealGases.Common.DataRecord In2Cl6(
  name="In2Cl6",
  MM=0.442354,
  Hf=-1994646.886882452,
  H0=86861.3418212563,
  Tlimit=1000,
  allow={30046.63159,-1585.621359,27.9648947,-0.01245506052,1.475341213e-005,-
9.2526892e-009,
        2.381233885e-012},

```

## 1098 Modelica.Media.IdealGases.Common.SingleGasesData

---

```
blow={-104882.83,-97.88404639999999},  
ahigh={-149940.0484,-29.88680816,22.02425362,-1.033652301e-005,  
2.397911922e-009,-2.855307516e-013,1.363051661e-017},  
bhigh={-113011.0233,-62.93985679999999},  
R=18.79596883943629);  
  
constant IdealGases.Common.DataRecord In2F2 (  
  name="In2F2",  
  MM=0.2676328064,  
  Hf=-1988673.612772758,  
  H0=67405.41356890992,  
  Tlimit=1000,  
  allow={7255.7171,-644.826565,12.47539512,-0.00524608962,6.28361621e-006,-  
3.97434234e-009,  
1.029559356e-012},  
blow={-63848.92449999999,-31.08004586},  
ahigh={-64097.1544,-11.63740672,10.00956511,-4.11408691e-006,  
9.60882832e-010,-1.150013486e-013,5.511437479999999e-018},  
bhigh={-67142.7785,-16.61718474},  
R=31.06671454759293);  
  
constant IdealGases.Common.DataRecord In2F4 (  
  name="In2F4",  
  MM=0.3056296128,  
  Hf=-4203742.282135299,  
  H0=81023.15012323308,  
  Tlimit=1000,  
  allow={122521.3611,-2444.998374,23.42425457,-0.0129142712,1.308700476e-005,-  
7.18167393e-009,  
1.648386404e-012},  
blow={-146694.2111,-89.43560050000001},  
ahigh={-220597.7486,-78.2208377,16.05923872,-2.39402324e-005,  
5.32958279e-009,-6.14460957e-013,2.859099598e-017},  
bhigh={-159571.6347,-44.6957458},  
R=27.20440576365511);  
  
constant IdealGases.Common.DataRecord In2F6 (  
  name="In2F6",  
  MM=0.3436264192,  
  Hf=-5703868.767026397,  
  H0=97684.86974356599,  
  Tlimit=1000,  
  allow={182549.0583,-3552.11171,32.75425230000001,-0.01868064635,  
1.892443162e-005,-1.038916311e-008,2.386637271e-012},  
blow={-223964.876,-136.3626758},  
ahigh={-317096.638,-116.7440973,22.08847456,-3.57773393e-005,  
7.96895904e-009,-9.191681699999999e-013,4.27849415000001e-017},  
bhigh={-242666.1118,-71.5532856},  
R=24.19625365056914);  
  
constant IdealGases.Common.DataRecord In2I2 (  
  name="In2I2",  
  MM=0.48344494,  
  Hf=-57531.93734947356,  
  H0=45627.69030119541,  
  Tlimit=1000,  
  allow={-6681.92501,-14.49103222,10.05929122,-0.0001314921486,
```

```

  1.628373589e-007,-1.056087811e-010,2.788963879e-014},
blow={-6279.602599999999,-7.29113098},
ahigh={-8148.157299999999,-0.2444544443,10.00021086,-9.37678925e-008,
  2.24298818e-011,-2.732299303e-015,1.327135699e-019},
bhigh={-6352.6515,-6.94772542},
R=17.19838457715578);

constant IdealGases.Common.DataRecord In2I4(
  name="In2I4",
  MM=0.73725388,
  Hf=-270114.7954623175,
  H0=45396.81635856565,
  Tlimit=1000,
  allow={-15010.28064,-97.447979,16.3938395,-0.000865938021,1.065671801e-006,-
6.87925268e-010,
  1.810240886e-013},
blow={-28303.88443,-29.86294828},
ahigh={-25056.99251,-1.642155234,16.00140149,-6.18706628e-007,
  1.472308565e-010,-1.786593333e-014,8.652338779999999e-019},
bhigh={-28796.55333,-27.57790375},
R=11.2776239305787);

constant IdealGases.Common.DataRecord In2I6(
  name="In2I6",
  MM=0.99106282,
  Hf=-322603.2028928297,
  H0=45494.71445210709,
  Tlimit=1000,
  allow={-22326.40607,-200.0786874,22.80417927,-0.001761331313,
  2.161510393e-006,-1.392400447e-009,3.65817188e-013},
blow={-44125.4768,-52.7380203},
ahigh={-43119.0585,-3.39035059,22.00288128,-1.268348706e-006,3.0121219e-010,
  -3.64967671e-014,1.765530622e-018},
bhigh={-45138.2027,-48.0686837},
R=8.389450024974199);

constant IdealGases.Common.DataRecord In2O(
  name="In2O",
  MM=0.2456354,
  Hf=-141525.6392197542,
  H0=52095.52043394397,
  Tlimit=1000,
  allow={50656.535,-807.036367,8.64206727,-0.001533035446,2.866419382e-007,
  4.98113497e-010,-2.554320291e-013},
blow={-1924.908459,-14.90718759},
ahigh={-92327.8636,-52.0112728,7.03872362,-1.543821179e-005,3.39970087e-009,
  -3.8854571e-013,1.79512527e-017},
bhigh={-6270.9268,-4.42182041},
R=33.84883449209683);

constant IdealGases.Common.DataRecord K(
  name="K",
  MM=0.0390983,
  Hf=2276313.803924979,
  H0=158508.8865756312,
  Tlimit=1000,
  allow={9.665143929999999,-0.1458059455,2.500865861,-2.601219276e-006,

```

---

**1100 Modelica.Media.IdealGases.Common.SingleGasesData**

---

```
        4.18730657999999e-009,-3.43972211e-012,1.131569009e-015},  
blow={9959.493490000001,5.03582226},  
ahigh={-3566422.36,10852.89825,-10.54134898,0.00800980135,-2.696681041e-006,  
        4.71529415e-010,-2.97689735e-014},  
bhigh={-58753.3701,97.3855124},  
R=212.6555886061542);  
  
constant IdealGases.Common.DataRecord Kplus(  
    name="Kplus",  
    MM=0.0390977514,  
    Hf=13146728.63769859,  
    H0=158511.1106926727,  
    Tlimit=1000,  
    allow={0,0,2.5,0,0,0},  
    blow={61075.1686,4.34740444},  
    ahigh={0,0,2.5,0,0,0},  
    bhigh={61075.1686,4.34740444},  
    R=212.6585724825086);  
  
constant IdealGases.Common.DataRecord Kminus(  
    name="Kminus",  
    MM=0.0390988486,  
    Hf=880284.9503859814,  
    H0=158506.6625210033,  
    Tlimit=1000,  
    allow={0,0,2.5,0,0,0},  
    blow={3394.15071,4.34744653},  
    ahigh={0,0,2.5,0,0,0},  
    bhigh={3394.15071,4.34744653},  
    R=212.6526048135341);  
  
constant IdealGases.Common.DataRecord KALF4 (  
    name="KALF4",  
    MM=0.1420734508,  
    Hf=-13428665.32245868,  
    H0=150776.8754779904,  
    Tlimit=1000,  
    allow={119932.7246,-2156.457352,18.15348004,0.003115762809,-9.51199167e-006,  
          8.43175623e-009,-2.615583061e-012},  
    blow={-222254.5809,-68.23299969999999},  
    ahigh={-343513.537,-242.0680834,16.1802925,-7.19381768e-005,  
          1.585712708e-008,-1.814017733e-012,8.38839117999999e-017},  
    bhigh={-233965.8121,-52.338987},  
    R=58.5223484977814);  
  
constant IdealGases.Common.DataRecord KBO2 (  
    name="KBO2",  
    MM=0.08190810000000001,  
    Hf=-8155765.998722958,  
    H0=172030.641658151,  
    Tlimit=1000,  
    allow={42963.08650000001,-720.961643,8.34963935,0.002407820401,-1.680003427e-  
007,  
          -1.231930486e-009,5.479951640000001e-013},  
    blow={-78685.2347,-14.76077068},  
    ahigh={88907.266,-1671.432436,11.1748393,-0.000451376026,9.69900346e-008,-  
1.09002769e-011},
```

```

        4.9768458e-016},
bhigh={-73753.36080000001,-32.7474937},
R=101.5097652124759);

constant IdealGases.Common.DataRecord KBr(
  name="KBr",
  MM=0.1190023,
  Hf=-1506280.786169679,
  H0=84948.96317130004,
  Tlimit=1000,
  allow={9203.30919,-213.7880361,5.58850915,-0.002753854319,
    4.013672610000001e-006,-2.856323127e-009,8.13520974e-013},
  blow={-21883.87013,-1.707501588},
  ahight={1562614.367,-4384.89478,9.045464600000001,-0.0020456809,
    4.40192385e-007,-1.448178289e-011,-2.27333876e-015},
  bhigh={5315.34201,-28.64647026},
  R=69.86816221199086);

constant IdealGases.Common.DataRecord KCN(
  name="KCN",
  MM=0.0651157,
  Hf=1220842.285347466,
  H0=191295.4018769667,
  Tlimit=1000,
  allow={17986.27669,-630.055897,9.988695,-0.01076486132,1.75101498e-005,-
    1.265592431e-008,
    3.46976281e-012},
  blow={10580.25371,-25.94060289},
  ahight={361566.387,-1749.018011,8.680702670000001,-0.000440235004,
    9.249499710000001e-008,-1.021916855e-011,4.604767189999999e-016},
  bhigh={18137.34856,-22.81619241},
  R=127.6876697939207);

constant IdealGases.Common.DataRecord KCL(
  name="KCL",
  MM=0.0745513,
  Hf=-2878217.831211528,
  H0=132594.8575008082,
  Tlimit=1000,
  allow={9058.35151,-245.6801212,5.68069619,-0.002900127425,4.13098306e-006,-
    2.907340629e-009,
    8.22385087e-013},
  blow={-25973.04884,-3.677976854},
  ahight={-212294.5722,934.61589,2.866264958,0.001468386693,-5.83426078e-007,
    1.255777709e-010,-9.15014799999999e-015},
  bhigh={-32737.8764,14.01864636},
  R=111.5268546624942);

constant IdealGases.Common.DataRecord KF(
  name="KF",
  MM=0.0580967032,
  Hf=-5653416.939500278,
  H0=162747.8579541842,
  Tlimit=1000,
  allow={14357.04906,-339.184823,5.72790672,-0.002518371562,3.26591564e-006,-
    2.21011086e-009,
    6.21204205e-013},

```

---

**1102 Modelica.Media.IdealGases.Common.SingleGasesData**

---

```
blow={-39142.5442,-5.81099962},
ahigh={-1483743.237,4550.04804,-1.081464319,0.00350396588,-1.093902537e-006,
1.771028824e-010,-1.029032783e-014},
bhigh={-69633.2981,40.8863592},
R=143.1143514525623);

constant IdealGases.Common.DataRecord KH(
  name="KH",
  MM=0.04010624,
  Hf=3126673.205964957,
  H0=219295.2019436377,
  Tlimit=1000,
  alow={223.778215,179.3554623,1.130735415,0.009966112559999999,-1.341218171e-
005,
  9.034529940000001e-009,-2.40988777e-012},
  blow={13382.50358,15.52739897},
  ahigh={-3752276.52,11727.78444,-10.14137678,0.008854548899999998,-
2.517188074e-006,
  3.34407236e-010,-1.701157835e-014},
  bhigh={-60251.033,101.030459},
  R=207.3111815019309);

constant IdealGases.Common.DataRecord KI(
  name="KI",
  MM=0.16600277,
  Hf=-773817.6718376447,
  H0=61710.55458893849,
  Tlimit=1000,
  alow={-457.281174,-63.31769730000001,4.82329242,-0.0007876668320000001,
  1.336737911e-006,-1.000530992e-009,3.004503956e-013},
  blow={-16503.40769,3.55259482},
  ahigh={3293747.78,-10028.24194,16.29302044,-0.00669639675,2.001412089e-006,
  -2.677673023e-010,1.273359646e-014},
  bhigh={46763.4867,-78.5912337},
  R=50.08634494472592);

constant IdealGases.Common.DataRecord KLi(
  name="KLi",
  MM=0.0460393,
  Hf=3707752.355053183,
  H0=221487.3597122459,
  Tlimit=1000,
  alow={-3426.04369,-19.86063081,4.48760705,0.0005249535550000001,-
1.207990165e-006,
  1.705161047e-009,-7.879994210000001e-013},
  blow={19278.68247,1.912285942},
  ahigh={12112968.43,-40810.0487,56.8379212,-0.03128695515,
  8.975062200000001e-006,-1.189303691e-009,5.88012918e-014},
  bhigh={273879.2595,-366.96936},
  R=180.5951002730276);

constant IdealGases.Common.DataRecord KNO2(
  name="KNO2",
  MM=0.08510380000000001,
  Hf=-2261913.028560417,
  H0=180281.0920311431,
  Tlimit=1000,
```

```

alow={-72462.36930000001,1226.722811,-2.258926457,0.03019643903,-
3.64025995e-005,
      2.219135205e-008,-5.471654230000001e-012},
blow={-30772.6896,45.37172549},
ahigh={-168941.4093,-851.5973150000001,10.63209586,-0.000252325481,
      5.57364441e-008,-6.39318592e-012,2.964484049e-016},
bhigh={-21877.01436,-27.55200432},
R=97.69801113463794);

constant IdealGases.Common.DataRecord KNO3 (
  name="KNO3",
  MM=0.1011032,
  Hf=-3123866.386029324,
  H0=157434.304749998,
  Tlimit=1000,
  alow={-25961.12303,820.060661,-2.886015624,0.042072889,-5.27032309e-005,
        3.30856173e-008,-8.36586037e-012},
  blow={-43350.5447,46.03187331},
  ahigh={-318368.287,-1375.234449,14.01587531,-0.000404066008,8.90084778e-008,
        -1.018771993e-011,4.71598202e-016},
  bhigh={-35138.5322,-48.04256439},
  R=82.23747616297011);

constant IdealGases.Common.DataRecord KNa (
  name="KNa",
  MM=0.06208807,
  Hf=2132524.444712164,
  H0=170308.5149852459,
  Tlimit=1000,
  alow={25424.05292,-411.920741,6.84108795,-0.00625048138,8.86101314e-006,-
4.922415349999999e-009,
        6.493212469999999e-013},
  blow={16525.9989,-9.13616691},
  ahigh={6260326.62,-25635.93614,43.7333545,-0.0267265511,8.23270579e-006,-
1.12850359e-009,
        5.66326489e-014},
  bhigh={169727.4909,-266.4117219},
  R=133.9141641864532);

constant IdealGases.Common.DataRecord KO (
  name="KO",
  MM=0.05509770000000001,
  Hf=1174882.327211481,
  H0=172078.2174210539,
  Tlimit=1000,
  alow={14625.62908,-338.476565,5.71660764,-0.002363265083,2.848716276e-006,-
1.739858233e-009,
        4.431006520000001e-013},
  blow={8141.83538,-4.02210152},
  ahigh={696010.338,-3304.83529,10.05743444,-0.004331112,1.747281632e-006,-
3.012370548e-010,
        1.79082787e-014},
  bhigh={26049.72496,-34.4878152},
  R=150.9041575238168);

constant IdealGases.Common.DataRecord KOH (
  name="KOH",

```

---

**1104 Modelica.Media.IdealGases.Common.SingleGasesData**

---

```
MM=0.05610564,
Hf=-4135056.653840862,
H0=208083.3584644966,
Tlimit=1000,
alow={17706.84196,-615.320522,8.684075719999999,-0.00396284951,
      3.40865059e-006,-9.60197222e-010,8.494054970000001e-015},
blow={-26779.03261,-21.74495666},
ahigh={891727.195,-2334.179072,7.97257871,0.0001038863156,-
6.315893469999999e-008,
      1.027938106e-011,-5.73668582e-016},
bhigh={-14436.96469,-20.76401416},
R=148.1931584774721);

constant IdealGases.Common.DataRecord K2(
  name="K2",
  MM=0.07819660000000001,
  Hf=1618309.862577145,
  H0=137360.5629912298,
  Tlimit=1000,
  alow={15241.69293,-330.178936,7.07079595,-0.00976707246,2.021535863e-005,-
1.886092452e-008,
      6.11297464e-012},
  blow={15334.02849,-9.1010358},
  ahigh={-27344707.45,65621.80009999999,-44.7635044,0.008938859150000001,
      2.984557092e-006,-1.064158914e-009,8.334936929999999e-014},
  bhigh={-422624.383,386.714251},
  R=106.3277943030771);

constant IdealGases.Common.DataRecord K2plus(
  name="K2plus",
  MM=0.07819605140000001,
  Hf=6709555.298594016,
  H0=138896.1565903314,
  Tlimit=1000,
  alow={51960.3657,-611.338253,7.26499054,-0.00581063482,6.5674965e-006,-
2.378020865e-009,
      -1.318637581e-013},
  blow={64798.2027,-10.42370517},
  ahigh={11079507.39,-41774.8382,64.48659840000001,-0.0402499803,
      1.360587923e-005,-2.361920107e-009,1.67343061e-013},
  bhigh={317761.205,-410.50631},
  R=106.3285402669322);

constant IdealGases.Common.DataRecord K2Br2(
  name="K2Br2",
  MM=0.2380046,
  Hf=-2263587.31301832,
  H0=88031.0968779595,
  Tlimit=1000,
  alow={-10930.40504,-48.5665148,10.19754923,-0.00043631571,5.38717366e-007,-
3.4861165e-010,
      9.19070295e-014},
  blow={-67580.73239999999,-12.94944706},
  ahigh={-15892.29976,-0.810137946,10.00069442,-3.074491521e-007,
      7.33102416e-011,-8.908944780000001e-015,4.319245429999999e-019},
  bhigh={-67825.9544,-11.80425726},
  R=34.93408110599543);
```

```

constant IdealGases.Common.DataRecord K2CO3 (
  name="K2CO3",
  MM=0.1382055,
  Hf=-5872770.352844134,
  H0=141208.8303287496,
  Tlimit=1000,
  allow={-44074.067,706.981544,0.693148247,0.0396783132,-4.8509583e-005,
         2.976635787e-008,-7.374761529999999e-012},
  blow={-103391.3232,29.80651689},
  ahighelement={-326426.316,-1521.168973,17.12283562,-0.000446458071,9.83337963e-008,
              -1.125478371e-011,5.21006912e-016},
  bhighelement={-94882.7509999999,-62.1522063},
  R=60.16021070073188);

constant IdealGases.Common.DataRecord K2C2N2 (
  name="K2C2N2",
  MM=0.1302314,
  Hf=-64254.85712355085,
  H0=190324.6221725329,
  Tlimit=1000,
  allow={4627.78948,-972.9810620000001,19.9666591,-0.01953015081,
         3.27533046e-005,-2.39398512e-008,6.59654824e-012},
  blow={-777.4411259999999,-67.55613632000001},
  ahighelement={726959.95,-3492.10868,18.35676822,-0.000878547514,1.845532919e-007,
               -2.038717568e-011,
               9.18542099e-016},
  bhighelement={15826.00773,-67.28484052},
  R=63.84383489696034);

constant IdealGases.Common.DataRecord K2CL2 (
  name="K2CL2",
  MM=0.1491026,
  Hf=-4127316.780525624,
  H0=133890.6766213332,
  Tlimit=1000,
  allow={-13285.26456,-124.1832429,10.5003573,-0.001097776004,1.348871402e-006,
         -8.69721512e-010,2.28658408e-013},
  blow={-76443.6315,-17.90150052},
  ahighelement={-26146.11331,-2.096229095,10.00178456,-7.864803710000001e-007,
               1.869281152e-010,-2.266272348e-014,1.096790704e-018},
  bhighelement={-77071.8933,-14.99720802},
  R=55.76342733124707);

constant IdealGases.Common.DataRecord K2F2 (
  name="K2F2",
  MM=0.1161934064,
  Hf=-7400381.034013649,
  H0=155995.3061157522,
  Tlimit=1000,
  allow={-1611.833856,-513.564489,11.99922511,-0.004280109549999999,
         5.16538754e-006,-3.2858149e-009,8.549648749999999e-013},
  blow={-103924.8602,-30.39925079},
  ahighelement={-57408.9253,-9.04618211,10.00750685,-3.25076484e-006,7.63006459e-010,
               -9.165649529999999e-014,4.40504844e-018},
  bhighelement={-106541.3546,-18.7403762},
  R=71.55717572628115);

```

---

**1106 Modelica.Media.IdealGases.Common.SingleGasesData**

---

```
constant IdealGases.Common.DataRecord K2I2(
  name="K2I2",
  MM=0.33200554,
  Hf=-1261772.475242431,
  H0=64627.07821080335,
  Tlimit=1000,
  allow={-8977.75244,-27.76460169,10.11331014,-0.0002508421994,
         3.102352114e-007,-2.010096209e-010,5.30443715e-014},
  blow={-53262.0839,-10.32092509},
  ahighelement={-11796.34729,-0.472476155,10.00040695,-1.807689054e-007,
                4.32056097e-011,-5.259699430000001e-015,2.553419144e-019},
  bhigh={-53402.1023000001,-9.6644405},
  R=25.04317247236296);

constant IdealGases.Common.DataRecord K2O(
  name="K2O",
  MM=0.094196,
  Hf=-786518.5570512548,
  H0=147121.93723725,
  Tlimit=1000,
  allow={23920.44068,-544.535839,8.82640323,-0.00348142943,3.83454207e-006,-
2.268494189e-009,
         5.56921225e-013},
  blow={-8234.29168,-16.63099064},
  ahighelement={-46114.6458,-13.63119524,7.01053044,-4.32365826e-006,
                9.747788419999999e-010,-1.135283694e-013,5.325860600000001e-018},
  bhigh={-11072.22244,-5.76871872},
  R=88.26778207142554);

constant IdealGases.Common.DataRecord K2Oplus(
  name="K2Oplus",
  MM=0.0941954513999999,
  Hf=3910914.12084894,
  H0=150093.4576974701,
  Tlimit=1000,
  allow={7201.10273,-333.65264,8.05073035,-0.001869353797,1.921291874e-006,-
1.064020951e-009,
         2.457514799e-013},
  blow={43899.8636,-10.7320415},
  ahighelement={-37800.335,-8.84953739,7.00684789,-2.815510856e-006,6.3550015e-010,-
7.408551990000001e-014,
                3.478334960000001e-018},
  bhigh={42145.1968000001,-4.41827661},
  R=88.26829614832124);

constant IdealGases.Common.DataRecord K2O2(
  name="K2O2",
  MM=0.1101954,
  Hf=-1738423.754530588,
  H0=147826.5245191723,
  Tlimit=1000,
  allow={48108.7061,-1003.601504,11.54229002,-0.000454633252,-1.516606209e-006,
         1.79942609e-009,-6.12833729e-013},
  blow={-20571.5192,-31.8119979},
  ahighelement={-147607.3073,-102.193204,10.07664713,-3.076517991e-005,
                6.81525281e-009,-7.82865445e-013,3.6325061e-017},
  bhigh={-25920.56324,-21.62287184},
  R=75.45207876190838);
```

```

constant IdealGases.Common.DataRecord K2O2H2 (
  name="K2O2H2",
  MM=0.11221128,
  Hf=-5712438.179120673,
  H0=199489.1511798101,
  Tlimit=1000,
  allow={8174.78837,-1130.63068,18.15303256,-0.00773743108,6.83761401e-006,-
2.054691111e-009,
  7.67287751e-014},
  blow={-75749.6517,-63.9175947},
  ahigh={1773523.196,-4665.292469999999,16.94308128,0.0002085297143,-
1.264714762e-007,
  2.057506565e-011,-1.148042134e-015},
  bhigh={-50512.63310000001,-63.3477392},
  R=74.09657923873607);

constant IdealGases.Common.DataRecord K2SO4 (
  name="K2SO4",
  MM=0.1742592,
  Hf=-6288627.06244491,
  H0=127877.7763240047,
  Tlimit=1000,
  allow={62714.9231,-815.204992,7.45541372,0.0384990978,-5.40063098e-005,
  3.6546568e-008,-9.760223570000001e-012},
  blow={-130469.2568,-10.3463477},
  ahigh={-544612.564,-959.0667229999999,19.71591285,-0.0002866586009,
  6.342480639999999e-008,-7.281462539999999e-012,3.37788693e-016},
  bhigh={-133787.3384,-73.99103550999999},
  R=47.71324555604525);

constant IdealGases.Common.DataRecord Kr (
  name="Kr",
  MM=0.0838,
  Hf=0,
  H0=73954.98806682577,
  Tlimit=1000,
  allow={0,0,2.5,0,0,0,0},
  blow={-745.375,5.49095651},
  ahigh={264.3639057,-0.7910050820000001,2.500920585,-5.32816411e-007,
  1.620730161e-010,-2.467898017e-014,1.47858504e-018},
  bhigh={-740.348894,5.48439815},
  R=99.21804295942721);

constant IdealGases.Common.DataRecord Krplus (
  name="Krplus",
  MM=0.0837994513999999,
  Hf=16192873.52518396,
  H0=73955.472219237,
  Tlimit=1000,
  allow={-5650.40286,69.3074081,2.157028132,0.0008711228930000001,-1.18160973e-
006,
  7.86219863e-010,-1.832589387e-013},
  blow={162116.4118,8.81824226},
  ahigh={-221656.7015,1166.16784,0.486965532,0.001429223599,-3.94962861e-007,
  4.98285351e-011,-2.406719258e-015},
  bhigh={155600.2861,20.59230986},
  R=99.21869249850485);

```

```
constant IdealGases.Common.DataRecord Li (
  name="Li",
  MM=0.006941,
  Hf=22950583.48941075,
  H0=892872.4967583921,
  Tlimit=1000,
  allow={0,0,2.5,0,0,0,0},
  blow={18413.90197,2.447622965},
  ahigh={1125610.652,-3463.53673,6.56661192,-0.002260983356,5.92228916e-007,-
6.2816351e-011,
  2.884948238e-015},
  bhigh={40346.374,-26.55918195},
  R=1197.878115545311);

constant IdealGases.Common.DataRecord Liplus (
  name="Liplus",
  MM=0.0069404514,
  Hf=98800407.70835166,
  H0=892943.0728381731,
  Tlimit=1000,
  allow={0,0,2.5,0,0,0,0},
  blow={81727.24550000001,1.754357228},
  ahigh={0,0,2.5,0,0,0,0},
  bhigh={81727.24550000001,1.754357228},
  R=1197.972800443499);

constant IdealGases.Common.DataRecord Liminus (
  name="Liminus",
  MM=0.0069415486,
  Hf=13465976.16560662,
  H0=892801.9318340579,
  Tlimit=1000,
  allow={0,0,2.5,0,0,0,0},
  blow={10496.98659,1.754594332},
  ahigh={0,0,2.5,0,0,0,0},
  bhigh={10496.98659,1.754594332},
  R=1197.783445613274);

constant IdealGases.Common.DataRecord LiAlF4 (
  name="LiAlF4",
  MM=0.1099161508,
  Hf=-16897315.00313783,
  H0=177720.8249908984,
  Tlimit=1000,
  allow={176152.936,-2999.300134,20.7973957,-0.001742914656,-
4.266642090000001e-006,
  5.362829770000001e-009,-1.866817087e-012},
  blow={-211794.7828,-87.6272589},
  ahigh={-402070.0930000001,-278.3355146,16.20791185,-8.316341570000001e-005,
  1.83690569e-008,-2.104932808e-012,9.74725302e-017},
  bhigh={-227869.6416,-55.9730484},
  R=75.64376972342085);

constant IdealGases.Common.DataRecord LiBO2 (
  name="LiBO2",
  MM=0.0497508,
  Hf=-13112397.71018758,
```

```

H0=269899.6599049664,
Tlimit=1000,
alow={65189.8757,-1060.320845,9.54804996,-5.46957607e-005,2.658687599e-006,
      -2.931603694e-009,9.66302973e-013},
blow={-75062.0411,-24.90883998},
ahigh={85544.04520000001,-1731.920898,11.2146209,-0.000465916217,
       9.999875689999999e-008,-1.122880592e-011,5.12352946e-016},
bhigh={-71547.4382,-36.1239729},
R=167.1223779316112);

constant IdealGases.Common.DataRecord LiBr(
  name="LiBr",
  MM=0.08684500000000001,
  Hf=-1740605.561632794,
  H0=105626.2306407968,
  Tlimit=1000,
  alow={38056.2047,-612.961499,6.57736955,-0.00421989176,
        5.391423370000001e-006,-3.69314481e-009,1.049158761e-012},
  blow={-16374.87617,-11.28654012},
  ahigh={63801.9142,-259.7050764,4.65676887,0.0001022867706,-
5.178252340000001e-008,
        2.058203991e-011,-1.942190308e-015},
  bhigh={-17933.63752,-0.2290806613},
  R=95.73921354136679);

constant IdealGases.Common.DataRecord LiCL(
  name="LiCL",
  MM=0.042394,
  Hf=-4570927.277444921,
  H0=213712.0583101382,
  Tlimit=1000,
  alow={49643.995,-718.734662,6.78514703,-0.00452214546,
        5.74848370000001e-006,-3.96625567e-009,1.137911398e-012},
  blow={-20910.14703,-14.06383901},
  ahigh={-235276.9705,612.004692,3.63429373,0.0006810973320000001,-
2.174238799e-007,
        4.22040361e-011,-2.848628426e-015},
  bhigh={-28623.58494,5.618511135},
  R=196.1237911025145);

constant IdealGases.Common.DataRecord LiF(
  name="LiF",
  MM=0.0259394032,
  Hf=-13143898.93133702,
  H0=340335.7406464926,
  Tlimit=1000,
  alow={29125.3732,-253.1413159,3.53972798,0.00369591704,-4.82946615e-006,
        2.944090711e-009,-6.83879474e-013},
  blow={-40648.4966,2.325294408},
  ahigh={-378424.649,766.806246,3.5854734,0.0006031084350000001,-1.588764206e-
007,
        2.569397177e-011,-1.406798386e-015},
  bhigh={-47564.6945,4.38516329},
  R=320.5344369680795);

constant IdealGases.Common.DataRecord LiH(
  name="LiH",

```

## 1110 Modelica.Media.IdealGases.Common.SingleGasesData

---

```
MM=0.00794894,
Hf=17519833.33626873,
H0=1092737.396432732,
Tlimit=1000,
alow={-49137.31570000001, 775.6092190000001, -1.011102377, 0.01145479597, --
1.151038734e-005,
      5.87506896e-009, -1.196789735e-012},
blow={12048.5891, 25.68801877},
ahigh={-2633686.357, 6996.429169999999, -3.23353306, 0.00403393598, --
9.09957964e-007,
      8.775909869999999e-011, -2.889490251e-015},
bhigh={-29900.43016, 49.71984499999999},
R=1045.984999257763);
```

```
constant IdealGases.Common.DataRecord Lii(
  name="Lii",
  MM=0.13384547,
  Hf=-637077.1308136167,
  H0=69409.2373839772,
  Tlimit=1000,
  alow={40719.7637, -666.7289470000001, 7.06425753, -0.00548004102,
        6.90579016e-006, -4.53716852e-009, 1.226373296e-012},
  blow={-8235.487020000001, -12.96236234},
  ahigh={1616342.632, -4877.71708, 9.987186550000001, -0.002889137815,
        8.03731377e-007, -9.028172610000001e-011, 3.078321954e-015},
  bhigh={19377.18764, -37.2927631},
  R=62.11993577369485);
```

```
constant IdealGases.Common.DataRecord Lin(
  name="Lin",
  MM=0.0209477,
  Hf=15978842.5459597,
  H0=429600.7676260401,
  Tlimit=1000,
  alow={37649.592, -488.345763, 5.18737345, 0.0002275252171, -1.416337564e-006,
        1.450236258e-009, -4.79001433e-013},
  blow={41619.1531, -5.952157244},
  ahigh={-59934.8998, -40.3113455, 4.52962421, 8.744156930000001e-005,
        2.557428364e-009, -2.9057518e-013, 1.336125922e-017},
  bhigh={38948.3113, -1.214952896},
  R=396.9157473135476);
```

```
constant IdealGases.Common.DataRecord LiNO2 (
  name="LiNO2",
  MM=0.0529465,
  Hf=-3815761.343998187,
  H0=252496.2367673028,
  Tlimit=1000,
  alow={-31337.73859, 538.994080999999, -0.02059877081, 0.02610777454, --
3.20923504e-005,
        1.974344044e-008, -4.89219916e-012},
  blow={-28382.14963, 27.10019529},
  ahigh={-219667.116, -861.004271, 10.63956465, -0.0002554392587, 5.64447854e-008,
        -6.47614005e-012, 3.003536591e-016},
  bhigh={-23135.47535, -32.38277103},
  R=157.0353470012182);
```

```

constant IdealGases.Common.DataRecord LiNO3 (
  name="LiNO3",
  MM=0.0689458999999999,
  Hf=-4519263.291943394,
  H0=201829.6229362443,
  Tlimit=1000,
  allow={19672.05518, 90.52795259999999, -0.51097704, 0.0375660595, -4.78273865e-
005,
         3.029277805e-008, -7.70801315e-012},
  blow={-39075.6399, 27.46873033},
  ahighe={-355940.245, -1447.644737, 14.06684061, -0.000423637128,
          9.32082450999999e-008, -1.065894872e-011, 4.93078062e-016},
  bhigh={-34348.2746, -52.75382697000001},
  R=120.5941470051156);

constant IdealGases.Common.DataRecord LiO(
  name="LiO",
  MM=0.0229404,
  Hf=3178423.785112727,
  H0=408105.9179438893,
  Tlimit=1000,
  allow={36270.2976, -349.936323, 4.39493318, 0.001079712984, -8.72403881e-007,
         5.60796297e-010, -2.05428457e-013},
  blow={9533.330530000001, -0.905814960999999},
  ahighe={1612392.133, -5551.31234, 11.20573851, -0.00343722688,
          9.13319465999999e-007, -1.027902258e-010, 3.822991e-015},
  bhigh={42015.7547, -48.5735458},
  R=362.4379696953846);

constant IdealGases.Common.DataRecord LiOF(
  name="LiOF",
  MM=0.0419388032,
  Hf=-2194817.042370918,
  H0=258170.5288147088,
  Tlimit=1000,
  allow={55101.10690000001, -577.068817, 4.42495493, 0.01102425422, -1.710493381e-
005,
         1.234361201e-008, -3.453693e-012},
  blow={-9278.986980000002, 0.1238830687},
  ahighe={-183356.3281, -211.0049301, 7.15681817, -6.24946201e-005,
          1.376567002e-008, -1.574099067e-012, 7.277167140000001e-017},
  bhigh={-12545.30277, -12.72130479},
  R=198.2524861367527);

constant IdealGases.Common.DataRecord LiOH(
  name="LiOH",
  MM=0.02394834,
  Hf=-9562249.408518503,
  H0=473392.3520377612,
  Tlimit=1000,
  allow={4574.19012, -103.0949027, 4.27240737, 0.008465219230000001, -1.386148524e-
005,
         1.101099795e-008, -3.29124821e-012},
  blow={-28487.28855, -0.877574577999999},
  ahighe={850075.137, -2430.540791, 8.055314620000001, 6.895680879999999e-005, -
5.52720745999999e-008,
          9.36805402999999e-012, -5.31378568e-016},
  bhigh={-13658.94396, -24.57598093},

```

## 1112 Modelica.Media.IdealGases.Common.SingleGasesData

---

```
R=347.1836461316316);

constant IdealGases.Common.DataRecord LiON(
  name="LiON",
  MM=0.0369471,
  Hf=4869448.48174823,
  H0=305869.1751179389,
  Tlimit=1000,
  allow={-9412.67267,100.93323,3.006187607,0.0098904622,-1.128294343e-005,
    6.41029174e-009,-1.466806373e-012},
  blow={19783.48318,10.16334813},
  ahigh={-97406.2913999999,-511.1560940000001,7.37849387,-0.0001508777884,
    3.32991498e-008,-3.81751429e-012,1.769558455e-016},
  bhigh={22110.06635,-14.5410504},
  R=225.0372018372214);

constant IdealGases.Common.DataRecord Li2(
  name="Li2",
  MM=0.013882,
  Hf=15552514.04696729,
  H0=696954.0412044374,
  Tlimit=1000,
  allow={6778.481580000001,-224.6205832,5.29603744,-0.001272412017,
    1.205843729e-006,-9.81854459e-011,-2.416702607e-013},
  blow={25736.38186,-6.86925758},
  ahigh={37676454,-118574.7185,148.2167789,-0.0837853211,2.424919798e-005,-
    3.27582024e-009,
    1.652000081e-013},
  bhigh={772307.201,-1021.697298},
  R=598.9390577726553);

constant IdealGases.Common.DataRecord Li2plus(
  name="Li2plus",
  MM=0.0138814514,
  Hf=51983859.55520473,
  H0=719601.9862879757,
  Tlimit=1000,
  allow={-10400.56453,26.75405642,4.24727175,0.001057450777,-1.281715096e-006,
    1.067477239e-009,-2.759413508e-013},
  blow={85298.1130999999,0.529818996},
  ahigh={12799310.73,-34928.6139,38.4478536,-0.01380635397,2.53139425e-006,-
    2.197815991e-010,
    7.08766925e-015},
  bhigh={311796.445,-250.9543641},
  R=598.9627280617069);

constant IdealGases.Common.DataRecord Li2Br2(
  name="Li2Br2",
  MM=0.17369,
  Hf=-2854703.920778398,
  H0=97590.39668374689,
  Tlimit=1000,
  allow={27863.12863,-1005.641437,13.70963066,-0.0076336633,8.94290592e-006,-
    5.56113509e-009,
    1.421736784e-012},
  blow={-57628.4584,-41.2184376},
  ahigh={-89009.8336,-19.88916335,10.01596418,-6.74913405e-006,
```

```

  1.556299963e-009,-1.844693069e-013,8.774879919999999e-018},
bhigh={-62799.8996,-19.43063525},
R=47.8696067706834);

constant IdealGases.Common.DataRecord Li2F2(
  name="Li2F2",
  MM=0.05187880640000001,
  Hf=-18029001.4536649,
  H0=265349.7826040963,
  Tlimit=1000,
  allow={144316.6185,-2466.874678,17.0286329,-0.0114554121,1.086098792e-005,-
5.56975427999999e-009,
  1.193819664e-012},
  blow={-102607.0133,-70.0038661},
  ahigh={-218893.1683,-92.5981453,10.06970757,-2.803632163e-005,
  6.217549520000001e-009,-7.14626702e-013,3.3168573e-017},
  bhigh={-115661.1527,-27.29853181},
  R=160.2672184840397);

constant IdealGases.Common.DataRecord Li2I2(
  name="Li2I2",
  MM=0.26769094,
  Hf=-1355299.320178711,
  H0=65966.56577170672,
  Tlimit=1000,
  allow={10148.72266,-712.042101,12.7039296,-0.00568553681,
  6.770340380000001e-006,-4.26330489e-009,1.100658186e-012},
  blow={-43130.6386,-33.03950980000001},
  ahigh={-69735.8706,-13.16825508,10.01074855,-4.60020378e-006,
  1.070510527e-009,-1.277717768e-013,6.11061224e-018},
  bhigh={-46774.7005,-17.21894034},
  R=31.05996788684742);

constant IdealGases.Common.DataRecord Li2O(
  name="Li2O",
  MM=0.0298814,
  Hf=-5600103.074153152,
  H0=428095.0691734658,
  Tlimit=1000,
  allow={26366.01597,-179.8629279,4.05111522,0.01189981375,-1.730095793e-005,
  1.20558455e-008,-3.29273388e-012},
  blow={-20619.07963,1.60599561},
  ahigh={726148.703999999,-9543.783720000001,28.87491643,-0.01959494099,
  8.086339840000001e-006,-1.370211764e-009,8.11172719e-014},
  bhigh={29907.26399,-156.4517822},
  R=278.2490780217794);

constant IdealGases.Common.DataRecord Li2Oplus(
  name="Li2Oplus",
  MM=0.0298808514,
  Hf=14694865.38793871,
  H0=436021.3444252797,
  Tlimit=1000,
  allow={108104.4623,-1261.961355,9.69201129,-0.001793477428,3.3803269e-007,
  4.14339515e-010,-1.990530222e-013},
  blow={57549.7604,-29.15719946},
  ahigh={-130216.6086,-146.6759528,7.61072945,-4.46942669e-005,

```

---

**1114 Modelica.Media.IdealGases.Common.SingleGasesData**

---

```
9.947160640000001e-009,-1.147038806e-012,5.33924147e-017},  
bhigh={50987.6231,-15.28011047},  
R=278.2541865590885);  
  
constant IdealGases.Common.DataRecord Li2O2 (  
  name="Li2O2",  
  MM=0.0458808,  
  Hf=-6089649.439416925,  
  H0=295287.1789506722,  
  Tlimit=1000,  
  allow={139261.973,-1764.036747,10.29649198,0.00760594995,-1.538519312e-005,  
         1.238199941e-008,-3.68543157e-012},  
  blow={-26380.56746,-34.3771056},  
  ahigh={-293923.2153,-235.9107937,10.17463402,-6.931624359999999e-005,  
         1.521202834e-008,-1.733841909e-012,7.99315462e-017},  
  bhigh={-36184.8191,-29.04221044},  
  R=181.2189848476923);  
  
constant IdealGases.Common.DataRecord Li2O2H2 (  
  name="Li2O2H2",  
  MM=0.04789668,  
  Hf=-15387287.80366405,  
  H0=325263.4629373059,  
  Tlimit=1000,  
  allow={189441.3579,-2779.117524,15.30234911,0.01070738317,-2.510075145e-005,  
         2.224910575e-008,-6.94935958e-012},  
  blow={-77027.7692,-65.1411694},  
  ahigh={1215377.492,-3982.05744,15.85877373,0.000750914568,-2.585218184e-007,  
         3.6468417e-011,-1.904139406e-015},  
  bhigh={-67387.4838,-66.472084},  
  R=173.5918230658158);  
  
constant IdealGases.Common.DataRecord Li2SO4 (  
  name="Li2SO4",  
  MM=0.1099446,  
  Hf=-9475826.916465202,  
  H0=178749.4610922228,  
  Tlimit=1000,  
  allow={100631.8702,-1543.457217,9.67005490999999,0.0339075738,-4.82348381e-  
005,  
         3.26736892e-008,-8.702518510000001e-012},  
  blow={-120193.9538,-29.11716635},  
  ahigh={-603875.0870000001,-1064.550976,19.79557349,-0.000318897825,  
         7.062411120000001e-008,-8.11443907e-012,3.76683021e-016},  
  bhigh={-126890.8595,-80.14316674},  
  R=75.62419618607918);  
  
constant IdealGases.Common.DataRecord Li3plus (  
  name="Li3plus",  
  MM=0.0208224514,  
  Hf=36335339.26749855,  
  H0=638224.9978501572,  
  Tlimit=1000,  
  allow={-6873.222290000001,-282.0780467,7.92269748,-0.001687949474,  
         1.771718993e-006,-9.972535399999999e-010,2.332934849e-013},  
  blow={90279.63650000001,-16.19319856},  
  ahigh={-43550.9806,-6.05643275,7.00472289,-1.953320933e-006,
```

```

  4.42915222999999e-010,-5.18198319e-014,2.439894871e-018},
bhigh={88799.02870000001,-10.66880792},
R=399.303225171653);

constant IdealGases.Common.DataRecord Li3Br3(
  name="Li3Br3",
  MM=0.260535,
  Hf=-3165176.245034256,
  H0=99935.92799431937,
  Tlimit=1000,
  allow={64319.66250000001,-1823.743601,22.43492856,-0.0127941826,
    1.459200355e-005,-8.884688150000001e-009,2.23370805e-012},
  blow={-94806.7767,-82.06864420000001},
  ahigh={-158467.0201,-40.4136094,16.03177633,-1.322713911e-005,
    3.014221791e-009,-3.54029216e-013,1.672030783e-017},
  bhigh={-104247.0448,-44.05478160000001},
  R=31.9130711804556);

constant IdealGases.Common.DataRecord Li3CL3(
  name="Li3CL3",
  MM=0.127182,
  Hf=-7674884.189586578,
  H0=193113.3808243305,
  Tlimit=1000,
  allow={98674.28999999999,-2324.074851,23.78484564,-0.014835224,
    1.634495612e-005,-9.675366520000001e-009,2.377033859e-012},
  blow={-110383.9177,-94.74114379},
  ahigh={-200593.0356,-59.3940019,16.04590929,-1.885890854e-005,
    4.25351737e-009,-4.95556094e-013,2.325408214e-017},
  bhigh={-122492.7071,-48.44413629},
  R=65.37459703417152);

constant IdealGases.Common.DataRecord Li3F3(
  name="Li3F3",
  MM=0.07781820960000001,
  Hf=-19591775.93312298,
  H0=263906.5342875737,
  Tlimit=1000,
  allow={182387.4443,-3163.78624,22.87811928,-0.007544903889999999,
    3.44679023e-006,2.690705228e-010,-5.25084096e-013},
  blow={-171244.6093,-99.7365499},
  ahigh={-361866.499,-193.7051395,16.14457133,-5.7756285e-005,
    1.274057038e-008,-1.458168875e-012,6.74485003000001e-017},
  bhigh={-188208.1974,-56.3361288},
  R=106.8448123226932);

constant IdealGases.Common.DataRecord Li3I3(
  name="Li3I3",
  MM=0.40153641,
  Hf=-1525284.611176356,
  H0=67821.2319525395,
  Tlimit=1000,
  allow={36117.6469,-1392.033487,21.10791892,-0.01046866587,1.222589173e-005,-
    7.58413543e-009,
    1.93519918e-012},
  blow={-71530.8458,-70.9388203},
  ahigh={-126669.6511,-27.83891914,16.02228068,-9.399364930000001e-006,
    1.222589173e-005}
);

```

---

**1116 Modelica.Media.IdealGases.Common.SingleGasesData**

---

```
 2.163919735e-009,-2.561731922e-013,1.217394107e-017},  
bhigh={-78695.5068,-40.9172406},  
R=20.70664525789828);  
  
constant IdealGases.Common.DataRecord Mg(  
  name="Mg",  
  MM=0.024305,  
  Hf=6052252.622917095,  
  H0=254985.7231022423,  
  Tlimit=1000,  
  allow={0,0,2.5,0,0,0},  
  blow={16946.58761,3.63433014},  
  ahigh={-536483.155,1973.709576,-0.36337769,0.002071795561,-  
7.738051719999999e-007,  
  1.359277788e-010,-7.766898397000001e-015},  
  bhigh={4829.188109999999,23.39104998},  
  R=342.0889528903518);  
  
constant IdealGases.Common.DataRecord Mgplus(  
  name="Mgplus",  
  MM=0.0243044514,  
  Hf=36661884.90886899,  
  H0=254991.4786391763,  
  Tlimit=1000,  
  allow={0,0,2.5,0,0,0},  
  blow={106422.3354,4.32744346},  
  ahight={-19147.58821,48.7734792,2.457662661,1.218104674e-005,  
  1.897261686e-009,-1.580433756e-012,2.135732238e-016},  
  bhigh={106102.2394,4.64644286},  
  R=342.0966745211126);  
  
constant IdealGases.Common.DataRecord MgBr(  
  name="MgBr",  
  MM=0.104209,  
  Hf=59143.78796457119,  
  H0=92008.51174082852,  
  Tlimit=1000,  
  allow={7361.41914,-239.5789881,5.36056042,-0.001667141829,1.981137765e-006,-  
1.201637202e-009,  
  3.032148099e-013},  
  blow={591.563444,-1.421771179},  
  ahight={24776.04216,-641.7687520000001,6.01993209,-0.001391004302,  
  6.44333096e-007,-1.197734078e-010,7.42164424000001e-015},  
  bhigh={2824.060334,-6.26443992},  
  R=79.78650596397624);  
  
constant IdealGases.Common.DataRecord MgBr2(  
  name="MgBr2",  
  MM=0.184113,  
  Hf=-1666059.534090477,  
  H0=80153.38949449523,  
  Tlimit=1000,  
  allow={21484.77999,-515.45289,9.27325875,-0.00347004904,3.92184079e-006,-  
2.377174864e-009,  
  5.96702559999999e-013},  
  blow={-36524.4385,-17.91077763},  
  ahight={-43268.9901,-12.83036001,7.5100243,-4.1523324e-006,9.42649918e-010,-
```

```

1.103873928e-013,
      5.20110431e-018},
bhigh={-39198.8602,-7.40916803},
R=45.15961393274782);

constant IdealGases.Common.DataRecord MgCL(
  name="MgCL",
  MM=0.05975800000000001,
  Hf=-915440.3929181029,
  H0=156683.8414940259,
  Tlimit=1000,
  allow={20439.9528,-407.215516,5.8803723,-0.002594175042,2.945953528e-006,-
1.750648628e-009,
        4.33795933000001e-013},
  blow={-5851.44495,-6.02354575},
  ahighe={1041328.453,-3380.15833,8.63777546999999,-0.002447789643,
    7.84196944e-007,-1.12640938e-010,5.81062073e-015},
  bhigh={13271.88977,-27.03802395},
  R=139.1357140466548);

constant IdealGases.Common.DataRecord MgCLplus(
  name="MgCLplus",
  MM=0.0597574514,
  Hf=10816045.81282394,
  H0=159239.2543032717,
  Tlimit=1000,
  allow={8182.385119999999,-262.225376,5.41986343,-0.001774129606,
    2.185811127e-006,-1.418143432e-009,3.91889858e-013},
  blow={77704.0371,-3.78093884},
  ahighe={-12683919.21,34788.2454,-30.0422295,0.01481739497,-2.470965605e-006,
    1.424433718e-010,2.789613105e-016},
  bhigh={-148701.374,255.2015117},
  R=139.1369913744347);

constant IdealGases.Common.DataRecord MgCL2(
  name="MgCL2",
  MM=0.095211,
  Hf=-4192476.667611936,
  H0=145997.8783964038,
  Tlimit=1000,
  allow={36378.2468,-730.784496,9.66103051,-0.00366021294,3.61081935e-006,-
1.928769286e-009,
        4.30991235000001e-013},
  blow={-46469.1457,-23.60112274},
  ahighe={-68352.1701,-24.90899393,7.5187832,-7.56451826e-006,1.67929377e-009,
    -1.931706275e-013,8.971657820000001e-018},
  bhigh={-50326.8691,-10.53268382},
  R=87.32680047473507);

constant IdealGases.Common.DataRecord MgF(
  name="MgF",
  MM=0.0433034032,
  Hf=-5363707.580377886,
  H0=207122.473921403,
  Tlimit=1000,
  allow={38230.0162,-480.331039,5.06846894,0.00052120293,-1.874026347e-006,
    1.759241129e-009,-5.60940319e-013},

```

---

**1118 Modelica.Media.IdealGases.Common.SingleGasesData**

---

```
blow={-26591.14296,-3.76896921},
ahigh={-169588.3782,358.875763,3.93600165,0.000481304414,-1.658385409e-007,
      3.33605229e-011,-2.270104205e-015},
bhigh={-31693.2399,4.4105659},
R=192.0050477695481);

constant IdealGases.Common.DataRecord MgFplus(
  name="MgFplus",
  MM=0.0433028546,
  Hf=11936113.21411591,
  H0=207131.2638128019,
  Tlimit=1000,
  allow={641329.384,-8518.53261,48.1421691,-0.1163071535,0.0001622560562,-
1.06348459e-007,
        2.634809398e-011},
  blow={102430.848,-245.0074295},
  ahigh={-10568523.62,20771.22379,-6.11952356,0.002245834831,-9.51573861e-008,
        -2.263502341e-011,2.284524246e-015},
  bhigh={-83315.4988,87.2710238},
  R=192.0074802643611);

constant IdealGases.Common.DataRecord MgF2 (
  name="MgF2",
  MM=0.0623018064,
  Hf=-11805405.05162624,
  H0=202601.5091594519,
  Tlimit=1000,
  allow={43384.2955,-661.651177,7.45344852,0.00352081405,-6.95576458e-006,
        5.61992348e-009,-1.688028906e-012},
  blow={-86871.8367,-15.4547679},
  ahigh={-124441.9584,-86.8734374999999,7.56358123,-2.499267978e-005,
        5.44012914e-009,-6.15824576e-013,2.822792666e-017},
  bhigh={-90600.1943999999,-14.2079699},
  R=133.4547500375527);

constant IdealGases.Common.DataRecord MgF2plus(
  name="MgF2plus",
  MM=0.0623012578,
  Hf=9352815.409771709,
  H0=199274.6926531554,
  Tlimit=1000,
  allow={78322.2026,-1176.632752,10.25829767,-0.00375527289,3.016621585e-006,-
1.327098871e-009,
        2.465289749e-013},
  blow={74132.2855,-29.90883551},
  ahigh={-150231.687,77.2528956,7.27156936,0.000223752977,-1.003070865e-007,
        1.996274309e-011,-1.233278721e-015},
  bhigh={66991.175,-10.94528006},
  R=133.4559251867945);

constant IdealGases.Common.DataRecord MgH (
  name="MgH",
  MM=0.02531294,
  Hf=9077811.743716849,
  H0=342990.7391239421,
  Tlimit=1000,
  allow={-49586.7915,750.027865,-0.64420475,0.00982630101,-8.789822439999999e-
```

```

006,
      3.82335352e-009,-6.00372576e-013},
blow={23022.79383,26.57165344},
ahigh={-100574.8598,1952.890106,-1.317191549,0.0056036658,-2.13733498e-006,
      3.3248805e-010,-1.824672746e-014},
bhigh={15985.82755,34.3123316},
R=328.4672582481529);

constant IdealGases.Common.DataRecord MgI (
  name="MgI",
  MM=0.15120947,
  Hf=404778.3316745968,
  H0=64421.34212890238,
  Tlimit=1000,
  allow={2943.889099,-169.0248574,5.14725183,-0.001321186997,1.623056505e-006,
      -1.005114222e-009,2.567337785e-013},
  blow={6845.89027,0.859313225},
  ahigh={-2370562.811,6916.45248,-3.18389449,0.00405155114,-
9.774290750000001e-007,
      1.0233294e-010,-3.79030487e-015},
  bhigh={-38185.5111,59.9310438},
  R=54.98645025341336);

constant IdealGases.Common.DataRecord MgI2 (
  name="MgI2",
  MM=0.27811394,
  Hf=-617394.5434018878,
  H0=54993.54329380253,
  Tlimit=1000,
  allow={15947.39709,-416.041272,9.01063832,-0.003090359024,3.62126426e-006,-
2.260138392e-009,
      5.80905555e-013},
  blow={-20804.414,-14.12175736},
  ahigh={-33402.9918,-9.34351178,7.50746854,-3.14770789e-006,
      7.241336470000001e-010,-8.56766495e-014,4.0696917e-018},
  bhigh={-22945.55505,-5.24127259},
  R=29.89591963639075);

constant IdealGases.Common.DataRecord MgN (
  name="MgN",
  MM=0.0383117,
  Hf=7535452.616302591,
  H0=234624.9579110298,
  Tlimit=1000,
  allow={37595.3864,-485.316428,5.16305424,0.0002591539493,-1.518993975e-006,
      1.522840476e-009,-4.992531450000001e-013},
  blow={36072.9461,-3.813908776},
  ahigh={-60185.891,-40.1086478,4.52949895,4.76049091e-005,2.547271298e-009,-
2.894058118e-013,
      1.330639131e-017},
  bhigh={33412.857,0.7925250209},
  R=217.0217453153997);

constant IdealGases.Common.DataRecord MgO (
  name="MgO",
  MM=0.0403044,
  Hf=800441.3165808199,

```

---

**1120 Modelica.Media.IdealGases.Common.SingleGasesData**

---

```
H0=221045.5186034279,
Tlimit=1000,
alow={351365.974,-5287.19716,33.8206006,-0.0840048962999999,0.000121001616,
       -7.630795020000001e-008,1.701022862e-011},
blow={27906.79519,-162.4886199},
ahigh={-15867383.67,34204.681,-17.74087677,0.00700496305,-1.104138249e-006,
       8.957488529999999e-011,-3.052513649e-015},
bhigh={-230050.4434,173.8984472},
R=206.2919185002134);

constant IdealGases.Common.DataRecord MgOH(
  name="MgOH",
  MM=0.04131234,
  Hf=-3205555.047232862,
  H0=269272.6676823439,
  Tlimit=1000,
  alow={38398.5162,-736.7383640000001,7.92066446,-0.000595094059,
         -2.112941162e-006,
         3.22828211e-009,-1.214159329e-012},
  blow={-13923.26188,-19.16078109},
  ahigh={664866.475,-1770.750355,7.26999927,0.000533684276,-1.980894443e-007,
         3.025677088e-011,-1.554849476e-015},
  bhigh={-6149.11456,-16.71027009},
  R=201.2588006392279);

constant IdealGases.Common.DataRecord MgOHplus(
  name="MgOHplus",
  MM=0.0413117914,
  Hf=14905414.38975217,
  H0=246621.3556645719,
  Tlimit=1000,
  alow={117022.4573,-1735.933343,11.64059613,-0.00844987621999999,
         7.35171374e-006,-2.790223071e-009,3.51498213e-013},
  blow={81188.07670000001,-42.7117437},
  ahigh={829633.954,-2459.700177,8.11873202,3.5005791e-005,-4.67057475e-008,
         8.312358260000001e-012,-4.80283622e-016},
  bhigh={88016.61709999999,-24.38155217},
  R=201.261473255793);

constant IdealGases.Common.DataRecord Mg_OH_2(
  name="Mg_OH_2",
  MM=0.05831968,
  Hf=-9465000.631004833,
  H0=293752.6063243146,
  Tlimit=1000,
  alow={52458.9467,-1289.056383,13.89327642,-0.000780669367,-4.15125723e-006,
         6.10947304e-009,-2.274138833e-012},
  blow={-62950.8915,-50.1535334},
  ahigh={1713709.254,-4730.00535,14.48925967,0.0001907819857,-1.226834131e-
007,
         2.015343753e-011,-1.128993279e-015},
  bhigh={-38877.2467,-58.4049812},
  R=142.5671745798331);

constant IdealGases.Common.DataRecord MgS(
  name="MgS",
  MM=0.05637,
```

```

Hf=2140310.643959553,
H0=163812.5066524748,
Tlimit=1000,
alow={-9565.78809,144.3637798,1.813794717,0.01147168775,-2.220170412e-005,
      1.995344981e-008,-6.09068874e-012},
blow={12765.01517,14.61333093},
ahigh={26507943.28,-77113.5586,84.63771680000001,-0.0364425068,
       8.403084420000002e-006,-9.53988217e-010,4.264658029999999e-014},
bhigh={507893.117,-583.4656096},
R=147.4981727869434);

constant IdealGases.Common.DataRecord Mg2 (
  name="Mg2",
  MM=0.04861,
  Hf=5894122.917095248,
  H0=196299.423986834,
  Tlimit=1000,
  alow={4545.195589999999,411.585004,0.484119617,0.00489196965,-6.39553684e-
006,
        4.29976455e-009,-1.164624418e-012},
  blow={31816.4179,26.40432143},
  ahigh={30382.24994,59.4524046,2.352706666,0.0001378537924,-5.89569204e-008,
         1.104045317e-011,-6.558868290000001e-016},
  bhigh={33510.3656,15.88177377},
  R=171.0444764451759);

constant IdealGases.Common.DataRecord Mg2F4 (
  name="Mg2F4",
  MM=0.1246036128,
  Hf=-13790683.60367798,
  H0=169684.5984228156,
  Tlimit=1000,
  alow={151195.6137,-3122.595912,25.17715088,-0.01558723757,1.552195619e-005,
        -8.40872905e-009,1.911618525e-012},
  blow={-195307.8883,-108.5103537},
  ahigh={-298061.0166,-120.1299868,16.09110864,-3.68745629e-005,
         8.22031705e-009,-9.48886978e-013,4.41979533e-017},
  bhigh={-211734.969,-53.08655469999999},
  R=66.72737501877634);

constant IdealGases.Common.DataRecord Mn (
  name="Mn",
  MM=0.054938049,
  Hf=5140335.434918703,
  H0=112807.5734906421,
  Tlimit=1000,
  alow={0.1034061359,-0.001551537349,2.500009148,-2.723162066e-008,
        4.33389743e-011,-3.51109389e-014,1.136032201e-017},
  blow={33219.3519,6.649325463},
  ahigh={5855.15582,883.858844000001,-0.0364866258,0.002703720687,-
1.324971998e-006,
        2.87260329e-010,-1.92363357e-014},
  bhigh={28678.03487,22.92541198},
  R=151.3426878337088);

constant IdealGases.Common.DataRecord Mnplus (
  name="Mnplus",

```

---

**1122 Modelica.Media.IdealGases.Common.SingleGasesData**

---

```
MM=0.0549375004,
Hf=18309375.57544937,
H0=112808.6999749992,
Tlimit=1000,
alow={345.80177,-4.25115133,2.521281028,-5.56508728e-005,8.03716221e-008,-
6.09355097e-011,
1.900014268e-014},
blow={120253.3602,6.683468162},
ahigh={647131.41,-2403.796253,5.93771575,-0.002341014594,7.46416564e-007,-
9.075969730000001e-011,
4.467879847e-015},
bhigh={134990.2108,-17.02666341},
R=151.344199125594);
```

```
constant IdealGases.Common.DataRecord Mo (
  name="Mo",
  MM=0.09594,
  Hf=6863664.790494058,
  H0=64596.91473837815,
  Tlimit=1000,
  alow={76.46367910000001,-1.159269043,2.506929462,-2.099249725e-005,
  3.41477943e-008,-2.841269591e-011,9.492443320999999e-015},
  blow={78458.99799999999,7.60183566},
  ahigh={5573271,-16623.65811,21.35147077,-0.01003069377,2.409784357e-006,-
1.811267352e-010,
  1.034189087e-015},
  bhigh={184264.6473,-127.5326434},
  R=86.66324786324788);
```

```
constant IdealGases.Common.DataRecord Moplus (
  name="Moplus",
  MM=0.0959394514,
  Hf=14061086.53233346,
  H0=64597.28411580224,
  Tlimit=1000,
  alow={129.8236623,-1.560279908,2.507600281,-1.923789063e-005,
  2.673316651e-008,-1.937174292e-011,5.729735412e-015},
  blow={161510.3759,7.44254346},
  ahigh={12988911.2,-39482.7623,48.6659978,-0.02605352326,7.21543192e-006,-
8.719164960000001e-010,
  3.78842304e-014},
  bhigh={411894.857,-321.679103},
  R=86.66374342015469);
```

```
constant IdealGases.Common.DataRecord Mominus (
  name="Mominus",
  MM=0.0959405486,
  Hf=6048794.138331621,
  H0=64596.54536517837,
  Tlimit=1000,
  alow={0,0,2.5,0,0,0,0},
  blow={69051.2369,7.48565954},
  ahigh={0,0,2.5,0,0,0,0},
  bhigh={69051.2369,7.48565954},
  R=86.66275231200837);
```

```
constant IdealGases.Common.DataRecord MoO (
```

```

name="MoO",
MM=0.1119394,
Hf=3198206.556404626,
H0=91595.30067161338,
Tlimit=1000,
alow={-28011.52706, 513.988348, 1.075385931, 0.00868104847000001, --
1.11118984e-005,
    7.23434933e-009, -1.893138381e-012},
blow={39413.7408, 22.72230239},
ahigh={1573131.992, -5241.48358, 11.02656868, -0.00390299662, 1.147334134e-006,
    -1.358975691e-010, 5.77526858e-015},
bhigh={74489.72, -42.5361293},
R=74.27654605974304);

constant IdealGases.Common.DataRecord MoO2 (
  name="MoO2",
  MM=0.1279388,
  Hf=-121605.6348816778,
  H0=83688.64644658228,
  Tlimit=1000,
  alow={32471.8322, -190.4783783, 2.120771647, 0.01650280086, -2.381696822e-005,
    1.652371586e-008, -4.49445243e-012},
  blow={-1862.932837, 16.40582056},
  ahigh={309614.3654, -1932.750274, 9.428673180000001, -0.001630508855,
    5.752760170000001e-007, -7.59045747e-011, 3.46133778e-015},
  bhigh={7327.72518, -25.33315948},
  R=64.98788483243551);

constant IdealGases.Common.DataRecord MoO3 (
  name="MoO3",
  MM=0.1439382,
  Hf=-2531727.99854382,
  H0=91655.89815629208,
  Tlimit=1000,
  alow={59773.8536, -768.455783, 5.88184444, 0.01686119817, -2.582485043e-005,
    1.850382718e-008, -5.15224985e-012},
  blow={-41558.7239, -6.52916216},
  ahigh={-409759.727, 237.9066513, 9.3111008, 0.000657933891, -2.895307725e-007,
    5.69263726e-011, -3.48965731e-015},
  bhigh={-49237.3872000001, -21.14864892},
  R=57.76417934919292);

constant IdealGases.Common.DataRecord MoO3minus (
  name="MoO3minus",
  MM=0.1439387486,
  Hf=-4552231.469101435,
  H0=94207.31479111943,
  Tlimit=1000,
  alow={182161.7352, -2224.65697, 13.20405967, -0.001249800352, -1.864166983e-006,
    2.27657947e-009, -7.366564570000001e-013},
  blow={-69389.86, -46.9937721},
  ahigh={-488109.872, 18.48536991, 10.42644941, -0.000626555791, 3.020554347e-007,
    -4.73319589e-011, 2.527517727e-015},
  bhigh={-83378.155, -27.01522825},
  R=57.763959190069);

constant IdealGases.Common.DataRecord Mo2O6 (

```

```
name="Mo206",
MM=0.2878764,
Hf=-3992848.364784332,
H0=89495.00202170099,
Tlimit=1000,
alow={156837.5811,-2159.930184,15.22500503,0.031941989,-5.06387715e-005,
      3.68937182e-008,-1.037872908e-011},
blow={-130993.1161,-54.089185},
ahigh={-631223.816,-664.282755,22.4938357,-0.0001968470329,4.33674061e-008,
       -4.959777990000001e-012,2.293212733e-016},
bhigh={-143057.325,-86.69628539999999},
R=28.88208967459646);

constant IdealGases.Common.DataRecord Mo309 (
  name="Mo309",
  MM=0.4318146,
  Hf=-4404740.712796649,
  H0=94448.21689678858,
  Tlimit=1000,
  alow={148338.6187,-1863.895133,17.38405871,0.0623301232,-9.32844135e-005,
        6.61363879e-008,-1.830607518e-011},
  blow={-224894.5631,-58.2666947},
  ahigh={-923029.054,-1076.171759,34.8000726,-0.00031896338,7.02839704e-008,
         -8.03964264e-012,
        3.71787953e-016},
  bhigh={-235742.2166,-144.8769485},
  R=19.25472644973097);

constant IdealGases.Common.DataRecord Mo4012 (
  name="Mo4012",
  MM=0.5757528,
  Hf=-4560163.273196414,
  H0=95342.7408429451,
  Tlimit=1000,
  alow={223487.6437,-2996.203728,26.82458827,0.0759734572,-0.0001155247012,
        8.25647257e-008,-2.296764551e-011},
  blow={-308461.9476,-106.5236386},
  ahigh={-1225283.388,-1368.141197,47.0168781,-0.00040529102,
        8.92847238999999e-008,-1.021091158e-011,4.721091689999999e-016},
  bhigh={-325647.777,-204.7486535},
  R=14.44104483729823);

constant IdealGases.Common.DataRecord Mo5015 (
  name="Mo5015",
  MM=0.7196910000000001,
  Hf=-4625746.934448255,
  H0=95759.62739564618,
  Tlimit=1000,
  alow={276890.3251,-3757.9331,34.0819776,0.0948541495999999,-0.00014426721,
        1.0311639e-007,-2.868600013e-011},
  blow={-391350.703,-142.6449143},
  ahigh={-1535389.516,-1710.367122,59.2712578,-0.0005066836290000001,
        1.116223706e-007,-1.276562233e-011,5.90231891e-016},
  bhigh={-412897.2380000001,-265.1287167},
  R=11.55283586983858);

constant IdealGases.Common.DataRecord N (
```

```

name="N",
MM=0.0140067,
Hf=33746706.93311058,
H0=442461.6790535958,
Tlimit=1000,
allow={0,0,2.5,0,0,0},
blow={56104.6378,4.193905036},
ahigh={88765.0138,-107.12315,2.362188287,0.0002916720081,-1.7295151e-007,
        4.01265788e-011,-2.677227571e-015},
bhigh={56973.5133,4.865231506},
R=593.6067739010616);

constant IdealGases.Common.DataRecord Nplus(
  name="Nplus",
  MM=0.0140061514,
  Hf=134378643.3723685,
  H0=508099.891023597,
  Tlimit=1000,
  allow={5237.07921,2.299958315,2.487488821,2.737490756e-005,-3.134447576e-008,
         1.850111332e-011,-4.447350984e-015},
  blow={225628.4738,5.076830786},
  ahigh={290497.0374,-855.7908610000001,3.47738929,-0.000528826719,
         1.352350307e-007,-1.389834122e-011,5.046166279e-016},
  bhigh={231080.9984,-1.994146545},
  R=593.6300245904811);

constant IdealGases.Common.DataRecord Nminus(
  name="Nminus",
  MM=0.0140072486,
  Hf=33806606.74502486,
  H0=463927.2983275246,
  Tlimit=1000,
  allow={1445.682471,7.33520511,2.476680939,4.22786918e-005,-4.42629332e-008,
         2.490985431e-011,-5.83160809e-015},
  blow={56176.25,5.145753977},
  ahigh={2404.189576,0.2954965336,2.499789368,8.30756497e-008,-1.82994277e-011,
         2.100136461e-015,-9.754986710000001e-020},
  bhigh={56214.13890000001,5.006484157},
  R=593.583525032889);

constant IdealGases.Common.DataRecord NCO(
  name="NCO",
  MM=0.04201680000000001,
  Hf=3137964.837874374,
  H0=242718.2698349232,
  Tlimit=1000,
  allow={11365.03036,-244.4613367,4.6713761,0.002309387548,2.798649599e-006,-4.54635738e-009,
         1.692880931e-012},
  blow={15776.49188,-0.2171476903},
  ahigh={108944.5289,-1735.459316,8.65561033,-0.000405322926,7.59971641e-008,
         -7.25380415e-012,3.24487241e-016},
  bhigh={23657.92776,-26.1953297},
  R=197.8844652615144);

constant IdealGases.Common.DataRecord ND (

```

---

**1126 Modelica.Media.IdealGases.Common.SingleGasesData**

---

```
name="ND",
MM=0.016020802,
Hf=22204805.78937309,
H0=539798.1948718922,
Tlimit=1000,
allow={22901.55757,-395.738851,6.17901033,-0.00884780697,1.44158297e-005,-
1.006647227e-008,
2.654403586e-012},
blow={43559.13430000001,-11.80414072},
ahigh={543965.796,-2084.583507,5.83408972,-0.000418939336,
9.768929529999999e-008,-1.056317654e-011,4.68307921e-016},
bhigh={54666.3341,-14.80810939},
R=518.9797614376608);

constant IdealGases.Common.DataRecord ND2 (
name="ND2",
MM=0.018034904,
Hf=10248824.72343629,
H0=552391.3517920583,
Tlimit=1000,
allow={19352.22164,-213.0631713,4.8440176,-0.002516949288,7.61638154e-006,-
5.46450177e-009,
1.292736999e-012},
blow={22119.98374,-3.171712417},
ahigh={1631308.357,-6564.63749,12.80547961,-0.003094456779,
9.181660230000001e-007,-1.241119284e-010,6.22629986e-015},
bhigh={61089.2958,-61.24502693},
R=461.0211398962811);

constant IdealGases.Common.DataRecord ND3 (
name="ND3",
MM=0.020049006,
Hf=-2730913.742057836,
H0=510449.2462120068,
Tlimit=1000,
allow={10451.2037,161.0166943,0.857496323,0.01319688794,-1.153090144e-005,
7.14249556e-009,-2.109194351e-012},
blow={-8220.948899999999,16.75921299},
ahigh={2599516.958,-10134.20124,17.98028169,-0.0035826098,1.009922e-006,-
1.537638609e-010,
9.106175650000001e-015},
bhigh={53972.0566,-98.10988569},
R=414.7074423540001);

constant IdealGases.Common.DataRecord NF (
name="NF",
MM=0.0330051032,
Hf=7059226.525914937,
H0=264750.1190058391,
Tlimit=1000,
allow={-35049.2775,667.450299,-1.201665982,0.01452074253,-1.822873148e-005,
1.160136864e-008,-2.973416333e-012},
blow={23954.14002,30.89260431},
ahigh={800298.733,-3237.69658,8.703408870000001,-0.002701025798,
9.15004211e-007,-1.36525663e-010,7.23462441e-015},
bhigh={46428.19450000001,-30.19933248},
R=251.9147402635602);
```

```

constant IdealGases.Common.DataRecord NF2 (
  name="NF2",
  MM=0.0520035064,
  Hf=661906.270997142,
  H0=203478.6831220289,
  Tlimit=1000,
  allow={15118.31104, 91.9638994, 0.494730179, 0.02001847323, -2.767712684e-005,
         1.872924867e-008, -5.020318040000001e-012},
  blow={2839.279606, 22.7051567},
  ahighe={-194501.0078, -353.603407, 7.26349436, -8.4023804e-005, 2.326721111e-008,
          -2.667577562e-012, 1.236070614e-016},
  bhigh={3435.44059, -13.36102511},
  R=159.8829112799979);

constant IdealGases.Common.DataRecord NF3 (
  name="NF3",
  MM=0.07100190960000001,
  Hf=-1854879.689038673,
  H0=166960.4531312493,
  Tlimit=1000,
  allow={87571.49280000001, -903.1832890000001, 4.02741727, 0.02314439555, -
         3.41510647e-005,
         2.409483651e-008, -6.63346419e-012},
  blow={-12372.32074, 0.3026430713},
  ahighe={-349626.876, -497.372867, 10.36866128, 8.90068765e-005, 5.88265436e-008,
          -3.157737664e-012, 1.714329953e-016},
  bhigh={-17131.83352, -30.98920858},
  R=117.1020898851994);

constant IdealGases.Common.DataRecord NH (
  name="NH",
  MM=0.01501464,
  Hf=23778925.16903503,
  H0=572847.7672458348,
  Tlimit=1000,
  allow={13596.5132, -190.0296604, 4.51849679, -0.002432776899, 2.377587464e-006, -
         2.592797084e-010,
         -2.659680792e-013},
  blow={42809.7219, -3.886561616},
  ahighe={1958141.991, -5782.861300000001, 9.33574202, -0.002292910311,
          6.07609248e-007, -6.647942750000001e-011, 2.384234783e-015},
  bhigh={78989.1234, -41.169704},
  R=553.7576658514623);

constant IdealGases.Common.DataRecord NHplus (
  name="NHplus",
  MM=0.0150140914,
  Hf=110948299.8085385,
  H0=632413.493899471,
  Tlimit=1000,
  allow={4253.656849999999, -245.8222206, 6.70891949, -0.0103848943,
         1.509008623e-005, -9.58051219e-009, 2.333206758e-012},
  blow={200107.7797, -13.95057632},
  ahighe={1405709.438, -4136.21571, 7.63201448, -0.001228325778, 2.721187746e-007,
          -2.010098289e-011, 3.71719018e-017},
  bhigh={225897.596, -27.86785234},
  R=553.7778996070318);

```

```
constant IdealGases.Common.DataRecord NHF (
  name="NHF",
  MM=0.0340130432,
  Hf=3292854.43062031,
  H0=294878.8775242552,
  Tlimit=1000,
  allow={-51106.59820000001, 961.225643, -2.706446594, 0.0203656268, -2.425558952e-
005,
         1.551553017e-008, -4.05845826e-012},
  blow={7909.62834, 40.99317124},
  ahigh={901390.2720000001, -3463.39705, 8.705804860000001, -
0.0004018963409999999,
         2.322774501e-008, 6.28048733e-012, -6.28309569e-016},
  bhigh={33370.6534, -29.00483634},
  R=244.4495175309689);

constant IdealGases.Common.DataRecord NHF2 (
  name="NHF2",
  MM=0.0530114464,
  Hf=-1942976.602124933,
  H0=203869.9136494416,
  Tlimit=1000,
  allow={-56261.1342, 1205.756556, -6.01752942, 0.0376002769, -4.6191986e-005,
         2.94247557e-008, -7.59873236e-012},
  blow={-18970.14374, 59.01714907},
  ahigh={739427.899, -4004.47177, 12.2132232, -0.000697043773, 1.271073981e-007, -
1.247136898e-011,
         5.08651759e-016},
  bhigh={9134.195979999999, -48.67843963},
  R=156.8429568448825);

constant IdealGases.Common.DataRecord NH2 (
  name="NH2",
  MM=0.01602258,
  Hf=11804260.79944678,
  H0=620241.371863957,
  Tlimit=1000,
  allow={-31182.40659, 475.424339, 1.372395176, 0.006306429719999999, -5.98789356e-
006,
         4.49275234e-009, -1.414073548e-012},
  blow={19289.39662, 15.40126885},
  ahigh={2111053.74, -6880.62723, 11.32305924, -0.001829236741, 5.64389009e-007, -
7.88645248e-011,
         4.078593449999999e-015},
  bhigh={65037.7856, -53.59155744},
  R=518.9221710860547);

constant IdealGases.Common.DataRecord NH2F (
  name="NH2F",
  MM=0.0350209832,
  Hf=-2141573.226876166,
  H0=288540.6141310162,
  Tlimit=1000,
  allow={-109237.476, 1844.91978, -7.6738716, 0.0322953344, -3.38810867e-005,
         1.97187155e-008, -4.81020515e-012},
  blow={-18783.1896, 68.61483738999999},
  ahigh={1927205.34, -7500.447160000001, 13.9558958, -0.00118480442,
         2.05067867e-007, -1.90876131e-011, 7.38923621e-016},
```

```

bhigh={35529.2734,-67.31185490999999},
R=237.4140084108204);

constant IdealGases.Common.DataRecord NH3(
  name="NH3",
  MM=0.01703052,
  Hf=-2697510.117130892,
  H0=589713.1150428759,
  Tlimit=1000,
  allow={-76812.26149999999,1270.951578,-3.89322913,0.02145988418,-
2.183766703e-005,
  1.317385706e-008,-3.33232206e-012},
  blow={-12648.86413,43.66014588},
  ahigh={2452389.535,-8040.89424,12.71346201,-0.000398018658,3.55250275e-008,
  2.53092357e-012,-3.32270053e-016},
  bhigh={43861.91959999999,-64.62330602},
  R=488.2101075011215);

constant IdealGases.Common.DataRecord NH2OH(
  name="NH2OH",
  MM=0.03302992,
  Hf=-1513779.022171413,
  H0=340169.7006835016,
  Tlimit=1000,
  allow={-56175.8667,1209.290057,-6.17959906,0.0405311644,-5.19010554e-005,
  3.59454458e-008,-9.93368163999999e-012},
  blow={-12658.88352,57.2793292800001},
  ahigh={4878285.05,-15336.04636,22.2723999,-0.002514583678,3.33958973e-007,-
1.881744532e-011,
  1.918174365e-016},
  bhigh={89230.2071,-126.9053624},
  R=251.7254658806319);

constant IdealGases.Common.DataRecord NH4plus(
  name="NH4plus",
  MM=0.0180379114,
  Hf=35752750.45424605,
  H0=553211.0552444559,
  Tlimit=1000,
  allow={-266831.5752,3763.02069,-15.71327725,0.0454882021,-4.37996212e-005,
  2.464478293e-008,-5.96153233e-012},
  blow={58232.8472,111.2087156},
  ahigh={4141889,-14420.72042,20.11893564,-0.001971492619,3.112721421e-007,-
2.602979969e-011,
  8.89434212999999e-016},
  bhigh={166419.6236,-120.1535761},
  R=460.944275399867);

constant IdealGases.Common.DataRecord NO (
  name="NO",
  MM=0.0300061,
  Hf=3041758.509103149,
  H0=305908.1320131574,
  Tlimit=1000,
  allow={-11439.16503,153.6467592,3.43146873,-0.002668592368,8.48139912e-006,-
7.685111050000001e-009,
  2.386797655e-012},

```

---

**1130 Modelica.Media.IdealGases.Common.SingleGasesData**

---

```
blow={9098.214410000001, 6.72872549},  
ahigh={223901.8716, -1289.651623, 5.43393603, -0.00036560349,  
      9.880966450000001e-008, -1.416076856e-011, 9.380184619999999e-016},  
bhigh={17503.17656, -8.50166909},  
R=277.0927244793559;  
  
constant IdealGases.Common.DataRecord NOCL(  
  name="NOCL",  
  MM=0.06545910000000001,  
  Hf=805064.9642295723,  
  H0=173612.0417176527,  
  Tlimit=1000,  
  alow={23088.35209, -549.598384, 7.73046336, -0.0050739109, 1.062996184e-005, -  
8.7932497e-009,  
      2.648180166e-012},  
  blow={7389.89839, -13.18393021},  
  ahigh={-613341.333, -391.929883, 9.13891722, -0.002605664613, 1.295687247e-006,  
      -2.215378352e-010, 1.280394898e-014},  
  bhigh={4517.32842, -23.07323335},  
  R=127.0178172324398);  
  
constant IdealGases.Common.DataRecord NOF(  
  name="NOF",  
  MM=0.0490045032,  
  Hf=-1326408.712577255,  
  H0=218760.9362398352,  
  Tlimit=1000,  
  alow={47550.2426, -725.390417000001, 7.21399636, -0.002532427181,  
      6.3777439e-006, -5.51830588e-009, 1.681935713e-012},  
  blow={-5609.72252, -12.89663616},  
  ahigh={1889069.274, -6731.02266, 14.19018767, -0.00369312462, 9.93857514e-007, -  
1.080748188e-010,  
      4.21035443e-015},  
  bhigh={32099.0078, -63.70266962},  
  R=169.667509250456);  
  
constant IdealGases.Common.DataRecord NOF3(  
  name="NOF3",  
  MM=0.08700130959999999,  
  Hf=-2149392.932816267,  
  H0=157441.9174030456,  
  Tlimit=1000,  
  alow={148836.0135, -2241.049812, 13.02355027, 0.00546397668, -  
8.641865250000001e-006,  
      5.91365903e-009, -1.577009169e-012},  
  blow={-13283.42568, -48.77320739},  
  ahigh={-278562.5217, -1252.321663, 13.90824337, -0.00035666875, 7.78501106e-008,  
      -8.85041197e-012, 4.07598003e-016},  
  bhigh={-20256.51446, -51.06881858999999},  
  R=95.56720511710552);  
  
constant IdealGases.Common.DataRecord NO2(  
  name="NO2",  
  MM=0.0460055,  
  Hf=743237.6346306421,  
  H0=221890.3174620426,  
  Tlimit=1000,
```

```

alow={-56420.3878,963.308572,-2.434510974,0.01927760886,-1.874559328e-005,
      9.145497730000001e-009,-1.777647635e-012},
blow={-1547.925037,40.6785121},
ahigh={721300.157,-3832.6152,11.13963285,-0.002238062246,6.54772343e-007,-
7.6113359e-011,
      3.32836105e-015},
bhigh={25024.97403,-43.0513004},
R=180.7277825477389);

constant IdealGases.Common.DataRecord NO2minus(
  name="NO2minus",
  MM=0.0460060486,
  Hf=-4348028.07646471,
  H0=221210.2388641132,
  Tlimit=1000,
  alow={-12820.67858,699.013818,-2.812596273,0.02412894252,-2.831606689e-005,
        1.670509365e-008,-3.98333013e-012},
  blow={-28099.15579,40.6327151},
  ahigh={132571.0335,-1557.032129,8.12672192,-0.000272862678,-4.7075418e-008,
        2.826729008e-011,-2.353985481e-015},
  bhigh={-17157.95217,-22.28576043},
  R=180.7256274558646);

constant IdealGases.Common.DataRecord NO2CL(
  name="NO2CL",
  MM=0.0814585,
  Hf=153452.3714529484,
  H0=149828.9190201145,
  Tlimit=1000,
  alow={8508.370340000001,-180.5383762,3.78538856,0.01414934934,-1.423946765e-
005,
        7.02822618e-009,-1.374688214e-012},
  blow={915.6246469999999,6.958904458},
  ahigh={-108677.3327,-1452.231167,11.05656962,-0.000400009928,
        9.101543039999999e-008,-1.036656913e-011,4.78166481e-016},
  bhigh={6294.26732,-35.21239681},
  R=102.0700356623311);

constant IdealGases.Common.DataRecord NO2F(
  name="NO2F",
  MM=0.0650039031999999,
  Hf=-1676822.385028719,
  H0=174552.3028838675,
  Tlimit=1000,
  alow={56678.5695,-653.825195,4.47277152,0.01368870672,-1.460533236e-005,
        7.779227940000001e-009,-1.689355106e-012},
  blow={-11021.79443,0.329207431},
  ahigh={-100857.7842,-1704.722752,11.22954945,-0.000468521597,
        1.047692566e-007,-1.189150595e-011,5.470307120000001e-016},
  bhigh={-6891.71918,-38.49788492},
  R=127.9072731127937);

constant IdealGases.Common.DataRecord NO3(
  name="NO3",
  MM=0.0620049,
  Hf=1147135.145770738,
  H0=176742.7090439627,

```

---

**1132 Modelica.Media.IdealGases.Common.SingleGasesData**

---

```
Tlimit=1000,
alow={34053.9841,226.6670652,-3.79308163,0.041707327,-5.709913270000001e-
005,
    3.83415811e-008,-1.021969284e-011},
blow={7088.112200000001,42.73091713},
ahigh={-394387.271,-824.426353,10.61325843,-0.0002448749816,5.40606032e-008,
    -6.19546675e-012,2.870000149e-016},
bhigh={8982.01173,-34.44666597},
R=134.0937893618085);

constant IdealGases.Common.DataRecord NO3minus(
  name="NO3minus",
  MM=0.0620054486,
  Hf=-5012132.611197655,
  H0=173744.4892867044,
  Tlimit=1000,
  alow={92048.1361,-391.117115,-0.2354356764,0.02836042108,-3.46132408e-005,
    2.08178746e-008,-5.02160127e-012},
  blow={-35764.115,22.99942308},
  ahigh={-311000.5758,-1369.087552,11.01342913,-0.000403687882,
    8.90208647e-008,-1.01973348e-011,4.72333079e-016},
  bhigh={-33643.2109,-38.78432657},
  R=134.0926029523154);

constant IdealGases.Common.DataRecord NO3F(
  name="NO3F",
  MM=0.0810033032,
  Hf=185177.6335955643,
  H0=178315.6418243448,
  Tlimit=1000,
  alow={64728.3203,-821.3134309999999,6.19491744,0.01805438628,-1.99669324e-
005,
    1.124482018e-008,-2.680013077e-012},
  blow={4206.66179,-7.016104301},
  ahigh={-341179.33,-2353.908798,16.28114887,-0.001910415273,4.69087356e-007,
    -5.68604014e-011,2.720906921e-015},
  bhigh={9760.583979999999,-65.58153684},
  R=102.6436166371052);

constant IdealGases.Common.DataRecord N2 (
  name="N2",
  MM=0.0280134,
  Hf=0,
  H0=309498.4543111511,
  Tlimit=1000,
  alow={22103.71497,-381.846182,6.08273836,-0.00853091441,1.384646189e-005,-
9.62579362e-009,
    2.519705809e-012},
  blow={710.846086,-10.76003744},
  ahigh={587712.406,-2239.249073,6.06694922,-0.00061396855,1.491806679e-007,-
1.923105485e-011,
    1.061954386e-015},
  bhigh={12832.10415,-15.86640027},
  R=296.8033869505308);

constant IdealGases.Common.DataRecord N2plus(
  name="N2plus",
```

```

MM=0.0280128514,
Hf=53886282.4938985,
H0=309540.07059774,
Tlimit=1000,
alow={-34740.4747, 269.6222703, 3.16491637, -0.002132239781,
      6.73047639999999e-006, -5.63730497e-009, 1.621756e-012},
blow={179000.4424, 6.832974166},
ahigh={-2845599.002, 7058.89303, -2.884886385, 0.003068677059, -4.36165231e-007,
       2.102514545e-011, 5.41199647e-016},
bhigh={134038.8483, 50.90897022},
R=296.809199509051);

constant IdealGases.Common.DataRecord N2minus (
  name="N2minus",
  MM=0.0280139486,
  Hf=5289624.969184102,
  H0=309641.5333609915,
  Tlimit=1000,
  alow={-81462.2711, 906.360079, -0.1520054079, 0.00602319084, -2.897138445e-006,
        -4.12910668e-011, 3.20698977e-013},
  blow={12188.08548, 26.38068855},
  ahigh={216963.7706, -1275.098516, 5.3910957, -0.000319890751,
         7.311051349999999e-008, -8.202017370000001e-012, 3.7400447e-016},
  bhigh={24249.64308, -9.014934294},
  R=296.7975746196665);

constant IdealGases.Common.DataRecord NCN (
  name="NCN",
  MM=0.0400241,
  Hf=12503880.73685604,
  H0=254351.178415005,
  Tlimit=1000,
  alow={-56346.80699999999, 732.380458, -0.782140184, 0.01838552441, -
1.950836491e-005,
        1.035712021e-008, -2.208158483e-012},
  blow={55397.897, 29.05308985},
  ahigh={-164188.0975, -776.784075, 7.99998187, -0.0001659081508,
         2.983403318e-008, -3.120157047e-012, 1.99269872e-016},
  bhigh={61844.24479999999, -21.4910882},
  R=207.7366386751982);

constant IdealGases.Common.DataRecord N2D2_cis (
  name="N2D2_cis",
  MM=0.032041604,
  Hf=6331060.392607061,
  H0=321707.8957720095,
  Tlimit=1000,
  alow={-27437.33656, 714.980883, -2.22324762, 0.02088722282, -1.821711897e-005,
        8.84407994e-009, -1.918010649e-012},
  blow={20111.15649, 36.37100195},
  ahigh={879807.471, -5299.36204, 13.55007485, -0.001316635227, 2.755816197e-007,
        -3.036294387e-011, 1.365324117e-015},
  bhigh={53563.11889999999, -62.71215875},
  R=259.4898807188304);

constant IdealGases.Common.DataRecord N2F2 (
  name="N2F2",

```

---

**1134 Modelica.Media.IdealGases.Common.SingleGasesData**

---

```
MM=0.06601020640000001,
Hf=944907.6044700869,
H0=194951.0038193124,
Tlimit=1000,
alow={15438.9315,-218.363513,3.89028425,0.0167401774,-2.05639309e-005,
      1.25869211e-008,-3.11049829e-012},
blow={7052.0387,5.265866442},
ahigh={-182488.386,-953.402996,10.6979548,-0.00027599687,6.0554397e-008,-
6.91121508e-012,
      3.19258723e-016},
bhigh={9283.46696,-32.46968772},
R=125.9573701317801);

constant IdealGases.Common.DataRecord N2F4 (
  name="N2F4",
  MM=0.1040070128,
  Hf=-211524.1982990593,
  H0=171253.5580100807,
  Tlimit=1000,
  alow={116291.4512,-1538.660418,9.054033049999999,0.02862113563,-4.33228699e-
005,
      3.067642499e-008,-8.45547411e-012},
  blow={2865.277526,-24.76493006},
  ahigh={-518859.471,-670.225651,16.50109262,-0.0002006404436,4.43675251e-008,
      -5.08980888e-012,2.359374159e-016},
  bhigh={-5281.4889,-60.40513435},
  R=79.94145563999894);

constant IdealGases.Common.DataRecord N2H2 (
  name="N2H2",
  MM=0.03002928,
  Hf=7055072.782297812,
  H0=332907.1159881289,
  Tlimit=1000,
  alow={-150400.5163,2346.687716,-9.40543029,0.032842998,-3.121920401e-005,
      1.72128319e-008,-4.014537220000001e-012},
  blow={13193.84041,78.32382629999999},
  ahigh={6217567.87,-17539.52096,20.22730509,-0.000975729766,-4.20841674e-007,
      1.117921171e-010,-7.627102210000001e-015},
  bhigh={137415.2574,-119.9559168},
  R=276.8788329257312);

constant IdealGases.Common.DataRecord NH2NO2 (
  name="NH2NO2",
  MM=0.06202808,
  Hf=-419164.9975301508,
  H0=196102.3620270045,
  Tlimit=1000,
  alow={-45730.3506,1201.365987,-8.10598411,0.054027152,-6.43807445e-005,
      4.02509792e-008,-1.02515419e-011},
  blow={-9615.78516,68.67353357},
  ahigh={1654040.575,-8125.220880000001,20.21742772,-0.001244291821,
      2.122804183e-007,-1.948359653e-011,7.43935136e-016},
  bhigh={42308.2258,-101.6190179},
  R=134.0436782824811);

constant IdealGases.Common.DataRecord N2H4 (
```

```

name="N2H4",
MM=0.03204516,
Hf=2970183.328777263,
H0=357286.5293854048,
Tlimit=1000,
allow={-166075.6354,3035.416736,-17.36889823,0.0715983402,-8.8667993e-005,
      5.79897028e-008,-1.530037218e-011},
blow={-3731.92723,119.0002218},
ahigh={3293486.7,-11998.50628,21.04406814,-0.001399381724,1.933173351e-007,
      -1.318016127e-011,3.16640017e-016},
bhigh={83484.337,-115.5751024},
R=259.4610855430274);

constant IdealGases.Common.DataRecord N2O(
  name="N2O",
  MM=0.0440128,
  Hf=1854006.107314236,
  H0=217685.1961247637,
  Tlimit=1000,
  allow={42882.2597,-644.011844,6.03435143,0.0002265394436,3.47278285e-006,-
3.62774864e-009,
      1.137969552e-012},
  blow={11794.05506,-10.0312857},
  ahigh={343844.804,-2404.557558,9.125636220000001,-0.000540166793,
      1.315124031e-007,-1.4142151e-011,6.38106687e-016},
  bhigh={21986.32638,-31.47805016},
  R=188.9103169986913);

constant IdealGases.Common.DataRecord N2Oplus(
  name="N2Oplus",
  MM=0.04401225139999999,
  Hf=30286033.19756553,
  H0=241373.8598248578,
  Tlimit=1000,
  allow={-56241.4708,669.621161,0.0878145619,0.01524476027,-1.527290811e-005,
      7.827237389999999e-009,-1.646739623e-012},
  blow={155729.5192,25.62354785},
  ahigh={-29835.53254,-1179.455967,8.30018669,-0.0002887267217,
      5.70510501e-008,-5.95888512e-012,2.835725557e-016},
  bhigh={164602.1769,-22.87356617},
  R=188.912671712227);

constant IdealGases.Common.DataRecord N2O3(
  name="N2O3",
  MM=0.0760116,
  Hf=1139702.295439117,
  H0=225240.029153445,
  Tlimit=1000,
  allow={-92044.44170000001,929.552015,3.20366481,0.01356473078,-6.26296607e-
006,
      -1.402915559e-009,1.43162093e-012},
  blow={3313.62208,18.44430953},
  ahigh={778388.186,-4483.02466,16.66668024,-0.002062143878,
      5.309541710000001e-007,-6.19045122e-011,2.692956658e-015},
  bhigh={33609.1245,-67.39212388},
  R=109.384251877345);

```

---

**1136 Modelica.Media.IdealGases.Common.SingleGasesData**

---

```
constant IdealGases.Common.DataRecord N204 (
  name="N2O4",
  MM=0.092011,
  Hf=120756.4204279923,
  H0=181948.1475041028,
  Tlimit=1000,
  allow={-38047.5144, 561.2828890000001, -0.2083648324, 0.0388708782, -4.42241226e-005,
         2.49881231e-008, -5.67910238e-012},
  blow={-3310.79473, 29.6392484},
  ahighelement1=-458284.3760000001, -1604.749805, 16.74102133, -0.0005091385080000001,
         1.14363467e-007, -1.316288176e-011, 5.976316620000001e-016},
  bhigh={4306.90052, -65.69450380000001},
  R=90.36389127386944);

constant IdealGases.Common.DataRecord N205 (
  name="N2O5",
  MM=0.1080104,
  Hf=123136.2905794257,
  H0=192550.2081281062,
  Tlimit=1000,
  allow={40078.2817, -876.9675120000001, 10.55932981, 0.01394613859, -8.884346920000001e-006,
         8.500431150000001e-010, 7.79155091e-013},
  blow={3038.962037, -23.8683186},
  ahighelement1=-53255.7896, -3109.277389, 20.36088958, -0.000995990114,
         2.401398635e-007, -3.057161911e-011, 1.495915511e-015},
  bhigh={13369.57281, -82.98623341000001},
  R=76.97843911327057);

constant IdealGases.Common.DataRecord N3 (
  name="N3",
  MM=0.0420201,
  Hf=10375986.73016009,
  H0=227769.7102101138,
  Tlimit=1000,
  allow={33374.0679, -296.5683604, 3.31427915, 0.00672168536, -4.18112639e-006,
         8.61844236e-010, 6.88335253e-014},
  blow={52988.4062, 5.312776486},
  ahighelement1=252926.4658, -2362.876591, 9.135267130000001, -0.000621287085,
         1.324094351e-007, -1.47898964e-011, 6.721230470000001e-016},
  bhigh={64126.95389999999, -31.35825973},
  R=197.8689246336872);

constant IdealGases.Common.DataRecord N3H (
  name="N3H",
  MM=0.04302804,
  Hf=6832753.711300817,
  H0=254419.745821562,
  Tlimit=1000,
  allow={3242.57606, 66.9266489, 1.766142217, 0.01487411419, -1.53908644e-005,
         9.172303550000001e-009, -2.337205474e-012},
  blow={33920.697, 15.13752057},
  ahighelement1=1170469.241, -5102.45199, 12.7828891, -0.000840948716, 1.592142834e-007,
         -1.512289051e-011, 6.102906629999999e-016},
  bhigh={64283.4447, -55.13119107999999},
  R=193.2338075357372);
```

```

constant IdealGases.Common.DataRecord Na (
  name="Na",
  MM=0.02298977,
  Hf=4675992.843773557,
  H0=269573.2928167615,
  Tlimit=1000,
  allow={0,0,2.5,0,0,0,0},
  blow={12183.82949,4.24402818},
  ahigh={952572.3380000001,-2623.807254,5.16259662,-0.001210218586,
    2.306301844e-007,-1.249597843e-011,7.226771190000001e-016},
  bhigh={29129.63564,-15.19717061},
  R=361.6596425279592);

constant IdealGases.Common.DataRecord Naplus (
  name="Naplus",
  MM=0.0229892214,
  Hf=26514291.9542286,
  H0=269579.7257405159,
  Tlimit=1000,
  allow={0,0,2.5,0,0,0,0},
  blow={72565.3707,3.55084508},
  ahigh={0,0,2.5,0,0,0,0},
  bhigh={72565.3707,3.55084508},
  R=361.6682729411619);

constant IdealGases.Common.DataRecord Naminus (
  name="Naminus",
  MM=0.0229903186,
  Hf=2107557.917879398,
  H0=269566.8602000148,
  Tlimit=1000,
  allow={0,0,2.500000001,0,0,0,0},
  blow={5082.19967,3.55091679},
  ahigh={0,0,2.500000001,0,0,0,0},
  bhigh={5082.19967,3.55091679},
  R=361.6510125266381);

constant IdealGases.Common.DataRecord NaALF4 (
  name="NaALF4",
  MM=0.1259649208,
  Hf=-14748880.35653812,
  H0=165194.1418916051,
  Tlimit=1000,
  allow={131535.9536,-2394.685695,19.09539612,0.001044339733,-
    6.94975412999999e-006,
    6.76909813999999e-009,-2.176199098e-012},
  blow={-215051.2366,-75.0510213},
  ahigh={-357117.793,-249.6543879,16.18634511,-7.449049300000001e-005,
    1.644510872e-008,-1.883696066e-012,8.719877450000001e-017},
  bhigh={-227953.3538,-53.7066514},
  R=66.00624957484196);

constant IdealGases.Common.DataRecord NaBO2 (
  name="NaBO2",
  MM=0.06579957,
  Hf=-9626952.425372992,
  H0=207944.6719788594,

```

---

**1138 Modelica.Media.IdealGases.Common.SingleGasesData**

---

```
Tlimit=1000,
allow={50468.6736,-869.99706,8.94220823999999,0.001092655824,
       1.440608828e-006,-2.254633535e-009,8.12331285e-013},
blow={-73783.1836,-19.51815135},
ahigh={85958.1404999999,-1691.902222,11.18847517,-0.000456420231,
       9.80450516e-008,-1.101655008e-011,5.02917276e-016},
bhigh={-69492.9206,-34.1635313},
R=126.360582599552);

constant IdealGases.Common.DataRecord NaBr (
  name="NaBr",
  MM=0.10289377,
  Hf=-1418247.985276465,
  H0=95449.10250640052,
  Tlimit=1000,
  allow={-14668.27136,76.2228287000001,3.92116347,0.001793780908,-
2.457391648e-006,
       1.739334522e-009,-4.79670138e-013},
  blow={-19264.91755,6.40438286},
  ahigh={897851.3019999999,-2697.899721,7.54537716,-0.001566692184,
       4.47039274e-007,-4.83281021e-011,1.4239613e-015},
  bhigh={-1750.862247,-18.56478949},
  R=80.80636952072025);

constant IdealGases.Common.DataRecord NaCN (
  name="NaCN",
  MM=0.04900717,
  Hf=1923514.457170247,
  H0=247432.6103710947,
  Tlimit=1000,
  allow={24461.6541,-753.601997,10.44966241,-0.01168769048,1.849279385e-005,-
1.319387974e-008,
       3.58976656e-012},
  blow={12978.20493,-29.88519681},
  ahigh={366557.029,-1782.094358,8.69962263,-0.0004463297,9.36222931e-008,-
1.03307132e-011,
       4.650466810000001e-016},
  bhigh={20109.26347,-24.19276094},
  R=169.6582765338215);

constant IdealGases.Common.DataRecord NaCL (
  name="NaCL",
  MM=0.05844277000000001,
  Hf=-3106370.283270283,
  H0=164521.9075002776,
  Tlimit=1000,
  allow={43623.78350000001,-758.303446,8.259173000000001,-0.009640915140000001,
       1.358854616e-005,-9.66703224999999e-009,2.74626129e-012},
  blow={-19504.09477,-19.36687551},
  ahigh={331449.8760000001,-896.831565,5.27728738,-0.0001475674008,-
1.491128988e-008,
       2.465673596e-011,-2.730355213e-015},
  bhigh={-17362.77667,-3.99828856},
  R=142.2669048712099);

constant IdealGases.Common.DataRecord NaF (
  name="NaF",
```

```

MM=0.0419881732,
Hf=-7029524.423320232,
H0=219707.3675022375,
Tlimit=1000,
alow={39598.8744,-653.462606999999,6.9411732,-0.00530781493,
      6.97972066e-006,-4.82042573e-009,1.364849175e-012},
blow={-33529.4089,-14.03212229},
ahigh={-1092926.912,3293.30364,0.413591984,0.00263499447,-
8.384295630000001e-007,
      1.417053025e-010,-8.600270160000001e-015},
bhigh={-57728.8463,29.09489906},
R=198.0193794189646);

constant IdealGases.Common.DataRecord NaH(
  name="NaH",
  MM=0.02399771,
  Hf=5868689.345775076,
  H0=363830.757184748,
  Tlimit=1000,
  alow={-32222.0641,623.762201,-0.921627750999999,0.01360765851,-
1.659249878e-005,
      1.033284544e-008,-2.589726392e-012},
  blow={13073.81654,26.41418597},
  ahigh={-4756184.75,14520.47626,-13.27563485,0.01055828277,-2.990041189e-006,
      3.90532288e-010,-1.923931194e-014},
  bhigh={-76329.2227,122.207006},
  R=346.4693922878474);

constant IdealGases.Common.DataRecord NaI(
  name="NaI",
  MM=0.14989424,
  Hf=-604678.8722501947,
  H0=66394.27905968903,
  Tlimit=1000,
  alow={12288.68506,-285.71928,5.961057,-0.00379253258,5.56688131e-006,-
4.045013680000001e-009,
      1.172998933e-012},
  blow={-10882.50556,-3.99073992},
  ahigh={2281549.408,-7093.15663,13.04994968,-0.00501349233,1.581735155e-006,
      -2.285837754e-010,1.19588995e-014},
  bhigh={32538.9362,-56.4002332},
  R=55.46892262170982);

constant IdealGases.Common.DataRecord NaLi(
  name="NaLi",
  MM=0.02993077,
  Hf=5967047.289461648,
  H0=333874.4709875489,
  Tlimit=1000,
  alow={-6569.99276,-5.25394043,4.32938426,0.001097919189,-1.965726597e-006,
      2.086472026e-009,-8.19598811e-013},
  blow={20162.24293,1.413253322},
  ahigh={10916648.6,-34800.1064,46.1431191999999,-0.02260951051,
      5.68532437e-006,-6.45814329999999e-010,2.696508992e-014},
  bhigh={239443.1755,-296.1780126},
  R=277.7901136522716);

```

---

**1140 Modelica.Media.IdealGases.Common.SingleGasesData**

---

```
constant IdealGases.Common.DataRecord NaNO2 (
  name="NaNO2",
  MM=0.06899527000000001,
  Hf=-2410205.496695643,
  H0=214612.1176132798,
  Tlimit=1000,
  allow={-69344.60610000001,1115.43267,-1.816284973,0.02916366016,-3.50578852e-005,
         2.128602855e-008,-5.226211990000001e-012},
  blow={-27072.75053,41.21853253},
  ahighelement1=-176555.7495,-863.521295,10.6410594,-0.0002559465044,5.65446563e-008,
        -6.48671032e-012,3.008171243e-016},
  bhighelement1=-18684.35672,-29.21304474},
  R=120.5078551036904);

constant IdealGases.Common.DataRecord NaNO3 (
  name="NaNO3",
  MM=0.0849946699999999,
  Hf=-3359371.958265148,
  H0=181202.997787979,
  Tlimit=1000,
  allow={-20438.52507,691.709341,-2.397485012,0.04100386430000001,-5.14170257999999e-005,
         3.22862247e-008,-8.1646916e-012},
  blow={-39064.0132,41.74749459},
  ahighelement1=-322559.219,-1396.782195,14.03092409,-0.000409816177,
        9.023845720000001e-008,-1.032543966e-011,4.77867155e-016},
  bhighelement1=-31388.89875,-49.59121431},
  R=97.82345175291582);

constant IdealGases.Common.DataRecord NaO (
  name="NaO",
  MM=0.03898917,
  Hf=2731664.126217614,
  H0=250149.387637644,
  Tlimit=1000,
  allow={18577.48013,-337.149732,5.64456002,-0.003136926368,6.33077539e-006,-5.42946247e-009,
         1.68718377e-012},
  blow={13203.32678,-4.99613115},
  ahighelement1=256974.4011,-2269.334161,9.22439762,-0.0036512691,1.446811119e-006,-2.443068386e-010,
        1.428508328e-014},
  bhighelement1=24132.39357,-29.89159486},
  R=213.2508078525396);

constant IdealGases.Common.DataRecord NaOH (
  name="NaOH",
  MM=0.03999711,
  Hf=-4775345.0186776,
  H0=284962.3635307651,
  Tlimit=1000,
  allow={34420.3674,-792.321818,8.9979323,-0.00407984452,3.065783937e-006,-5.11918934e-010,
         -1.541016409e-013},
  blow={-20869.51091,-25.1059009},
  ahighelement1=875378.776,-2342.514649,7.97846989,0.0001016451512,-6.26853195e-008,
        1.022715136e-011,-5.71328641e-016},
```

```

bhigh={-9509.90171,-22.02310401},
R=207.87681910018);

constant IdealGases.Common.DataRecord NaOHplus(
  name="NaOHplus",
  MM=0.0399965614,
  Hf=17098029.10707219,
  H0=292396.9109004455,
  Tlimit=1000,
  allow={22780.39363,-667.219422,8.921593120000001,-0.004481263270000001,
    3.95188392e-006,-1.173341234e-009,2.041806631e-014},
  blow={83633.8219,-22.58912262},
  ahigh={881541.365,-2363.159755,8.053203930000001,7.27127974e-005,-
5.89741789e-008,
    1.007487881e-011,-5.47440621e-016},
  bhigh={95844.41650000001,-20.79484012},
  R=207.8796703758639);

constant IdealGases.Common.DataRecord Na2 (
  name="Na2",
  MM=0.04597954,
  Hf=3095705.720413906,
  H0=226255.5258273571,
  Tlimit=1000,
  allow={6848.62868,-153.0836599,5.32523039,-0.001944906088,2.657477888e-006,-
9.096841120000001e-010,
    -2.44875673e-013},
  blow={16491.70574,-2.653564394},
  ahigh={19299407.58,-62692.8012,82.67682110000001,-0.0456513781,
    1.259515667e-005,-1.560445735e-009,7.02467717e-014},
  bhigh={409082.08,-550.997089},
  R=180.8298212639796);

constant IdealGases.Common.DataRecord Na2Br2 (
  name="Na2Br2",
  MM=0.20578754,
  Hf=-2336625.691720694,
  H0=95202.78535814169,
  Tlimit=1000,
  allow={-13849.46731,-157.3241177,10.63192589,-0.001383418339,
    1.697156537e-006,-1.092992547e-009,2.87098802e-013},
  blow={-60103.9182,-18.61473856},
  ahigh={-30217.30975,-2.65923278,10.00225819,-9.935490630000001e-007,
    2.358676253e-010,-2.857194816e-014,1.381909879e-018},
  bhigh={-60900.3992,-14.94517863},
  R=40.40318476036013);

constant IdealGases.Common.DataRecord Na2Cl2 (
  name="Na2Cl2",
  MM=0.11688554,
  Hf=-4828670.706402178,
  H0=159979.8144406913,
  Tlimit=1000,
  allow={-10829.5525,-313.9528641,11.24411332,-0.002697665795,3.28635948e-006,
    -2.105377639e-009,5.508056609999999e-013},
  blow={-69386.7543,-25.11011582},
  ahigh={-44121.0523,-5.39349128,10.00453407,-1.981155641e-006,4.6801882e-010,
    1.007487881e-011,-5.47440621e-016});

```

```
-5.64894068e-014, 2.724721177e-018},  
bhigh={-70980.6341, -17.87230397},  
R=71.13345243560495);  
  
constant IdealGases.Common.DataRecord Na2F2 (  
  name="Na2F2",  
  MM=0.0839763464,  
  Hf=-9932115.551052362,  
  H0=198655.1060525777,  
  Tlimit=1000,  
  allow={23515.80802, -1000.737062, 13.75845076, -0.007838346979999999,  
         9.2762887e-006, -5.81361297e-009, 1.49535482e-012},  
  blow={-98358.31880000001, -43.8114809},  
  ahight={-90307.50689999999, -18.93938035, 10.01535518, -6.53967545e-006,  
          1.516333603e-009, -1.804877459e-013, 8.61347492e-018},  
  bhigh={-103489.6664, -21.78860202},  
  R=99.00968970948229);  
  
constant IdealGases.Common.DataRecord Na2I2 (  
  name="Na2I2",  
  MM=0.29978848,  
  Hf=-1190405.755417953,  
  H0=67534.50966494776,  
  Tlimit=1000,  
  allow={-13050.63983, -89.9250668, 10.36367976, -0.0008000070190000001,  
         9.84877798e-007, -6.35937312e-010, 1.673774775e-013},  
  blow={-45514.6549, -14.87509548},  
  ahight={-22312.36397, -1.515570558, 10.00129413, -5.71503538e-007,  
          1.360287207e-010, -1.650916036e-014, 7.996149680000001e-019},  
  bhigh={-45969.2175, -12.76523965},  
  R=27.73446131085491);  
  
constant IdealGases.Common.DataRecord Na2O (  
  name="Na2O",  
  MM=0.06197894,  
  Hf=-267184.7888976482,  
  H0=232501.0721383747,  
  Tlimit=1000,  
  allow={39011.49290000001, -726.620789, 9.62371078, -0.00355641864,  
         3.47070435e-006, -1.835177736e-009, 4.06213471e-013},  
  blow={-459.307325, -23.49565832},  
  ahight={-66005.2516, -25.69021634, 7.51938542, -7.81163524e-006, 1.73501589e-009,  
          -1.996634558e-013, 9.27643355e-018},  
  bhigh={-4297.33965, -10.63530214},  
  R=134.1499548072297);  
  
constant IdealGases.Common.DataRecord Na2Oplus (  
  name="Na2Oplus",  
  MM=0.0619783914,  
  Hf=8403476.699461419,  
  H0=235671.9442060253,  
  Tlimit=1000,  
  allow={26479.69755, -596.906216, 9.294129529999999, -0.003080702005,  
         3.080460653e-006, -1.669452593e-009, 3.78978414e-013},  
  blow={63473.1024999999, -20.25274459},  
  ahight={-57409.0933, -19.89178143, 7.51529526, -6.25833323e-006,  
          1.407321045e-009, -1.635833952e-013, 7.66244513e-018},
```

```

bhigh={60329.7746999999,-9.42652674},
R=134.1511422318069);

constant IdealGases.Common.DataRecord Na2O2 (
  name="Na2O2",
  MM=0.07797834000000001,
  Hf=-1589291.911061456,
  H0=199609.9814384353,
  Tlimit=1000,
  allow={73824.5892,-1355.12534,12.55579861,-0.002112046045,4.92035352e-008,
    1.003950603e-009,-4.44773207e-013},
  blow={-10588.58918,-40.2181178},
  ahigh={-173229.5239,-113.7561118,10.08529143,-3.42209894e-005,
    7.577721890000001e-009,-8.70125673e-013,4.03606139e-017},
  bhigh={-17803.00574,-23.8678919},
  R=106.6254039262698);

constant IdealGases.Common.DataRecord Na2O2H2 (
  name="Na2O2H2",
  MM=0.07999422000000001,
  Hf=-7800563.59071943,
  H0=241844.2357460327,
  Tlimit=1000,
  allow={108941.4289,-2634.822704,23.06608858,-0.01689310193,1.67348319e-005,-
    7.82116600999999e-009,
    1.47503447e-012},
  blow={-65931.3733000001,-98.0298758},
  ahigh={1675713.839,-4704.92117000001,16.97343934,0.0001961487616,-
    1.236951748e-007,
    2.025311778e-011,-1.132991623e-015},
  bhigh={-48562.4557,-68.1348098},
  R=103.93840955009);

constant IdealGases.Common.DataRecord Na2SO4 (
  name="Na2SO4",
  MM=0.14204214,
  Hf=-7322702.276944011,
  H0=147644.1991087997,
  Tlimit=1000,
  allow={83442.8321,-1210.880769,8.742353,0.0360079663,-5.11764596e-005,
    3.48156904e-008,-9.32193121999999e-012},
  blow={-121738.7045,-20.61505785},
  ahigh={-575448.311,-981.706033999999,19.73310397,-0.000293639552,
    6.49870816999999e-008,-7.462479400000001e-012,3.46249166e-016},
  bhigh={-127059.7762,-76.67871596000001},
  R=58.53524876490879);

constant IdealGases.Common.DataRecord Na3CL3 (
  name="Na3CL3",
  MM=0.17532831,
  Hf=-5205517.061106674,
  H0=168125.4841274635,
  Tlimit=1000,
  allow={-12542.1667,-550.216046,18.16110649,-0.00465651526,5.64643822e-006,-
    3.60476032e-009,
    9.40555623999999e-013},
  blow={-111927.0172,-52.9010496},

```

---

**1144 Modelica.Media.IdealGases.Common.SingleGasesData**

---

```
ahigh={-71602.9797,-9.57349381,16.00799614,-3.47835697e-006,
     8.190943689999999e-010,-9.86321003e-014,4.74900535e-018},
bhigh={-114725.2948,-40.31342160000001},
R=47.42230162373664);

constant IdealGases.Common.DataRecord Na3F3 (
  name="Na3F3",
  MM=0.1259645196,
  Hf=-10701548.26359533,
  H0=203249.8840252791,
  Tlimit=1000,
  allow={52470.3035,-1775.597273,22.50927609,-0.01333033615,1.555790239e-005,-
9.64601146e-009,
         2.460254271e-012},
  blow={-158073.6389,-85.80122369999999},
  ahigh={-155399.5698,-35.5197019,16.02841159,-1.198056094e-005,
         2.75724873e-009,-3.26330097e-013,1.550483212e-017},
  bhigh={-167214.1773,-47.5378068},
  R=66.00645980632153);

constant IdealGases.Common.DataRecord Nb (
  name="Nb",
  MM=0.09290638,
  Hf=7783244.799765098,
  H0=89919.54050948923,
  Tlimit=1000,
  allow={78896.60669999999,-1212.813914,10.34579819,-0.01676630056,
         1.979119979e-005,-1.218224409e-008,3.058098336e-012},
  blow={91653.1514,-35.9474285},
  ahigh={-1096553.196,2546.650713,2.236054882,-0.001280029198,
         8.464237990000001e-007,-1.486269508e-010,8.714309406000001e-015},
  bhigh={68791.24550000001,13.9816903},
  R=89.49301436564421);

constant IdealGases.Common.DataRecord Nbplus (
  name="Nbplus",
  MM=0.0929058314,
  Hf=15000185.17675092,
  H0=92449.25609696444,
  Tlimit=1000,
  allow={131444.7859,-2000.135035,15.05024212,-0.02996583942,3.72986863e-005,-
2.269869569e-008,
         5.449089902e-012},
  blow={176005.4029,-62.2459552},
  ahigh={-1077639.646,2159.046421,2.310604767,-0.0005363991760000001,
         5.05791509e-007,-1.032401533e-010,6.629241279999999e-015},
  bhigh={151794.5546,12.10678502},
  R=89.49354281328783);

constant IdealGases.Common.DataRecord Nbminus (
  name="Nbminus",
  MM=0.0929069285999999,
  Hf=6792324.507001301,
  H0=93144.91535134014,
  Tlimit=1000,
  allow={-72209.2485,525.950125,3.47046839,-0.00495050553,7.40185903e-006,-
5.01630207e-009,
```

```

    1.314100719e-012},
blow={71788.28870000001,5.15551007},
ahigh={111745.8019,134.0072834,2.391474129,4.52481343e-005,-1.025345e-008,
1.195577988e-012,-5.60633044999999e-017},
bhigh={74739.9675,9.67531561},
R=89.49248592424141);

constant IdealGases.Common.DataRecord NbCL5 (
  name="NbCL5",
  MM=0.27017138,
  Hf=-2603273.522162118,
  H0=97505.37973341218,
  Tlimit=1000,
  alow={73482.7797,-1919.172996,22.90799567,-0.01404352774,1.638106985e-005,-
1.018941843e-008,
2.612266639e-012},
blow={-79741.1881,-84.39637087},
ahigh={-156381.6638,-44.5052128,16.0354463,-1.489949737e-005,
3.42077792e-009,-4.04112442e-013,1.917266128e-017},
bhigh={-89628.15120000001,-43.74582607},
R=30.77480671712896);

constant IdealGases.Common.DataRecord NbO (
  name="NbO",
  MM=0.10890578,
  Hf=1937350.855023489,
  H0=80621.12038497864,
  Tlimit=1000,
  alow={-6797.834360000001,283.9767438,0.56457084,0.01134928619,-1.549821141e-
005,
1.047624988e-008,-2.762937835e-012},
blow={23179.93893,23.65708974},
ahigh={553225.878,-1287.669306,4.98006604,0.0001116014163,4.03183868e-008,
1.04877371e-011,-1.893595022e-015},
bhigh={32684.5779,-1.868958549},
R=76.34555300921586);

constant IdealGases.Common.DataRecord NbOCL3 (
  name="NbOCL3",
  MM=0.21526478,
  Hf=-3494765.841397743,
  H0=95840.10909727082,
  Tlimit=1000,
  alow={-110848.2359,117.3066983,11.65558433,0.001287422064,-1.659778281e-006,
1.102107826e-009,-2.949903264e-013},
blow={-95040.2392,-23.86450092},
ahigh={-151555.1427,53.0554523,12.28915282,-0.0001527366571,
6.778576440000001e-008,-9.36517065e-012,4.67003693e-016},
bhigh={-94907.68120000001,-27.45263834},
R=38.62439550027646);

constant IdealGases.Common.DataRecord NbO2 (
  name="NbO2",
  MM=0.12490518,
  Hf=-1611356.326454996,
  H0=85299.52080450147,
  Tlimit=1000,

```

---

**1146 Modelica.Media.IdealGases.Common.SingleGasesData**

---

```
alow={17390.46582,38.8102944,0.902429912,0.01909523319,-2.667013583e-005,
      1.813901051e-008,-4.872365880000001e-012},
blow={-25285.18558,22.89323379},
ahigh={-685185.99,911.224932,6.33919984,-8.2300803e-005,1.984439605e-007,-
3.26117149e-011,
      1.602430854e-015},
bhigh={-33250.2732,-3.55955655},
R=66.56627051015819);

constant IdealGases.Common.DataRecord Ne (
  name="Ne",
  MM=0.0201797,
  Hf=0,
  H0=307111.9986917546,
  Tlimit=1000,
  alow={0,0,2.5,0,0,0,0},
  blow={-745.375,3.35532272},
  ahigh={0,0,2.5,0,0,0,0},
  bhigh={-745.375,3.35532272},
  R=412.0215860493466);

constant IdealGases.Common.DataRecord Neplus (
  name="Neplus",
  MM=0.0201791514,
  Hf=103421888.4942803,
  H0=312412.2454425909,
  Tlimit=1000,
  alow={72815.5148,-869.5697989999999,6.10864697,-0.00584135693,
        5.041044170000001e-006,-2.293759207e-009,4.33906568e-013},
  blow={254599.689,-16.73449355},
  ahigh={-111274.2658,476.569797,2.196650531,0.0001102593151,-2.287564425e-
008,
        2.510218183e-012,-1.126646096e-016},
  bhigh={247253.6944,7.46614054},
  R=412.0327874639962);

constant IdealGases.Common.DataRecord Ni (
  name="Ni",
  MM=0.0586934,
  Hf=7328193.715136625,
  H0=116282.4610603577,
  Tlimit=1000,
  alow={-32358.1055,601.526462,-1.079270657,0.01089505519,-1.369578748e-005,
        8.317725790000001e-009,-2.019206968e-012},
  blow={48138.1081,27.188292},
  ahigh={-493826.221,1092.909991,2.410485014,-1.599071827e-005,-1.047414069e-
008,
        4.62479521e-012,-4.448865218e-017},
  bhigh={43360.7217,9.677195599999999},
  R=141.6594029311643);

constant IdealGases.Common.DataRecord Niplus (
  name="Niplus",
  MM=0.0586928514,
  Hf=19978490.48104008,
  H0=105733.0296956743,
  Tlimit=1000,
```

```

alow={-89693.86030000002,1173.6015,-3.41062041,0.01390739137,-1.501714923e-
005,
    7.8963379e-009,-1.648686761e-012},
blow={134558.95,40.3149516},
ahigh={-3961999.32,10170.84853,-6.02933129,0.002770858029,-8.9020777e-008,-
5.54100058e-011,
    5.235342833e-015},
bhigh={73403.9512,71.37503100000001},
R=141.6607270165784);

constant IdealGases.Common.DataRecord Niminus (
  name="Niminus",
  MM=0.0586939486,
  Hf=5311695.062887626,
  H0=105754.6331105078,
  Tlimit=1000,
  alow={-84376.2475,1135.476552,-3.38061583,0.01423003786,-1.582586302e-005,
    8.608840410000001e-009,-1.875316029e-012},
  blow={31243.0759,39.980613},
  ahigh={-543342.48,1182.64533,2.12644124,3.73045594e-005,6.95360843e-009,-
1.945719381e-012,
    1.271571579e-016},
  bhigh={28547.59122,10.43462235},
  R=141.6580788705022);

constant IdealGases.Common.DataRecord NiCL (
  name="NiCL",
  MM=0.09414640000000001,
  Hf=1933201.906817467,
  H0=100534.0724658617,
  Tlimit=1000,
  alow={-23579.97085,220.549163,2.714260287,0.00445359847,-2.849162229e-006,-
1.691898007e-010,
    5.15669926e-013},
  blow={19572.2949,14.23643068},
  ahigh={-3905769.19,9961.53399,-3.83943234,0.002767979391,-7.00595938e-008,-
5.74140366e-011,
    5.3076358e-015},
  bhigh={-45053.9705,67.69402548000001},
  R=88.31428498593679);

constant IdealGases.Common.DataRecord NiCL2 (
  name="NiCL2",
  MM=0.1295994,
  Hf=-570457.8879223206,
  H0=109618.9179888179,
  Tlimit=1000,
  alow={71099.7653,-1218.958288,12.08750596,-0.00867440314,1.043608104e-005,-
6.29849300999999e-009,
    1.485857653e-012},
  blow={-5006.998710000001,-34.99472036},
  ahigh={158588.9817,-1161.488738,9.60801537999999,-0.000946122450999999,
    2.608172043e-007,-3.26463725e-011,1.525950893e-015},
  bhigh={-4578.98554,-22.12397347},
  R=64.15517355790227);

constant IdealGases.Common.DataRecord NiO (

```

---

**1148 Modelica.Media.IdealGases.Common.SingleGasesData**

---

```
name="NiO",
MM=0.0746928,
Hf=3977143.713985819,
H0=118707.1444637234,
Tlimit=1000,
alow={23206.30462,-190.7136094,3.19017862,0.00529138349,-8.080210429999999e-
006,
      5.83981276e-009,-1.639499596e-012},
blow={35767.1112,7.84126567},
ahigh={-72340.9194,-80.3555427,4.55922102,2.30515451e-005,
      5.139832819999999e-009,-5.85347491e-013,2.697077329e-017},
bhigh={34611.4385,1.209414814},
R=111.3155752629437);

constant IdealGases.Common.DataRecord Nis (
  name="Nis",
  MM=0.0907583999999999,
  Hf=3938136.85565193,
  H0=101572.1630174177,
  Tlimit=1000,
  alow={-12724.88974,143.2801381,2.471951556,0.00550527112,-3.41835528e-006,-
3.099052241e-010,
      6.26815467e-013},
  blow={41177.1608,15.17981835},
  ahigh={-735812.898,1178.231095,4.83786558,-0.000393120139,1.341974852e-007,
      -1.569591708e-011,6.67704506e-016},
  bhigh={32755.6594,3.690652283},
  R=91.6104647062973);

constant IdealGases.Common.DataRecord O (
  name="O",
  MM=0.0159994,
  Hf=15574021.71331425,
  H0=420353.4507544033,
  Tlimit=1000,
  alow={-7953.611300000001,160.7177787,1.966226438,0.00101367031,-
1.110415423e-006,
      6.5175075e-010,-1.584779251e-013},
  blow={28403.62437,8.40424181999999},
  ahigh={261902.0262,-729.872203,3.31717727,-0.000428133436,1.036104594e-007,
      -9.438304329999999e-012,2.725038297e-016},
  bhigh={33924.2806,-0.667958535},
  R=519.6739877745415);

constant IdealGases.Common.DataRecord Oplus (
  name="Oplus",
  MM=0.0159988514,
  Hf=98056240.96239808,
  H0=387367.0581126842,
  Tlimit=1000,
  alow={0,0,2.5,0,0,0,0},
  blow={187935.2842,4.39337676},
  ahigh={-216651.3208,666.545615,1.702064364,0.000471499281,-1.427131823e-007,
      2.016595903e-011,-9.10715776199999e-016},
  bhigh={183719.1966,10.05690382},
  R=519.691807375622);
```

```

constant IdealGases.Common.DataRecord Ominus(
  name="Ominus",
  MM=0.0159999486,
  Hf=6365407.448871429,
  H0=410675.8192960695,
  Tlimit=1000,
  allow={-5695.857110000001, 109.9287334, 2.184719661, 0.0005326359799999999, -
5.298878440000001e-007,
  2.870216236e-010, -6.52469274e-014},
  blow={10932.87498, 6.72986386},
  ahighe={9769.363179999999, 7.15960478, 2.494961726, 1.968240938e-006, -
4.30417485e-010,
  4.912083080000001e-014, -2.271600083e-018},
  bhigh={11495.54438, 4.83703644},
  R=519.6561693954442);

constant IdealGases.Common.DataRecord OD(
  name="OD",
  MM=0.018013502,
  Hf=1952535.825626799,
  H0=0,
  Tlimit=1000,
  allow={21186.91536, -278.598236, 5.45621012, -0.00614811983,
  9.117670560000001e-006, -5.527812710000001e-009, 1.239794711e-012},
  blow={0, 4464.87825},
  ahighe={8999.108, 783247.316, -2532.992554, 5.95212465, -0.000374359528,
  4.95952762e-008, 3.45445473e-012},
  bhigh={-7.3806268e-016, 0},
  R=461.5688831633072);

constant IdealGases.Common.DataRecord ODminus(
  name="ODminus",
  MM=0.0180140506,
  Hf=-8191266.876978795,
  H0=0,
  Tlimit=1000,
  allow={56061.2832, -751.415615, 7.54418847, -0.01092083064, 1.503688938e-005, -
9.35166584e-009,
  2.246466046e-012},
  blow={0, -15157.0737},
  ahighe={8642.103999999999, 302946.7029, -1079.684654, 4.21141738, 0.000626083007,
  -2.411665054e-007, 4.248425540000001e-011},
  bhigh={-2.387099102e-015, -11889.35343},
  R=461.5548265418994);

constant IdealGases.Common.DataRecord OH(
  name="OH",
  MM=0.01700734,
  Hf=2191889.266634288,
  H0=518194.2620068747,
  Tlimit=1000,
  allow={-1998.85899, 93.0013616, 3.050854229, 0.001529529288, -3.157890998e-006,
  3.31544618e-009, -1.138762683e-012},
  blow={2991.214235, 4.67411079},
  ahighe={1017393.379, -2509.957276, 5.11654786, 0.000130529993, -
8.28432225999999e-008,
  2.006475941e-011, -1.556993656e-015},
  bhigh={20196.40206, -11.01282337},

```

---

**1150 Modelica.Media.IdealGases.Common.SingleGasesData**

---

```
R=488.8755090449183);

constant IdealGases.Common.DataRecord OHplus(
  name="OHplus",
  MM=0.0170067914,
  Hf=76393787.19021624,
  H0=505862.7931427441,
  Tlimit=1000,
  allow={60316.3086,-757.35203,7.30775293,-0.00950688167,1.202555795e-005,-
6.8290261e-009,
  1.501588659e-012},
  blow={158926.2158,-19.50106996},
  ahighelement 1={504072.91,-1380.052958,4.1254622,0.000833194884,-3.44285629e-007,
  6.792853949999999e-011,-4.36387213e-015},
  bhighelement 1={164383.9235,-3.99705849},
  R=488.8912790451466);

constant IdealGases.Common.DataRecord OHminus (
  name="OHminus",
  MM=0.0170078886,
  Hf=-8540519.015393833,
  H0=506006.6068400753,
  Tlimit=1000,
  allow={29108.80827,-321.690494,4.85102905,-0.002579035357,2.004980024e-006,-
7.956852959999999e-011,
  -2.320495634e-013},
  blow={-16888.86234,-7.1215913},
  ahighelement 1={471133.117,-857.233669,3.16618121,0.001233581296,-3.99924458e-007,
  6.23908167e-011,-3.35434322e-015},
  bhighelement 1={-12248.49139,1.48773626},
  R=488.8597400620322);

constant IdealGases.Common.DataRecord O2 (
  name="O2",
  MM=0.0319988,
  Hf=0,
  H0=271263.4223783392,
  Tlimit=1000,
  allow={-34255.6342,484.700097,1.119010961,0.00429388924,-6.83630052e-007,-
2.0233727e-009,
  1.039040018e-012},
  blow={-3391.45487,18.4969947},
  ahighelement 1={-1037939.022,2344.830282,1.819732036,0.001267847582,-2.188067988e-
007,
  2.053719572e-011,-8.193467050000001e-016},
  bhighelement 1={-16890.10929,17.38716506},
  R=259.8369938872708);

constant IdealGases.Common.DataRecord O2plus(
  name="O2plus",
  MM=0.0319982514,
  Hf=36621639.76872811,
  H0=290988.1506837588,
  Tlimit=1000,
  allow={-86072.0545,1051.875934,-0.543238047,0.00657116654,-3.27426375e-006,
  5.940645339999999e-011,3.23878479e-013},
  blow={134554.4668,29.0270975},
```

```

ahigh={73846.5488,-845.955954,4.98516416,-0.000161101089,6.42708399e-008,-
1.504939874e-011,
       1.578465409e-015},
bhigh={144632.1044,-5.81123065},
R=259.841448711163);

constant IdealGases.Common.DataRecord O2minus(
  name="O2minus",
  MM=0.0319993486,
  Hf=-1500903.084008404,
  H0=292181.2914654145,
  Tlimit=1000,
  allow={18838.74344,114.9551768,1.518876821,0.00801611138,-9.850571029999999e-
006,
         6.04419621e-009,-1.486439845e-012},
  blow={-7101.53876,15.0121038},
  ahigh={-56552.0805,-236.7815862,4.67583367,-2.1972453e-005,1.71150928e-008,
         -1.757645062e-012,8.24817279e-017},
  bhigh={-5960.17775,-2.436885556},
  R=259.8325392161265);

constant IdealGases.Common.DataRecord O3 (
  name="O3",
  MM=0.0479982,
  Hf=2954277.45207112,
  H0=215972.786479493,
  Tlimit=1000,
  allow={-12823.14507,589.8216640000001,-2.547496763,0.02690121526,-
3.52825834e-005,
         2.312290922e-008,-6.04489327e-012},
  blow={13483.68701,38.5221858},
  ahigh={-38696624.8,102334.4994,-89.615516,0.0370614497,-4.13763874e-006,-
2.725018591e-010,
         5.24818811e-014},
  bhigh={-651791.818,702.9109520000001},
  R=173.2246625915139);

constant IdealGases.Common.DataRecord P (
  name="P",
  MM=0.030973761,
  Hf=10218326.40860114,
  H0=200086.389250566,
  Tlimit=1000,
  allow={50.4086657,-0.763941864999999,2.504563992,-1.381689958e-005,
         2.245585515e-008,-1.866399889e-011,6.227063395e-015},
  blow={37324.2191,5.359303481},
  ahigh={1261794.642,-4559.83819,8.91807931,-0.00438140146,1.454286224e-006,-
2.030782763e-010,
         1.021022887e-014},
  bhigh={65417.2395999999,-39.15974795},
  R=268.4359836056074);

constant IdealGases.Common.DataRecord Pplus (
  name="Pplus",
  MM=0.0309732124,
  Hf=43148687.41222335,
  H0=262867.7934614235,

```

---

**1152 Modelica.Media.IdealGases.Common.SingleGasesData**

---

```
Tlimit=1000,
allow={-73169.0811,962.7979140000001,-0.369393805,0.00476677834000001,-
4.57476858e-006,
2.371262331e-009,-5.131314899999999e-013},
blow={154940.6849,23.76640762},
ahigh={-559424.936,1722.576545,0.843038108999999,0.000628736894000001,-
6.317195459999999e-008,
-1.810842484e-012,4.31811257e-016},
bhigh={149049.3431,18.45275207},
R=268.4407381650862);

constant IdealGases.Common.DataRecord Pminus(
  name="Pminus",
  MM=0.0309743096,
  Hf=7710479.041637784,
  H0=217853.6047176335,
  Tlimit=1000,
  allow={-10089.49093,182.6468403,1.962456304,0.000919737753999999,-
9.21499863e-007,
  5.012872360000001e-010,-1.142677121e-013},
  blow={27030.82432,9.47813782200001},
  ahigh={15434.88016,9.49500933999999,2.493170861,2.700167711e-006,-
5.94921039e-010,
  6.82376629e-014,-3.16695991e-018},
  bhigh={27975.72693,6.246793872},
  R=268.4312292145488);

constant IdealGases.Common.DataRecord PCL(
  name="PCL",
  MM=0.066426761,
  Hf=2026519.28189002,
  H0=139870.0141348153,
  Tlimit=1000,
  allow={34888.617,-560.3119340000001,6.26456848,-0.00318336437,
  3.49532684e-006,-2.091034291e-009,5.4187586e-013},
  blow={17746.53764,-8.074572148},
  ahigh={-347168.151,993.555793,3.39943027,0.00040197117,
  8.295588390000001e-008,-3.090212484e-011,2.106384858e-015},
  bhigh={8433.59115,10.64902238},
  R=125.1675059092525);

constant IdealGases.Common.DataRecord PCL2(
  name="PCL2",
  MM=0.101879761,
  Hf=-532906.1676931104,
  H0=120227.5788613207,
  Tlimit=1000,
  allow={51900.845,-1060.9042,10.5854994,-0.00687991753,7.62077318e-006,-
4.53011547e-009,
  1.11667166e-012},
  blow={-3220.33118,-27.53518359},
  ahigh={-83498.33970000001,-26.2524947,7.02036181,-8.388768680000001e-006,
  1.89678474e-009,-2.21487992e-013,1.04165355e-017},
  bhigh={-8742.99368,-6.23402319},
  R=81.61063510936191);

constant IdealGases.Common.DataRecord PCL2minus(
```

```

name="PCL2minus",
MM=0.1018803096,
Hf=-3497096.292687356,
H0=122234.0906588686,
Tlimit=1000,
alow={49348.3277,-960.159054,9.8847697,-0.00495020295,4.94621787e-006,-
2.678675201e-009,
6.07672361e-013},
blow={-39980.724,-24.13131695},
ahigh={-85649.8676,-31.8555594,7.02449496,-1.002251172e-005,
2.253747944e-009,-2.619657192e-013,1.227059658e-017},
bhigh={-45038.1143,-6.721522361},
R=81.6101956564922);

constant IdealGases.Common.DataRecord PCL3(
name="PCL3",
MM=0.137332761,
Hf=-2108018.493853772,
H0=116013.585425549,
Tlimit=1000,
alow={77177.4547,-1617.65086,15.3608047,-0.0101107705,1.10340834e-005,-
6.47612102e-009,
1.57915114e-012},
blow={-29558.9364,-52.44338688},
ahigh={-133307.693,-41.4588321,10.0318606,-1.30194616e-005,2.92291689e-009,
-3.39186924e-013,1.58642553e-017},
bhigh={-38003.3668,-20.50737678},
R=60.54252415416013);

constant IdealGases.Common.DataRecord PCL5(
name="PCL5",
MM=0.208238761,
Hf=-1805619.6559871,
H0=111917.0412274975,
Tlimit=1000,
alow={102907.02,-2515.70746,24.3011497,-0.0156461069,1.70987303e-005,-
1.00601143e-008,
2.46015879e-012},
blow={-37225.8237,-98.16335170000001},
ahigh={-225590.689,-69.38542820000001,16.0535963,-2.20215696e-005,
4.97139229e-009,-5.79986005e-013,2.72597077e-017},
bhigh={-50342.3422,-48.7245487},
R=39.92759061796377);

constant IdealGases.Common.DataRecord PF(
name="PF",
MM=0.0499721642,
Hf=-959429.9900263273,
H0=177921.1915740883,
Tlimit=1000,
alow={22278.66728,-172.7508706,3.085366934,0.00548971823,-
8.24797253999999e-006,
5.86888291e-009,-1.617252113e-012},
blow={-5809.27965,7.70971885},
ahigh={-1132572.282,3200.03855,0.778050237,0.001969518902,-4.36581897e-007,
5.243648570000001e-011,-2.618755712e-015},
bhigh={-27667.73009,27.59312302},
R=166.3820675591232);

```

---

**1154 Modelica.Media.IdealGases.Common.SingleGasesData**

---

```
constant IdealGases.Common.DataRecord PFplus(
  name="PFplus",
  MM=0.0499716156,
  Hf=18040601.43294627,
  H0=188704.0850446308,
  Tlimit=1000,
  allow={-17791.38157, 390.999127, 1.257434156, 0.007861363749999999, -9.34231268e-006,
         5.630915650000001e-009, -1.370669924e-012},
  blow={105487.3882, 19.01676576},
  ahighelement1=-40115.2518, ahighelement2=-159.7504413, ahighelement3=4.62139748, ahighelement4=-2.378072183e-005,
        ahighelement5=1.431831329e-008, ahighelement6=-2.300109157e-012, ahighelement7=1.694069882e-016,
  bhigh={107863.3756, -0.04072343949999999},
  R=166.3838941400966);

constant IdealGases.Common.DataRecord PFminus (
  name="PFminus",
  MM=0.0499727128,
  Hf=-3282711.520115834,
  H0=190416.7788145374,
  Tlimit=1000,
  allow={7705.21095, -115.2432706, 4.25495231, 0.001163521526, -1.627947259e-006,
         1.084382815e-009, -2.809490132e-013},
  blow={-20355.58966, 2.217923793},
  ahighelement1=-130379.3388, ahighelement2=271.5526386, ahighelement3=4.13761981, ahighelement4=0.0002728610614, ahighelement5=-7.64441625e-008,
        ahighelement6=1.182073001e-011, ahighelement7=-6.083906640000001e-016},
  bhigh={-22907.99827, 3.7485261},
  R=166.3802410182543);

constant IdealGases.Common.DataRecord PFCL (
  name="PFCL",
  MM=0.0854251642,
  Hf=-3314996.484373161,
  H0=136147.0136922488,
  Tlimit=1000,
  allow={54901.8853, -865.867948, 7.94446765, 0.0009503779099999999, -3.33909021e-006,
         3.023977817e-009, -9.45037754e-013},
  blow={-31328.43217, -14.40426886},
  ahighelement1=-128214.3211, ahighelement2=-97.3904355, ahighelement3=7.07266166, ahighelement4=-2.903614134e-005,
        ahighelement5=6.408504490000001e-009, ahighelement6=-7.33899377e-013, ahighelement7=3.39672317e-017},
  bhigh={-36006.4926, -7.626493035},
  R=97.33047724127172);

constant IdealGases.Common.DataRecord PFCLminus (
  name="PFCLminus",
  MM=0.0854257128,
  Hf=-6195665.094877616,
  H0=137462.042927197,
  Tlimit=1000,
  allow={89343.3685, -1266.972513, 9.912194530000001, -0.0038779283,
         3.036692411e-006, -1.29655938e-009, 2.32422658e-013},
  blow={-58945.0571, -26.1705046},
  ahighelement1=-123156.515, ahighelement2=-97.6586111, ahighelement3=7.07402957, ahighelement4=-2.997070692e-005,
        ahighelement5=6.685280620000001e-009, ahighelement6=-7.7222622e-013, ahighelement7=3.5993823e-017},
  bhigh={-65588.1722, -8.208878059},
  R=97.32985218942183);
```

```

constant IdealGases.Common.DataRecord PFCL2 (
  name="PFCL2",
  MM=0.1208781642,
  Hf=-4235052.239484623,
  H0=122997.8970842114,
  Tlimit=1000,
  allow={82665.60560000001,-1512.138928,13.03404207,-0.002903247162,
    7.83665297e-007,6.45834066e-010,-3.73764182e-013},
  blow={-56442.3787,-41.00905668},
  ahigh={-186458.9813,-114.8004379,10.08599754,-3.44745652e-005,
    7.62797201e-009,-8.75309743e-013,4.05778835e-017},
  bhigh={-64504.7219,-21.74054563},
  R=68.7839036523025);

constant IdealGases.Common.DataRecord PFCL4 (
  name="PFCL4",
  MM=0.1917841642,
  Hf=-3311099.44686455,
  H0=113324.5807372035,
  Tlimit=1000,
  allow={141207.1573,-2882.47108,23.87345233,-0.01239510207,1.141290085e-005,-
5.705483820000001e-009,
    1.195229703e-012},
  blow={-66134.7865,-98.5969355},
  ahigh={-295437.588,-132.5154294,16.1000538,-4.03535917e-005,
    8.971130739999999e-009,-1.03328881e-012,4.80446835e-017},
  bhigh={-81357.1939999999,-50.62403920000001},
  R=43.35327702723852);

constant IdealGases.Common.DataRecord PF2 (
  name="PF2",
  MM=0.0689705673999999,
  Hf=-7439458.284056397,
  H0=160669.2161271099,
  Tlimit=1000,
  allow={55247.6572,-628.05178,5.13743792,0.009120673350000001,-1.46877376e-
005,
    1.08126845e-008,-3.06494449e-012},
  blow={-59775.4147,-1.689589472},
  ahigh={-170846.712,-169.205154,7.1254916,-4.990852849999999e-005,
    1.09733523e-008,-1.25287026e-012,5.78479114e-017},
  bhigh={-63382.9584,-10.41715673},
  R=120.5510163745587);

constant IdealGases.Common.DataRecord PF2minus (
  name="PF2minus",
  MM=0.068971116,
  Hf=-10284565.30991901,
  H0=160133.0185812856,
  Tlimit=1000,
  allow={140479.3057,-1699.945753,10.36823971,-0.00359958769,1.9619258e-006,-
3.82932987e-010,
    -3.37554952e-014},
  blow={-78104.34510000001,-32.17375182},
  ahigh={-165397.6056,-164.0604822,7.12409152,-5.0154314e-005,
    1.117302152e-008,-1.289308601e-012,6.00470667e-017},
  bhigh={-87002.2331999999,-11.09074866},
  R=120.5500575052316);

```

---

**1156 Modelica.Media.IdealGases.Common.SingleGasesData**

---

```
constant IdealGases.Common.DataRecord PF2CL(
  name="PF2CL",
  MM=0.1044235674,
  Hf=-7039375.308681516,
  H0=132884.9257452202,
  Tlimit=1000,
  allow={83975.9016,-1318.075883,10.17203515,0.005571231219999999,-
1.100920478e-005,
  8.72638318e-009,-2.567714099e-012},
  blow={-83816.56169999999,-27.66012951},
  ahighelement1=-240928.1185,-203.348467,10.15150769,-6.04836731e-005,
  1.333942001e-008,-1.526772528e-012,7.063339179999999e-017},
  bhighelement1=-91011.1059,-24.17825549},
  R=79.62256229143155);

constant IdealGases.Common.DataRecord PF2CL3(
  name="PF2CL3",
  MM=0.1753295674,
  Hf=-5011961.045881188,
  H0=115013.3334555869,
  Tlimit=1000,
  allow={184209.9543,-3334.40239,23.88510873,-0.01030943175,
  7.356504219999999e-006,-2.498621785e-009,2.513997646e-013},
  blow={-92795.49890000001,-102.7991502},
  ahighelement1=-365527.254,-202.969416,16.15238015,-6.11879657e-005,
  1.355610375e-008,-1.55712964e-012,7.22435997e-017},
  bhighelement1=-110494.1533,-53.87308191},
  R=47.42196152820714);

constant IdealGases.Common.DataRecord PF3(
  name="PF3",
  MM=0.0879689706,
  Hf=-10883383.00959952,
  H0=147057.4557342837,
  Tlimit=1000,
  allow={84809.84480000001,-1102.340691,7.16071511,0.01438278338,-2.318137222e-
005,
  1.702244315e-008,-4.81099195e-012},
  blow={-111183.7495,-14.57652362},
  ahighelement1=-295704.0675,-299.7689931,10.22297023,-8.890263450000001e-005,
  1.958901359e-008,-2.240491075e-012,1.035952523e-016},
  bhighelement1=-117374.3464,-27.77419177},
  R=94.51596333673592);

constant IdealGases.Common.DataRecord PF3CL2(
  name="PF3CL2",
  MM=0.1588749706,
  Hf=-7062301.328916815,
  H0=119887.8837117469,
  Tlimit=1000,
  allow={182249.7206,-3089.610245,20.67710388,-0.001168732391,-5.02054074e-006,
  5.80457335e-009,-1.965367906e-012},
  blow={-122812.0667,-86.27977355},
  ahighelement1=-422383.8050000001,-320.796522,16.24013,-9.62365531e-005,
  2.129279712e-008,-2.443515401e-012,1.13290435e-016},
  bhighelement1=-139264.2151,-55.31285745},
  R=52.33342903919946);
```

```

constant IdealGases.Common.DataRecord PF4CL(
  name="PF4CL",
  MM=0.1424203738,
  Hf=-9583666.708505718,
  H0=128637.5292465354,
  Tlimit=1000,
  allow={180424.0323,-2741.096618,16.80192838,0.00991931525,-2.025636171e-005,
         1.617979147e-008,-4.77261152e-012},
  blow={-153238.2587,-66.79119965},
  ahighe={-478114.004,-426.481283,16.31812121,-0.0001271374903,
          2.806715254e-008,-3.2151663e-012,1.488512118e-016},
  bhigh={-168048.8362,-57.35844805},
  R=58.37979341127105);

constant IdealGases.Common.DataRecord PF5(
  name="PF5",
  MM=0.125965777,
  Hf=-12648673.61553289,
  H0=131286.5239580112,
  Tlimit=1000,
  allow={185577.8473,-2436.979828,12.01012406,0.02339343049,-3.81394441e-005,
         2.796136412e-008,-7.87070488e-012},
  blow={-181456.6568,-44.8648874},
  ahighe={-566872.784,-667.791791,16.49789569,-0.000199009944,4.3950071e-008,-
          5.03684954e-012,
         2.332945909e-016},
  bhigh={-194432.1623,-62.61721040000001},
  R=66.00580092480199);

constant IdealGases.Common.DataRecord PH(
  name="PH",
  MM=0.031981701,
  Hf=7215129.176525038,
  H0=270407.881056733,
  Tlimit=1000,
  allow={22736.33198,-397.267406,6.23369766,-0.0091817846,1.523328123e-005,-
          1.085888585e-008,
         2.929760547e-012},
  blow={28527.68404,-10.95191197},
  ahighe={781473.0649999999,-3038.451204,7.46748102,-0.001837522255,
          7.1659477e-007,-1.142128853e-010,6.175410560000001e-015},
  bhigh={45362.6018,-24.6729814},
  R=259.9759156024878);

constant IdealGases.Common.DataRecord PH2(
  name="PH2",
  MM=0.032989641,
  Hf=3623970.021377317,
  H0=302566.8269624395,
  Tlimit=1000,
  allow={15552.68372,-184.1602025,4.89589604,-0.0034954366,1.053418945e-005,-
          8.377562920000002e-009,
         2.27076615e-012},
  blow={14098.39468,-2.210564792},
  ahighe={1127884.913,-4715.238249999999,10.214983,-0.00116757382,
          2.150542671e-007,-1.624213739e-011,3.76622524e-016},
  bhigh={41830.7463,-42.3162325},
  R=252.0328123607044);

```

```
constant IdealGases.Common.DataRecord PH2minus(
  name="PH2minus",
  MM=0.0329901896,
  Hf=-280853.735984591,
  H0=301910.4200601502,
  Tlimit=1000,
  allow={69506.84490000001,-771.916291,7.38353762,-0.00852294721,
         1.474476507e-005,-9.84716081e-009,2.413107871e-012},
  blow={1582.29585,-17.61306419},
  ahighelement1={1382525.815,-5213.40067,10.21694832,-0.001116316349,2.215004171e-007,
             -2.338005187e-011,1.015580932e-015},
  bhighelement1={29906.83733,-43.78711584999999},
  R=252.0286212601821);

constant IdealGases.Common.DataRecord PH3(
  name="PH3",
  MM=0.033997581,
  Hf=159981.9704819587,
  H0=298157.1541810578,
  Tlimit=1000,
  allow={-6384.32534,405.756741,-0.1565680086,0.01338380613,-8.27539143e-006,
         3.024360831e-009,-6.421764630000001e-013},
  blow={-2159.842124,23.85561888},
  ahighelement1={1334801.106,-6725.46352,14.45857073,-0.001639736883,3.40921857e-007,
               -3.73627208e-011,1.672947506e-015},
  bhighelement1={39103.2571,-71.9878119},
  R=244.5606938917213);

constant IdealGases.Common.DataRecord PN(
  name="PN",
  MM=0.044980461,
  Hf=3812484.380718108,
  H0=193464.1132290752,
  Tlimit=1000,
  allow={-51032.0384,820.2926679999999,-1.392772765,0.01287989789,-
1.401425371e-005,
         7.775633459999999e-009,-1.75153933e-012},
  blow={15732.2652,32.51070633},
  ahighelement1={-249562.5593,176.043883,4.14412196,0.0002478018097,-
5.674896300000001e-008,
               4.263645119999999e-012,3.063920924e-016},
  bhighelement1={17703.17267,1.325517397},
  R=184.8463047099495);

constant IdealGases.Common.DataRecord PO(
  name="PO",
  MM=0.046973161,
  Hf=-593055.4045532512,
  H0=199903.7918695742,
  Tlimit=1000,
  allow={-68457.54059999999,1141.295708,-2.77955606,0.01678458047,-
1.974879516e-005,
         1.19260232e-008,-2.927460912e-012},
  blow={-9847.74504,41.84328297},
  ahighelement1={-336666.744,622.9355840000001,3.56560546,0.0006516620719999999,-
2.061770841e-007,
               3.18441323e-011,-1.573691908e-015},
  bhighelement1={-8939.79039,6.954859188},
```

```

R=177.0047368113038);

constant IdealGases.Common.DataRecord P0minus(
  name="P0minus",
  MM=0.0469737096,
  Hf=-2981818.876829775,
  H0=186872.743812424,
  Tlimit=1000,
  allow={54343.46320000001,-412.362355,3.74648388,0.00385863647,-5.96334304e-
006,
        4.278286360000001e-009,-1.139722989e-012},
  blow={-15558.10937,3.404436468},
  ahigh={-848.0294779999999,347.054487,2.877890167,0.001622182445,-
4.87019099e-007,
       6.64575227e-011,-3.39570376e-015},
  bhigh={-19872.0954,10.75444355},
  R=177.0026695954198);

constant IdealGases.Common.DataRecord POCL3 (
  name="POCL3",
  MM=0.153332161,
  Hf=-3706984.864056015,
  H0=115539.7268548246,
  Tlimit=1000,
  allow={47937.8816,-1270.38828,13.6771646,0.0010826721,-1.73610406e-006,
         8.460890640000001e-010,-1.17466133e-013},
  blow={-65075.7188,-43.14705941},
  ahigh={-218593.797,-508.026968,13.3778465,-0.000151073105,3.34098678e-008,-
3.83532976e-012,
        1.77932988e-016},
  bhigh={-70093.15090000001,-39.84746301},
  R=54.2252319785671);

constant IdealGases.Common.DataRecord POFCL2 (
  name="POFCL2",
  MM=0.1368775642,
  Hf=-5799995.906122357,
  H0=120210.9936436172,
  Tlimit=1000,
  allow={46940.6156,-1069.2426,10.71284279,0.00949118329999999,-1.306412424e-
005,
        8.426906020000001e-009,-2.140294969e-012},
  blow={-92748.94989999999,-28.15077719},
  ahigh={-270764.7503,-655.650227,13.48668421,-0.0001942765035,
         4.29118449e-008,-4.9218448e-012,2.282079808e-016},
  bhigh={-96543.92140000001,-41.64963494},
  R=60.74386294492521);

constant IdealGases.Common.DataRecord POF2CL (
  name="POF2CL",
  MM=0.1204229674,
  Hf=-8491794.06618741,
  H0=123700.323298959,
  Tlimit=1000,
  allow={52264.3816,-969.1037700000001,7.96363598,0.01755599796,-2.398693997e-
005,
        1.572376747e-008,-4.079746e-012},

```

---

**1160 Modelica.Media.IdealGases.Common.SingleGasesData**

---

```
blow={-120265.6893,-15.93967075},
ahigh={-334708.659,-817.6104810000001,13.6066432,-0.0002420426112,
      5.34371272e-008,-6.126461349999999e-012,2.839570116e-016},
bhigh={-123340.1194,-44.96103437000001},
R=69.04390565615641);

constant IdealGases.Common.DataRecord POF3 (
  name="POF3",
  MM=0.1039683706,
  Hf=-12042123.89570718,
  H0=136256.3337123223,
  Tlimit=1000,
  allow={29660.3071,-427.670575,3.6691461,0.0283949378,-3.768318e-005,
         2.44910454e-008,-6.34197287e-012},
  blow={-150112.453,5.160417996},
  ahigh={-369060.242,-1005.55254,13.7452994,-0.000297103711,6.55452664e-008,-
7.50998733e-012,
        3.47896991e-016},
  bhigh={-149976.173,-48.82214181000001},
  R=79.97116769280214);

constant IdealGases.Common.DataRecord PO2 (
  name="PO2",
  MM=0.062972561,
  Hf=-4470633.439856448,
  H0=166943.5994511959,
  Tlimit=1000,
  allow={-63726.9822,1036.741044,-2.877797967,0.02278134083,-2.567920328e-005,
         1.465060412e-008,-3.3879967e-012},
  blow={-39935.4472,44.2530938},
  ahigh={492621.0990000001,-2605.465745,9.51760561,-0.001180371565,
        2.532912819e-007,-1.789964539e-011,1.800381054e-016},
  bhigh={-20288.84763,-29.69743125},
  R=132.0332517522989);

constant IdealGases.Common.DataRecord PO2minus (
  name="PO2minus",
  MM=0.06297310960000001,
  Hf=-9490141.979585521,
  H0=168544.1463414727,
  Tlimit=1000,
  allow={27024.23135,29.39956284,1.162686114,0.01599897972,-1.979229263e-005,
         1.208847527e-008,-2.95664048e-012},
  blow={-72859.4627,19.63118182},
  ahigh={1628679.152,-5989.19998,13.69318836,-0.00362384379,
        9.488198340000001e-007,-1.040308759e-010,4.01903366e-015},
  bhigh={-36859.595,-59.5679622},
  R=132.0321015241718);

constant IdealGases.Common.DataRecord PS (
  name="PS",
  MM=0.063038761,
  Hf=2386330.467377048,
  H0=152542.9092745018,
  Tlimit=1000,
  allow={-768.9786800000001,-46.3782407,4.10464917,0.001555262273,-
2.315757288e-006,
```

```

  1.661425178e-009,-4.6312384e-013},
blow={17078.75808,4.230652171},
ahigh={-270272.9081,888.354822,3.16919012,0.001022480817,-3.80374048e-007,
  7.01986187999999e-011,-4.26912231e-015},
bhigh={11215.10462,11.47334049},
R=131.8945973573307);

constant IdealGases.Common.DataRecord P2(
  name="P2",
  MM=0.061947522,
  Hf=2324548.187738647,
  H0=143736.2902102848,
  Tlimit=1000,
  allow={30539.2251,-324.617759,4.02246381,0.00323209479,-5.511052450000001e-
006,
  4.19557293e-009,-1.21503218e-012},
blow={17969.1087,1.645350331},
ahigh={-780693.649,2307.91087,1.41174313,0.00210823742,-7.36085662e-007,
  1.25936012e-010,-7.07975249e-015},
bhigh={1329.82474,21.69741365},
R=134.2179918028037);

constant IdealGases.Common.DataRecord P203(
  name="P203",
  MM=0.109945722,
  Hf=-6227120.633215725,
  H0=146389.3338205556,
  Tlimit=1000,
  allow={-66457.5389,758.090055,0.988843677,0.0285987873,-3.23114211e-005,
  1.83452517e-008,-4.2124893e-012},
blow={-88200.3674,26.89914458},
ahigh={-217534.411,-1369.902,14.0114296,-0.000402385696,8.86834578e-008,-
1.01565728e-011,
  4.70435606e-016},
bhigh={-79227.5699999999,-47.39487232000001},
R=75.62342443846974);

constant IdealGases.Common.DataRecord P204(
  name="P204",
  MM=0.125945122,
  Hf=-7413980.090471467,
  H0=137564.5179810934,
  Tlimit=1000,
  allow={-43760.9126,490.510283,0.964364937,0.0388607358,-4.67550922e-005,
  2.81262563e-008,-6.8244061e-012},
blow={-116899.779,23.79582767},
ahigh={-367011.486,-1638.42373,17.2118861,-0.000482658015,1.0644619e-007,-
1.21961007e-011,
  5.65065458e-016},
bhigh={-109043.593,-67.14276593},
R=66.01662587614946);

constant IdealGases.Common.DataRecord P205(
  name="P205",
  MM=0.141944522,
  Hf=-7921195.817616688,
  H0=155341.7116019454,

```

---

**1162 Modelica.Media.IdealGases.Common.SingleGasesData**

---

```
Tlimit=1000,
alow={-29991.5922,-45.9450659,6.98748683,0.0288330297,-3.10450953e-005,
      1.64386868e-008,-3.4679423e-012},
blow={-138190.14,-3.400572711},
ahigh={-324708.016,-2016.97508,20.4869934,-0.000591064115,1.30199302e-007,-
1.49068294e-011,
      6.90357341e-016},
bhigh={-130629.016,-80.31642952},
R=58.57550459044837);

constant IdealGases.Common.DataRecord P3(
  name="P3",
  MM=0.09292128300000001,
  Hf=2259977.404745907,
  H0=129201.8535732013,
  Tlimit=1000,
  alow={46933.5895,-864.358932,8.734247529999999,1.09453456e-005,-1.99857332e-
006,
      2.09675051e-009,-6.92288971e-013},
  blow={27748.4734,-20.63587261},
  ahigh={-125351.306,-83.0673462,7.56195677,-2.47440395e-005,5.45727693e-009,
      -6.24466057e-013,2.88772422e-017},
  bhigh={23087.2228,-12.28310621},
  R=89.47866120186912);

constant IdealGases.Common.DataRecord P306(
  name="P306",
  MM=0.188917683,
  Hf=-8340569.310285264,
  H0=124071.403098883,
  Tlimit=1000,
  alow={201449.3813,-3053.488073,16.02114098,0.043168899,-6.37654426e-005,
      4.42286792e-008,-1.200020334e-011},
  blow={-177650.3811,-65.5626789},
  ahigh={-878796.8100000001,-1651.988071,27.72258397,-0.00048593267,
      1.067867708e-007,-1.218015776e-011,5.61491196e-016},
  bhigh={-190916.5548,-122.0665277},
  R=44.01108391743297);

constant IdealGases.Common.DataRecord P4(
  name="P4",
  MM=0.123895044,
  Hf=475402.3897840498,
  H0=113875.8060411198,
  Tlimit=1000,
  alow={120514.185,-2345.71179,17.7394655,-0.0145631497,1.58759409e-005,-
9.314710310000001e-009,
      2.27149167e-012},
  blow={16088.4648,-70.88590647000001},
  ahigh={-185870.586,-62.8454689,10.0484488,-1.98766885e-005,
      4.482408440000001e-009,-5.22610367e-013,2.45579129e-017},
  bhigh={3848.21092,-24.77297797},
  R=67.10899590140184);

constant IdealGases.Common.DataRecord P406(
  name="P406",
  MM=0.219891444,
```

```

Hf=-7303603.863731961,
H0=112771.659273837,
Tlimit=1000,
alow={376089.475,-5685.83262,29.1422545,0.0225153212,-4.51026963e-005,
      3.58831269e-008,-1.05757206e-011},
blow={-168856.248,-145.1359851},
ahigh={-1008997.24,-887.275399,28.6606483,-0.000263601894,5.81094072e-008,-
6.64808696e-012,
      3.07439193e-016},
bhigh={-199713.89,-128.1853821},
R=37.81171221923487);

constant IdealGases.Common.DataRecord P407 (
  name="P407",
  MM=0.235890844,
  Hf=-8412569.183906095,
  H0=111798.8157268198,
  Tlimit=1000,
  alow={321858.696,-4871.44473,24.54417068,0.0401550955,-6.52419715e-005,
        4.75655418e-008,-1.332975491e-011},
  blow={-218451.7427,-118.1504792},
  ahigh={-1102947.375,-1511.077128,32.109795,-0.000437180376,
        9.516639610000001e-008,-1.075204176e-011,4.91102568e-016},
  bhigh={-242909.654,-147.3140859},
  R=35.24711624669926);

constant IdealGases.Common.DataRecord P408 (
  name="P408",
  MM=0.251890244,
  Hf=-9139750.874988236,
  H0=110946.4803249784,
  Tlimit=1000,
  alow={271932.6942,-4118.600390000001,20.28714235,0.0568541699,-8.40073963e-
005,
        5.823389060000001e-008,-1.57864716e-011},
  blow={-260453.9703,-94.0716207},
  ahigh={-1173177.311,-2200.632995,35.6301013,-0.000648984974,
        1.429623989e-007,-1.635821725e-011,7.571070759999999e-016},
  bhigh={-278384.3144,-167.963696},
  R=33.00831293807473);

constant IdealGases.Common.DataRecord P409 (
  name="P409",
  MM=0.267889644,
  Hf=-9757671.375307065,
  H0=110199.6873010888,
  Tlimit=1000,
  alow={219991.6395,-3336.46494,15.86413106,0.07402452330000001,-
0.0001034835052,
        6.94443832e-008,-1.84073345e-011},
  blow={-301874.2692,-69.9796767},
  ahigh={-1233696.783,-2917.032808,39.1775711,-0.000873640298,
        1.937914895e-007,-2.230510118e-011,1.037248818e-015},
  bhigh={-312963.7516,-189.717642},
  R=31.03692952012733);

constant IdealGases.Common.DataRecord P4010 (

```

```
name="P4010",
MM=0.283889044,
Hf=-10237180.71698463,
H0=109537.7002291078,
Tlimit=1000,
alow={167014.2268,-2540.2433,11.36853403,0.09137812159999999,-
0.0001231990626,
     8.08103223e-008,-2.106787011e-011},
blow={-341015.191,-46.4343134},
ahigh={-1337201.132,-3516.38233,42.6040331,-0.001037406404,2.287613089e-007,
      -2.620219946e-011,1.213529528e-015},
bhigh={-345951.505,-211.5498906},
R=29.28775229522418);

constant IdealGases.Common.DataRecord Pb (
  name="Pb",
  MM=0.2072,
  Hf=942084.942084942,
  H0=29910.36679536679,
  Tlimit=1000,
  alow={1213.382285,-19.06116019,2.619299546,-0.000382951961,6.68818045e-007,
        -6.06123108e-010,2.240022429e-013},
  blow={22820.96238,6.2013692},
  ahigh={-9084313.07,26726.7318,-26.26244039,0.01358282305,-2.685523566e-006,
         2.3524328e-010,-7.324114532e-015},
  bhigh={-148165.0666,215.4011624},
  R=40.12776061776062);

constant IdealGases.Common.DataRecord Pbplus (
  name="Pbplus",
  MM=0.2071994514,
  Hf=4425670.636693588,
  H0=29910.44598875806,
  Tlimit=1000,
  alow={9.48668904,-0.1134955793,2.500549799,-1.382464681e-006,
        1.906022991e-009,-1.368396828e-012,4.003528117e-016},
  blow={109543.8901,7.5388559},
  ahigh={1320690.183,-4096.04801,7.38910151,-0.002807751909,7.83099165e-007,-
9.31060091e-011,
        4.016371727e-015},
  bhigh={135434.7306,-27.22020908},
  R=40.12786686364751);

constant IdealGases.Common.DataRecord Pbminus (
  name="Pbminus",
  MM=0.2072005486,
  Hf=742671.4313245792,
  H0=29910.28760239489,
  Tlimit=1000,
  alow={0,0,2.5,0,0,0,0},
  blow={17762.2614,8.2351321},
  ahigh={0,0,2.5,0,0,0,0},
  bhigh={17762.2614,8.2351321},
  R=40.12765437243635);

constant IdealGases.Common.DataRecord PbBr (
  name="PbBr",
```

```

MM=0.287104,
Hf=225776.4503455194,
H0=35339.53549933125,
Tlimit=1000,
alow={-2393.487988,-45.7659121,4.69260828,-0.000384529063,5.8882135e-007,-
4.08479289e-010,
1.202515201e-013},
blow={6662.43477,5.99170551},
ahigh={-2581831.67,7143.49076,-2.65819364,0.00310772226,-4.63138154e-007,
1.095529211e-011,1.508555088e-015},
bhigh={-39704.4098,59.5923865},
R=28.95979157378512);

constant IdealGases.Common.DataRecord PbBr2 (
  name="PbBr2",
  MM=0.367008,
  Hf=-283122.0654590636,
  H0=40930.81622198971,
  Tlimit=1000,
  alow={-6172.76636,-67.8320763,7.27315614,-0.000599059612,7.35863461e-007,-
4.74360481e-010,
1.246922291e-013},
  blow={-14278.90796,-0.698799215},
  ahigh={-13204.21479,-1.143074947,7.00097254,-4.28447161e-007,
1.018044861e-010,-1.234016791e-014,5.97133765e-019},
  bhigh={-14622.1375,0.8868582650000001},
  R=22.65474322085622);

constant IdealGases.Common.DataRecord PbBr3 (
  name="PbBr3",
  MM=0.446912,
  Hf=-232732.5222862667,
  H0=44681.37575182587,
  Tlimit=1000,
  alow={-11644.2425,-155.5151665,10.62317321,-0.001361947117,1.668738791e-006,
-1.073686019e-009,2.81825034e-013},
  blow={-14782.08304,-14.43758579},
  ahigh={-27879.30766,-2.633265111,10.00223199,-9.80776141e-007,
2.326232632e-010,-2.81599446e-014,1.361283735e-018},
  bhigh={-15569.81182,-10.81765474},
  R=18.60427108692539);

constant IdealGases.Common.DataRecord PbBr4 (
  name="PbBr4",
  MM=0.5268160000000001,
  Hf=-346298.3489491587,
  H0=49109.15575836725,
  Tlimit=1000,
  alow={-11764.58199,-256.7166983,14.02223292,-0.002224117068,
2.716201994e-006,-1.743340164e-009,4.567359749999999e-013},
  blow={-24621.25551,-28.8202075},
  ahigh={-38802.1603,-4.38293377,13.00369823,-1.620061335e-006,
3.83413088e-010,-4.63396718e-014,2.237427066e-018},
  bhigh={-25923.27065,-22.87710636},
  R=15.78249711474215);

constant IdealGases.Common.DataRecord PbCL (

```

---

**1166 Modelica.Media.IdealGases.Common.SingleGasesData**

---

```
name="PbCL",
MM=0.242653,
Hf=36344.56610880558,
H0=40333.80176630827,
Tlimit=1000,
alow={-2902.700749,-82.93459710000001,4.71482448,-0.0001931473228,
      9.10557902e-008,5.41938522999999e-011,-3.2849697e-014},
blow={125.4460217,4.323194455},
ahigh={-385428.553,692.21975,4.50997124,-0.0006853421299999999,
      5.0873955e-007,-9.77275380999999e-011,5.807471579999999e-015},
bhigh={-5269.68472,6.646491015},
R=34.26486381787986);

constant IdealGases.Common.DataRecord PbCL2 (
  name="PbCL2",
  MM=0.278106,
  Hf=-631225.0400926265,
  H0=50349.61129928876,
  Tlimit=1000,
  alow={2830.580143,-313.4256659,8.21282729,-0.002585096173,3.109456077e-006,
        -1.972978208e-009,5.12354248e-013},
  blow={-21675.69042,-9.219412244000001},
  ahigh={-31491.92744,-5.56547758,7.00459954,-1.986025455e-006,
        4.65171959e-010,-5.57913701e-014,2.67814357e-018},
  bhigh={-23274.39309,-2.140739285},
  R=29.896773172819);

constant IdealGases.Common.DataRecord PbCL3 (
  name="PbCL3",
  MM=0.313559,
  Hf=-566571.2258299076,
  H0=58220.91855121364,
  Tlimit=1000,
  alow={862.3588970000001,-525.315275,12.03336337,-0.00433518498,
        5.21569835e-006,-3.31002238e-009,8.59697060999999e-013},
  blow={-21805.76379,-26.94206962},
  ahigh={-56644.1279,-9.32951478999999,10.00771099,-3.32972757e-006,
        7.79933081e-010,-9.3546061e-014,4.4905897e-018},
  bhigh={-24485.08643,-15.0749087},
  R=26.51645144932851);

constant IdealGases.Common.DataRecord PbCL4 (
  name="PbCL4",
  MM=0.349012,
  Hf=-938163.6218811962,
  H0=67188.05657112076,
  Tlimit=1000,
  alow={17394.05796,-890.213932,16.37565058,-0.00708899226,8.43256734e-006,-
5.30530939e-009,
        1.368671294e-012},
  blow={-38882.1562,-48.50133784},
  ahigh={-82666.7956,-16.39486405,13.01336711,-5.7161414e-006,1.32936839e-009,
        -1.585922492e-013,7.58176361e-018},
  bhigh={-43439.84220000001,-28.74559263},
  R=23.82288288081785);

constant IdealGases.Common.DataRecord PbF (
```

```

name="PbF",
MM=0.2261984032,
Hf=-437084.3807972558,
H0=40973.37058478404,
Tlimit=1000,
alow={26848.64255,-475.172564,6.01964437,-0.002718531008,2.942882954e-006,-
1.687543009e-009,
        4.08612208e-013},
blow={-10790.38653,-4.98304884},
ahigh={-473420.275,1027.785283,3.88865873,-0.0001649813448,2.93116326e-007,
      -5.76645965e-011,3.24748619e-015},
bhigh={-20261.65076,9.319483890000001},
R=36.75743012495325);

constant IdealGases.Common.DataRecord PbF2 (
  name="PbF2",
  MM=0.2451968064,
  Hf=-1808454.997886954,
  H0=51275.5862712574,
  Tlimit=1000,
  alow={59205.5228,-1052.529292,10.28969284,-0.00587634627,6.10233313e-006,-
3.42479264e-009,
        8.024025419999999e-013},
  blow={-49990.50000000001,-25.25115478},
  ahigh={-85058.99769999999,-30.74244334,7.02334604,-9.455183590000001e-006,
        2.108489027e-009,-2.434212002e-013,1.133872429e-017},
  bhigh={-55522.3677,-5.49607322},
  R=33.90938129282258);

constant IdealGases.Common.DataRecord PbF3 (
  name="PbF3",
  MM=0.2641952096,
  Hf=-1853071.498689279,
  H0=58801.39546633173,
  Tlimit=1000,
  alow={85530.9087,-1731.036694,15.5659615,-0.01022225919,1.089984998e-005,-
6.26999881999999e-009,
        1.502616508e-012},
  blow={-53003.3567,-53.3573348},
  ahigh={-145970.4258,-48.2039643,10.03685845,-1.501095764e-005,
        3.36247499e-009,-3.89602071e-013,1.820152103e-017},
  bhigh={-62068.6959,-20.06978661},
  R=31.47094155336267);

constant IdealGases.Common.DataRecord PbF4 (
  name="PbF4",
  MM=0.2831936128,
  Hf=-2824658.243845816,
  H0=69303.21205323456,
  Tlimit=1000,
  alow={130996.3563,-2307.299499,19.4520144,-0.01024522729,9.39587496e-006,-
4.62689415e-009,
        9.45088143000001e-013},
  blow={-88041.47459999999,-75.2464582000001},
  ahigh={-213261.6322,-85.7862195,13.06428551,-2.575563598e-005,
        5.69332459e-009,-6.526219550000001e-013,3.02234196e-017},
  bhigh={-100290.8979,-35.9024028},
  R=29.35967346788974);

```

```
constant IdealGases.Common.DataRecord PbI (
  name="PbI",
  MM=0.33410447,
  Hf=325958.8954317193,
  H0=30945.78171911319,
  Tlimit=1000,
  allow={-2717.33406,-15.42541996,4.5653279,-0.0001061215818,2.414841342e-007,
         -1.876070301e-010,6.46061426999999e-014},
  blow={11818.64978,7.66980267},
  ahighe={-3901070.25,11479.76718,-8.17610902,0.0065234547,-1.522611682e-006,
          1.626692202e-010,-6.55332888e-015},
  bhigh={-61544.3369,99.38404970000001},
  R=24.88584483769403);

constant IdealGases.Common.DataRecord PbI2 (
  name="PbI2",
  MM=0.46100894,
  Hf=-22239.54702483644,
  H0=33073.91392453257,
  Tlimit=1000,
  allow={-5904.98672,-48.3330703,7.19543803,-0.000429860009,5.29141207e-007,-
3.41640282e-010,
         8.99134192e-014},
  blow={-3107.780155,1.324773678},
  ahighe={-10883.39477,-0.816441837,7.0006973,-3.079876763e-007,
          7.331718140000001e-011,-8.89919388e-015,4.3107125e-019},
  bhigh={-3352.09638,2.458605463},
  R=18.03538126614204);

constant IdealGases.Common.DataRecord PbI3 (
  name="PbI3",
  MM=0.58791341,
  Hf=37004.44934569531,
  H0=35830.12675284954,
  Tlimit=1000,
  allow={-10149.96331,-59.380782,10.24073798,-0.000530469453,
         6.538623030000001e-007,-4.22589572e-010,1.113029818e-013},
  blow={-113.8482403,-8.97554049999999},
  ahighe={-16242.29573,-1.002945584,10.00085839,-3.79667974e-007,
          9.04675172e-011,-1.098840054e-014,5.325388290000001e-019},
  bhigh={-413.825893,-7.57941711},
  R=14.14234113149418);

constant IdealGases.Common.DataRecord PbI4 (
  name="PbI4",
  MM=0.714817879999999,
  Hf=-57674.07356962029,
  H0=38500.07361315585,
  Tlimit=1000,
  allow={-12170.96972,-72.2631744,13.29294204,-0.00064546609,7.95576185e-007,-
5.14163283e-010,
         1.354188031e-013},
  blow={-8528.148679999998,-20.10457498},
  ahighe={-19584.77744,-1.223385179,13.00104732,-4.63329569e-007,
          1.104213978e-010,-1.341397771e-014,6.50170611e-019},
  bhigh={-8893.193200000002,-18.40570345},
  R=11.63159488959622);
```

```

constant IdealGases.Common.DataRecord PbO (
  name="PbO",
  MM=0.2231994,
  Hf=305274.6019926577,
  H0=40152.92155803287,
  Tlimit=1000,
  allow={34240.25,-419.805011,4.72030641,0.001451061751,-3.20603546e-006,
    2.713956273e-009,-8.333275609999999e-013},
  blow={9253.186469999999,0.448289121},
  ahigh={242699.685,-110.7731354,3.18068449,0.001991815065,-1.079782025e-006,
    2.540990941e-010,-1.834652493e-014},
  bhigh={8339.391539999999,10.50920926},
  R=37.25131877594653);

constant IdealGases.Common.DataRecord PbO2 (
  name="PbO2",
  MM=0.2391988,
  Hf=569203.2485112802,
  H0=51216.53620335888,
  Tlimit=1000,
  allow={70710.58289999999,-1076.757462,9.401711730000001,-0.001003861584,-
    1.233815617e-006,
    1.858386979e-009,-6.83462121e-013},
  blow={19996.45981,-25.03976748},
  ahigh={-130779.667,-73.1436145,7.55408341,-2.143421561e-005,
    4.696463999999999e-009,-5.34497218e-013,2.460760535e-017},
  bhigh={14133.86101,-12.52684909},
  R=34.75967270738816);

constant IdealGases.Common.DataRecord PbS (
  name="PbS",
  MM=0.239265,
  Hf=534743.1216433662,
  H0=39412.83932041879,
  Tlimit=1000,
  allow={15298.40503,-343.974635,5.68063064,-0.002232931917,2.504608209e-006,-
    1.470427328e-009,
    3.57752719e-013},
  blow={15785.51202,-2.628900926},
  ahigh={2105441.649,-5520.41358,9.36924926,-0.001410262019,-1.642683362e-007,
    1.451447577e-010,-1.393564925e-014},
  bhigh={50093.3453,-32.17417657},
  R=34.75005537792824);

constant IdealGases.Common.DataRecord PbS2 (
  name="PbS2",
  MM=0.27133,
  Hf=899454.5571812922,
  H0=51675.83754100174,
  Tlimit=1000,
  allow={24463.7588,-665.485546,9.92020117999999,-0.00492419438,
    5.71690629e-006,-3.52948883e-009,8.971138350000001e-013},
  blow={30443.1013,-22.95765552},
  ahigh={-54167.003,-13.43600342,7.51070036,-4.497318629999999e-006,
    1.032444494e-009,-1.219580043e-013,5.78580465e-018},
  bhigh={27012.29533,-8.714618097000001},
  R=30.64339365348469);

```

---

**1170 Modelica.Media.IdealGases.Common.SingleGasesData**

---

```
constant IdealGases.Common.DataRecord Rb (
  name="Rb",
  MM=0.0854678,
  Hf=946555.3108890132,
  H0=72511.8465667772,
  Tlimit=1000,
  allow={13.52856616,-0.2042232679,2.501213823,-3.6506199e-006,5.88472267e-009,
         -4.84227472e-012,1.596211946e-015},
  blow={8985.56921,6.20700548},
  ahight={-1138274.064,3804.04194,-2.750899258,0.0038914607,-1.632296823e-006,
          3.51189314e-010,-2.521064422e-014},
  bhigh={-14664.54849,42.53442370000001},
  R=97.28192371864024);

constant IdealGases.Common.DataRecord Rbplus (
  name="Rbplus",
  MM=0.0854672514,
  Hf=5734700.952369694,
  H0=72512.3120081992,
  Tlimit=1000,
  allow={0,0,2.5,0,0,0,0},
  blow={58203.2736,5.52050692},
  ahight={0,0,2.5,0,0,0,0},
  bhigh={58203.2736,5.52050692},
  R=97.28254815504691);

constant IdealGases.Common.DataRecord Rbminus (
  name="Rbminus",
  MM=0.08546834859999999,
  Hf=325483.3918716899,
  H0=72511.3811313303,
  Tlimit=1000,
  allow={0,0,2.5,0,0,0,0},
  blow={2600.405796,5.52052617},
  ahight={0,0,2.5,0,0,0,0},
  bhigh={2600.405796,5.52052617},
  R=97.28129929024978);

constant IdealGases.Common.DataRecord RbBO2 (
  name="RbBO2",
  MM=0.1282776,
  Hf=-5293030.91108658,
  H0=111731.8846002731,
  Tlimit=1000,
  allow={41986.5775,-679.944833,8.18515101,0.002772536733,-6.17577047e-007,-
9.42613625e-010,
        4.72241424e-013},
  blow={-80203.3959,-12.39198887},
  ahight={91544.63399999999,-1668.328341,11.17268757,-0.000450553353,
          9.68133059e-008,-1.088037652e-011,4.96773366e-016},
  bhigh={-75079.0325,-31.31951149},
  R=64.81624227456703);

constant IdealGases.Common.DataRecord RbBr (
  name="RbBr",
  MM=0.1653718,
  Hf=-1158062.474980619,
```

```

H0=62399.53849447124,
Tlimit=1000,
alow={-5703.8597,20.92281597,4.37060737,0.000449460478,-4.835438840000001e-
007,
      3.5260686e-010,-9.88653832e-014},
blow={-24491.12515,6.4274317},
ahigh={1609092.474,-4466.77459,9.06409206,-0.001995333358,4.01663352e-007,-
4.1630914e-012,
      -3.092443341e-015},
bhigh={4427.46769,-27.57714089},
R=50.27744754547027);

constant IdealGases.Common.DataRecord RbCL(
  name="RbCL",
  MM=0.1209208,
  Hf=-1846852.476993206,
  H0=83171.14177213515,
  Tlimit=1000,
  alow={-5740.809579999999,-15.28835285,4.5152009,0.0001067270745,-
6.38594559e-009,
      -1.301381692e-011,1.641459742e-014},
  blow={-28142.43986,4.190115141},
  ahigh={-1143043.004,4003.86188,-1.096156782,0.00403358128,-1.454151464e-006,
      2.719263762e-010,-1.81046398e-014},
  bhigh={-52976.86040000001,43.11317822},
  R=68.75965094508142);

constant IdealGases.Common.DataRecord RbF(
  name="RbF",
  MM=0.1044662032,
  Hf=-3192533.803123803,
  H0=91781.98026057867,
  Tlimit=1000,
  alow={32873.6019,-608.090746,7.40662236,-0.00727786472,1.013930727e-005,-
7.141372500000001e-009,
      2.014332982e-012},
  blow={-38498.3413,-13.75750778},
  ahigh={-854481.1149999999,2633.756505,1.216216506,0.002136494067,-
6.72842608e-007,
      1.134905135e-010,-6.825624440000001e-015},
  bhigh={-58101.7911,25.88118896},
  R=79.59006592861414);

constant IdealGases.Common.DataRecord RbH(
  name="RbH",
  MM=0.08647574,
  Hf=1379856.431410705,
  H0=101995.1491597528,
  Tlimit=1000,
  alow={8908.270610000001,46.498704,1.833967623,0.00844427914,-1.167538647e-
005,
      8.01038942e-009,-2.162578321e-012},
  blow={13282.48009,12.79544976},
  ahigh={-2937703.53,9379.81726,-7.54256346,0.007493226800000001,-
2.159228887e-006,
      2.871300173e-010,-1.459981256e-014},
  bhigh={-45955.18829999999,83.65664320000001},
  R=96.14802949358978);

```

```
constant IdealGases.Common.DataRecord RbI (
  name="RbI",
  MM=0.21237227,
  Hf=-652066.6469308824,
  H0=49249.01918692116,
  Tlimit=1000,
  allow={-769.30528,-29.68608844,4.64874584,-0.0002962415236,5.97813534e-007,-
4.351255250000001e-010,
  1.313617336e-013},
  blow={-17866.10717,5.81240983},
  ahigh={4297424.85,-12969.11161,19.58730402,-0.008458425589999999,
  2.463480035e-006,-3.19157573e-010,1.458455589e-014},
  bhigh={64329.4804,-100.9499336},
  R=39.15045970926431);

constant IdealGases.Common.DataRecord RbK (
  name="RbK",
  MM=0.1245661,
  Hf=963450.9710105718,
  H0=86766.2229129755,
  Tlimit=1000,
  allow={24946.95429,-455.306066,7.68618995,-0.01108112244,2.093398586e-005,-
1.824445366e-008,
  5.61764536e-012},
  blow={15161.3616,-10.63563675},
  ahigh={-4175391.07,5918.588510000001,9.20264942,-0.010609336,
  4.96482606e-006,-8.716698790000001e-010,5.22188796e-014},
  bhigh={-32727.6904,-13.72401997},
  R=66.74746981722957);

constant IdealGases.Common.DataRecord RbLi (
  name="RbLi",
  MM=0.0924088,
  Hf=1776684.947753893,
  H0=111202.8724537057,
  Tlimit=1000,
  allow={-2263.413044,-28.8849691,4.55582654,0.000362566753,-
9.59800220000001e-007,
  1.631010244e-009,-8.159825980000001e-013},
  blow={18534.5525,2.831743182},
  ahigh={10855151.94,-38051.0697,55.1655504,-0.03128845612,9.26165025e-006,-
1.264671408e-009,
  6.43166205e-014},
  bhigh={254396.4131,-352.075871},
  R=89.9748941659236);

constant IdealGases.Common.DataRecord RbNO2 (
  name="RbNO2",
  MM=0.1314733,
  Hf=-1427132.178168495,
  H0=119950.0050580612,
  Tlimit=1000,
  allow={-71994.2003,1265.802595,-2.415715742,0.0305464142,-3.68400405e-005,
  2.247716481e-008,-5.54752743e-012},
  blow={-30373.81667,48.14256904},
  ahigh={-164499.4453,-849.944344,10.63080575,-0.0002517893988,
  5.56142012e-008,-6.378805599999999e-012,2.957679896e-016},
  bhigh={-21285.90982,-25.68367325},
```

```

R=63.24076447461196);

constant IdealGases.Common.DataRecord RbNO3 (
  name="RbNO3",
  MM=0.1474727,
  Hf=-2135801.968771169,
  H0=110986.7589052075,
  Tlimit=1000,
  allow={-26646.69172, 880.763479, -3.159459457, 0.0427122902, -5.35102921e-005,
         3.36096975e-008, -8.503290509999999e-012},
  blow={-43535.95460000001, 49.43052899},
  ahight={-314741.3867, -1371.457846, 14.01319759, -0.000403027502,
          8.878332289999999e-008, -1.016221214e-011, 4.704254090000001e-016},
  bhigh={-35043.3994, -46.19379241},
  R=56.37973672415301);

constant IdealGases.Common.DataRecord RbNa (
  name="RbNa",
  MM=0.10845757,
  Hf=1212182.127997151,
  H0=98334.56530512347,
  Tlimit=1000,
  allow={29744.55567, -508.279059, 7.73403352, -0.0101300684, 1.745455325e-005,
         -1.39872547e-008,
         3.96810677e-012},
  blow={16823.79436, -12.4008529},
  ahight={4297114.27, -18310.13588, 34.543942, -0.02230134993, 7.33987453e-006,
          -1.064784996e-009,
          5.61759846e-014},
  bhigh={124261.0951, -199.366198},
  R=76.6610574070579);

constant IdealGases.Common.DataRecord RbO (
  name="RbO",
  MM=0.1014672,
  Hf=517303.3650283047,
  H0=101876.5078764369,
  Tlimit=1000,
  allow={190008.1316, -3428.22899, 24.80251273, -0.0505165458,
         6.476338760000001e-005, -4.198012420000001e-008, 1.093889043e-011},
  blow={20838.9845, -109.755915},
  ahight={617075.688, -2175.08962, 8.29041221, -0.002996487663, 1.233144275e-006,
          -2.115810631e-010,
          1.24260743e-014},
  bhigh={18281.64827, -21.17386042},
  R=81.94246022359934);

constant IdealGases.Common.DataRecord RbOH (
  name="RbOH",
  MM=0.10247514,
  Hf=-2322514.514251945,
  H0=114771.2703783572,
  Tlimit=1000,
  allow={12138.32721, -544.141159, 8.459814639999999, -0.00356152656,
         2.99287239e-006, -7.2802674e-010, -4.55384718e-014},
  blow={-27872.62486, -19.13421991},
  ahight={895858.313, -2332.339, 7.97119248, 0.0001044439355, -6.32825775e-008,
          -6.32825775e-008});

```

---

**1174 Modelica.Media.IdealGases.Common.SingleGasesData**

---

```
 1.029358824e-011,-5.74327779e-016},  
bhigh={-15155.68947,-19.50051629},  
R=81.13647856445964);  
  
constant IdealGases.Common.DataRecord Rb2Br2 (  
  name="Rb2Br2",  
  MM=0.3307436,  
  Hf=-1668366.169443641,  
  H0=65599.35248936033,  
  Tlimit=1000,  
  alow={-8005.5254,-21.11968088,10.08629404,-0.000191193609,2.36605949e-007,-  
1.53372238e-010,  
  4.04872506e-014},  
  blow={-69273.136,-9.622434630000001},  
  ahight={-10145.35336,-0.360155032,10.00031058,-1.380745805e-007,  
  3.30206795e-011,-4.02158182e-015,1.953006729e-019},  
  bhigh={-69379.6088,-9.12256236},  
  R=25.13872377273513);  
  
constant IdealGases.Common.DataRecord Rb2CL2 (  
  name="Rb2CL2",  
  MM=0.2418416,  
  Hf=-2556936.176406375,  
  H0=85947.05790897844,  
  Tlimit=1000,  
  alow={-11250.39522,-67.56953990000001,10.27381915,-0.000603184899,  
  7.43332478e-007,-4.803372160000001e-010,1.264973638e-013},  
  blow={-77067.575,-13.63652575},  
  ahight={-18188.75133,-1.13790358,10.00097318,-4.3021564e-007,  
  1.024725787e-010,-1.244291058e-014,6.02891239e-019},  
  bhigh={-77408.9767,-12.04843385},  
  R=34.37982547254071);  
  
constant IdealGases.Common.DataRecord Rb2F2 (  
  name="Rb2F2",  
  MM=0.2089324064,  
  Hf=-4091817.840662175,  
  H0=90410.85739392509,  
  Tlimit=1000,  
  alow={-8178.3471,-333.473354,11.31540799,-0.002842914926,3.45495958e-006,-  
2.209373419e-009,  
  5.77207327e-013},  
  blow={-104223.0504,-23.68991587},  
  ahight={-43765.7439,-5.76145204,10.00482691,-2.104159548e-006,  
  4.96241627e-010,-5.98217497e-014,2.882751742e-018},  
  bhigh={-105917.6154,-16.03254014},  
  R=39.79503296430707);  
  
constant IdealGases.Common.DataRecord Rb2I2 (  
  name="Rb2I2",  
  MM=0.42474454,  
  Hf=-1019333.837699244,  
  H0=52307.95432944236,  
  Tlimit=1000,  
  alow={-5839.66355,-9.5275379,10.03904495,-8.668754e-005,1.074384368e-007,-  
6.97212223e-011,  
  1.842065721e-014},
```

```

blow={-55027.8096,-7.17411332},
ahigh={-6799.83464,-0.1642527699,10.000142,-6.32142072999999e-008,
      1.51281117e-011,-1.843027189e-015,8.95124417999999e-020},
bhigh={-55075.7962,-6.94804539},
R=19.57522985463216);

constant IdealGases.Common.DataRecord Rb2O (
  name="Rb2O",
  MM=0.186935,
  Hf=-582712.8199641587,
  H0=76046.64188086768,
  Tlimit=1000,
  alow={19712.76138,-462.750886,8.58829313,-0.003085559728,3.45198014e-006,-
2.068570368e-009,
      5.13248049999999e-013},
  blow={-12848.54711,-12.64891828},
  ahigh={-38463.6009,-10.82831019,7.00842189,-3.47622812e-006,7.86978506e-010,
      -9.195683850000001e-014,4.325201520000001e-018},
  bhigh={-15253.73768,-3.23038878},
  R=44.47787733704229);

constant IdealGases.Common.DataRecord Rb2O2 (
  name="Rb2O2",
  MM=0.2029344,
  Hf=-1063636.662882192,
  H0=83017.6254001293,
  Tlimit=1000,
  alow={32605.5323,-757.816472,10.70815124,0.001148004329,-3.29124367e-006,
      2.853106637e-009,-8.72159068e-013},
  blow={-24753.17258,-24.3427646},
  ahigh={-131856.7396,-96.47737790000001,10.07222726,-2.894921706e-005,
      6.4056391e-009,-7.35141939e-013,3.40856171e-017},
  bhigh={-28821.79733,-19.10948823},
  R=40.97123011179967);

constant IdealGases.Common.DataRecord Rb2O2H2 (
  name="Rb2O2H2",
  MM=0.20495028,
  Hf=-3117829.358418052,
  H0=114210.036697681,
  Tlimit=1000,
  alow={-4577.07653,-841.7631,17.04083828,-0.00535788254,3.95177877e-006,-
2.051775094e-010,
      -4.08541691e-013},
  blow={-76949.8217,-54.3541278},
  ahigh={1792711.652,-4659.406599999999,16.93822422,0.0002106246492,-
1.26961747e-007,
      2.063382927e-011,-1.150861507e-015},
  bhigh={-50241.1763,-60.27054920000001},
  R=40.56823928222982);

constant IdealGases.Common.DataRecord Rb2SO4 (
  name="Rb2SO4",
  MM=0.2669982,
  Hf=-4107114.025487812,
  H0=87348.70871788649,
  Tlimit=1000,

```

## 1176 Modelica.Media.IdealGases.Common.SingleGasesData

---

```
alow={60948.54930000001,-688.97059,7.03424218,0.0393394547,-5.49892051e-005,
      3.71634108e-008,-9.919863380000001e-012},
blow={-131187.7657,-4.450168488},
ahigh={-530289.831,-951.543545,19.71013517,-0.0002842916301,
       6.28914293999999e-008,-7.2193243e-012,3.34871819e-016},
bhigh={-133871.3151,-70.56246881999999},
R=31.1405545056109);

constant IdealGases.Common.DataRecord Rn (
  name="Rn",
  MM=0.2220176,
  Hf=0,
  H0=27914.12933028733,
  Tlimit=1000,
  alow={3.38943209e-006,-1.311675533e-007,2.50000001,-2.978593139e-012,
        4.33705073e-015,-3.18204022e-018,9.24778703e-022},
  blow={-745.374999,6.95244198},
  ahigh={27301.90029,-82.84672620000001,2.598178483,-5.81372985e-005,
         1.819136527e-008,-2.866656182e-012,1.789322176e-016},
  bhigh={-220.280934,6.25500571},
  R=37.44960759867686);

constant IdealGases.Common.DataRecord Rnplus (
  name="Rnplus",
  MM=0.2220170514,
  Hf=4699054.678103881,
  H0=27914.19830558113,
  Tlimit=1000,
  alow={-0.09697148970000001,0.001106742047,2.499994937,1.189388239e-008,-
1.515895754e-011,
        9.95893481e-015,-2.640751817e-018},
  blow={124730.476,8.33876169999999},
  ahigh={-19982.85319,59.3067566,2.432476003,3.71602592e-005,-1.012057848e-
008,
        1.192256661e-012,-3.18452198e-017},
  bhigh={124352.8478,8.82199778999999},
  R=37.44970013595992);

constant IdealGases.Common.DataRecord S (
  name="S",
  MM=0.032065,
  Hf=8644004.366131296,
  H0=207622.7974426945,
  Tlimit=1000,
  alow={-317.484182,-192.4704923,4.68682593,-0.0058413656,7.53853352e-006,-
4.86358604e-009,
        1.256976992e-012},
  blow={33235.9218,-5.718523969},
  ahigh={-485424.479,1438.830408,1.258504116,0.000379799043,1.630685864e-009,
        -9.54709584999999e-012,8.041466646e-016},
  bhigh={23349.9527,15.59554855},
  R=259.300545766412);

constant IdealGases.Common.DataRecord Splus (
  name="Splus",
  MM=0.03206445140000001,
  Hf=39997454.25240614,
```

```

H0=193280.337863507,
Tlimit=1000,
allow={0,0,2.5,0,0,0,0},
blow={153502.6117,5.43622334},
ahigh={1346218.684,-4056.87151,7.15343655,-0.002523562352,6.42953961e-007,-
6.43167216e-011,
2.141387919e-015},
bhigh={179282.3835,-27.86935079},
R=259.3049822146652);

constant IdealGases.Common.DataRecord Sminus(
  name="Sminus",
  MM=0.0320655486,
  Hf=2194520.539093474,
  H0=201615.2937423937,
  Tlimit=1000,
  allow={-2596.051473,-142.2398653,4.00782567,-0.00360885591,4.23623e-006,-
2.520987604e-009,
6.07947976e-013},
  blow={8197.79307,-2.582377345},
  ahigh={2730.311692,141.4072078,2.403340775,3.69357753e-005,-7.94408044e-009,
8.95220838e-013,-4.09966282e-017},
  bhigh={6931.1957,6.574986902},
  R=259.2961094699624);

constant IdealGases.Common.DataRecord SCL(
  name="SCL",
  MM=0.067518,
  Hf=2317382.031458278,
  H0=145430.122337747,
  Tlimit=1000,
  allow={16130.51454,-560.624955,8.06493113,-0.0091863264,1.224205395e-005,-
8.20956201999999e-009,
2.205087197e-012},
  blow={19977.3928,-16.9335441},
  ahigh={-94051.23449999999,362.976085,4.06995936,0.0003034029742,-
8.24830139e-008,
1.252700734e-011,-6.41746365e-016},
  bhigh={15231.11069,6.014528409},
  R=123.1445244231168);

constant IdealGases.Common.DataRecord SCL2(
  name="SCL2",
  MM=0.102971,
  Hf=-170659.7003039691,
  H0=120862.4272853522,
  Tlimit=1000,
  allow={57923.1183,-1062.447681,10.37694738,-0.00613022902,
6.463485999999999e-006,-3.67903805e-009,8.73193165e-013},
  blow={1262.475821,-26.91769492},
  ahigh={-234103.9963,428.7569580000001,6.47141539,0.000317947674,-
9.74748308e-008,
1.39331857e-011,-6.39479517e-016},
  bhigh={-7210.4737,-2.774273151},
  R=80.7457633702693);

constant IdealGases.Common.DataRecord SCL2plus(

```

---

**1178 Modelica.Media.IdealGases.Common.SingleGasesData**

---

```
name="SCL2plus",
MM=0.1029704514,
Hf=8753805.919510614,
H0=120940.0059015377,
Tlimit=1000,
alow={49348.7687,-960.1641020000001,9.88479278,-0.004950257170000001,
      4.94628697e-006,-2.678720604e-009,6.07684403e-013},
blow={111281.2037,-23.42664942},
ahigh={-233940.3605,551.628326,6.12521216,0.000677105632,-2.674382769e-007,
      4.93593888e-011,-2.955831247e-015},
bhigh={102678.7381,0.1365611954},
R=80.74619356286517);

constant IdealGases.Common.DataRecord SD(
  name="SD",
  MM=0.034079102,
  Hf=4063806.962988637,
  H0=272731.4528416858,
  Tlimit=1000,
  alow={-32955.7499,164.7104687,4.97898455,-0.008031585710000001,
        1.653766397e-005,-1.345197981e-008,4.00043153e-012},
  blow={14358.62329,-1.997334017},
  ahigh={245673.249,-1180.034105,5.28469005,-0.0002384319882,
        6.009960290000001e-008,-7.13116364e-012,4.0358587e-016},
  bhigh={22655.45124,-8.350483655},
  R=243.9756775281227);

constant IdealGases.Common.DataRecord SF(
  name="SF",
  MM=0.0510634032,
  Hf=302478.0964853514,
  H0=185457.9484823683,
  Tlimit=1000,
  alow={12064.56774,-291.027077,5.48010157,-0.001975880235,2.47368599e-006,-
1.610904623e-009,
        4.31003854e-013},
  blow={1991.337574,-4.544138975},
  ahigh={763458.075,-2423.63843,7.35219871,-0.001598026079,4.86851502e-007,-
6.47800053e-011,
        2.98029731e-015},
  bhigh={15774.61591,-19.02584835},
  R=162.8264369187207);

constant IdealGases.Common.DataRecord SFplus(
  name="SFplus",
  MM=0.0510628546,
  Hf=19477372.06998999,
  H0=173596.2094058094,
  Tlimit=1000,
  alow={65783.0367,-698.669997,5.62129224,-0.000690927411,-1.28239278e-007,
        4.02132042e-010,-1.424033453e-013},
  blow={122175.1247,-6.682377715},
  ahigh={-937494.39,2377.21424,2.052929113,0.001081609837,-1.535468024e-007,
        8.2226919e-012,3.25974688e-017},
  bhigh={102684.9122,18.87705768},
  R=162.8281862643848);
```

```

constant IdealGases.Common.DataRecord SFminus (
  name="SFminus",
  MM=0.0510639518,
  Hf=-4530532.495920145,
  H0=173803.7634603909,
  Tlimit=1000,
  allow={50198.7928,-532.154931,4.90484901,0.0009915307419999999,-2.270841545e-
006,
         1.844749756e-009,-5.3570572e-013},
  blow={-26113.88646,-3.637468065},
  ahighe={206892.3125,-1324.372811,6.64621124,-0.001641781588,6.78388312e-007,
          -1.198331099e-010,7.12173954e-015},
  bhigh={-21509.94816,-14.70239183},
  R=162.8246876106444);

constant IdealGases.Common.DataRecord SF2 (
  name="SF2",
  MM=0.0700618064,
  Hf=-4184715.597056031,
  H0=157694.6637219448,
  Tlimit=1000,
  allow={68708.71130000001,-843.241825,6.2694964,0.00648414245,-1.145126533e-
005,
         8.770291560000001e-009,-2.542611601e-012},
  blow={-32299.807,-8.80002578},
  ahighe={-170261.2287,-149.7038204,7.11093414,-4.4077416e-005,9.68220831e-009,
          -1.104468959e-012,5.09530878e-017},
  bhigh={-37042.8949,-10.9525396},
  R=118.6733889293497);

constant IdealGases.Common.DataRecord SF2plus (
  name="SF2plus",
  MM=0.07006125780000001,
  Hf=10077122.83749493,
  H0=160347.9633789846,
  Tlimit=1000,
  allow={131164.8923,-1552.127914,9.803424720000001,-0.002488118587,
         7.501565130000001e-007,3.135384495e-010,-1.986281446e-013},
  blow={91377.5839,-27.92300566},
  ahighe={-384753.102,532.789302,6.2819089,0.000452279012,-1.426421096e-007,
          2.079708115e-011,-9.69170989e-016},
  bhigh={78842.1381,-4.21611053},
  R=118.6743181764573);

constant IdealGases.Common.DataRecord SF2minus (
  name="SF2minus",
  MM=0.07006235499999999,
  Hf=-5634908.504003327,
  H0=173058.3421011184,
  Tlimit=1000,
  allow={52845.7248,-1032.342905,10.06404631,-0.00520950325,5.16838137e-006,-
2.783420867e-009,
         6.28641008e-013},
  blow={-44233.03780000001,-26.9926697},
  ahighe={-93766.18949999999,-36.3899527,7.02794282,-1.142143365e-005,
          2.566295898e-009,-2.981125977e-013,1.395700722e-017},
  bhigh={-49670.5033,-8.483087898999999},
  R=118.6724596967944);

```

```
constant IdealGases.Common.DataRecord SF3(
  name="SF3",
  MM=0.0890602096,
  Hf=-5660231.401476514,
  H0=152124.7823337708,
  Tlimit=1000,
  allow={128400.8253,-2084.871223,14.23421102,-0.00401503219,
         9.152258039999999e-007,1.109809721e-009,-5.98860272e-013},
  blow={-52395.1734,-51.87194724},
  ahighelement={-241128.346,-141.8950375,10.10590548,-4.231526910000001e-005,
               9.336120599999999e-009,-1.068727665e-012,4.9443426e-017},
  bhigh={-63584.4586,-24.89561796},
  R=93.35787595092299);

constant IdealGases.Common.DataRecord SF3plus(
  name="SF3plus",
  MM=0.08905966100000001,
  Hf=4419323.760956152,
  H0=139335.2260795154,
  Tlimit=1000,
  allow={215763.3857,-2564.488498,13.23216044,-0.0001749580869,-3.71295871e-
006,
         3.58068842e-009,-1.084376153e-012},
  blow={58760.8842,-50.24130184},
  ahighelement={-377826.075,-162.1770252,9.96026597,0.0001237729568,-
6.442207889999999e-008,
               1.300856025e-011,-7.915106980000001e-016},
  bhigh={44154.2680999999,-26.7390077},
  R=93.35845102756454);

constant IdealGases.Common.DataRecord SF3minus(
  name="SF3minus",
  MM=0.0890607581999999,
  Hf=-8871743.548664289,
  H0=154084.6190550397,
  Tlimit=1000,
  allow={154650.0675,-2469.768526,16.57548496,-0.01020137434,9.36800846e-006,-
4.7256726e-009,
         1.009661431e-012},
  blow={-85001.579,-65.32166474},
  ahighelement={-225386.0964,-130.8890242,10.0997238,-4.05234553e-005,
               9.064565220000001e-009,-1.049320641e-012,4.89920126e-017},
  bhigh={-98002.272,-25.24368023},
  R=93.35730088136619);

constant IdealGases.Common.DataRecord SF4(
  name="SF4",
  MM=0.1080586128,
  Hf=-7033220.03037966,
  H0=142356.8339570615,
  Tlimit=1000,
  allow={163618.3265,-2531.316261,15.59106568,0.00332670431,-1.054935525e-005,
         9.39136305e-009,-2.914816694e-012},
  blow={-81155.5892,-61.31532958},
  ahighelement={-377712.516,-293.7595659,13.21902007,-8.74766945e-005,
               1.929905265e-008,-2.209423506e-012,1.022335559e-016},
  bhigh={-94831.7534,-42.43061278},
  R=76.944093437409);
```

```

constant IdealGases.Common.DataRecord SF4plus(
  name="SF4plus",
  MM=0.1080580642,
  Hf=3850818.215934689,
  H0=152747.0172836948,
  Tlimit=1000,
  allow={230958.7324,-3383.77767,21.01756367,-0.01106529555,
         9.038097280000001e-006,-4.05591264e-009,7.71257518e-013},
  blow={64253.7515,-89.38224908000001},
  ahighelement={-418917.275,40.0288964,12.83873264,0.0001340235036,-4.81813518e-008,
             7.574595959999999e-012,-3.69667078e-016},
  bhigh={44699.4423,-37.60378598},
  R=76.94448407488684);

constant IdealGases.Common.DataRecord SF4minus(
  name="SF4minus",
  MM=0.1080591614,
  Hf=-8212763.346505111,
  H0=171020.3999417675,
  Tlimit=1000,
  allow={100552.3355,-2267.351994,19.7928456,-0.01163761842,1.161792139e-005,-
        6.28889547e-009,
         1.426371272e-012},
  blow={-98955.9743,-79.16366078},
  ahighelement={320882.208,-1710.416599,14.94425485,-0.001057207329,2.74284692e-007,
             -2.905761211e-011,1.084482543e-015},
  bhigh={-100523.1225,-51.64256578},
  R=76.94370280389758);

constant IdealGases.Common.DataRecord SF5(
  name="SF5",
  MM=0.127057016,
  Hf=-7104395.455029419,
  H0=148055.4131697851,
  Tlimit=1000,
  allow={222417.685,-4043.99175,27.07196786,-0.01744176604,1.604198715e-005,-
        7.99724302e-009,
         1.667968061e-012},
  blow={-92200.7827,-123.2426279},
  ahighelement={-389242.637,-182.5826536,16.13765057,-5.54483423e-005,
               1.231433706e-008,-1.417173409e-012,6.5848894e-017},
  bhigh={-113567.1054,-55.77524201},
  R=65.43890500308933);

constant IdealGases.Common.DataRecord SF5plus(
  name="SF5plus",
  MM=0.1270564674,
  Hf=1358797.466456241,
  H0=128662.0377106439,
  Tlimit=1000,
  allow={377713.562,-5646.07512,30.4940427,-0.02176488368,1.941993792e-005,-
        9.55620114e-009,
         2.000077302e-012},
  blow={45922.0481,-148.9889627},
  ahighelement={-981434.232,720.881739,15.64386818,-0.0002452519972,2.241690996e-007,
             -3.99003589e-011,2.279087545e-015},
  bhigh={9090.95319,-55.31460631},
  R=65.43918755291948);

```

```
constant IdealGases.Common.DataRecord SF5minus(
  name="SF5minus",
  MM=0.1270575646,
  Hf=-9480917.494305568,
  H0=149931.356389307,
  Tlimit=1000,
  allow={209503.1134,-3930.08114,27.02127215,-0.01792973203,1.718122152e-005,-
9.00056019e-009,
  1.988018147e-012},
  blow={-129181.6581,-123.1672857},
  ahighe={-373321.92,-179.3729521,16.13699748,-5.57753997e-005,
  1.249477046e-008,-1.448104799e-012,6.7674623e-017},
  bhigh={-149855.4401,-56.31439391},
  R=65.4386224556991);

constant IdealGases.Common.DataRecord SF6(
  name="SF6",
  MM=0.1460554192,
  Hf=-8348885.694752777,
  H0=115983.830608868,
  Tlimit=1000,
  allow={330952.674,-4737.68505,22.47738068,0.01046954309,-2.560641961e-005,
  2.153716967e-008,-6.51609896e-012},
  blow={-125536.0583,-109.1760145},
  ahighe={-730672.65,-636.705655,19.47442853,-0.0001894325671,4.17872283e-008,
  -4.78374495e-012,2.213516129e-016},
  bhigh={-151060.9837,-81.47574587},
  R=56.92682986733026);

constant IdealGases.Common.DataRecord SF6minus(
  name="SF6minus",
  MM=0.1460559678,
  Hf=-9187409.821127487,
  H0=119753.504519245,
  Tlimit=1000,
  allow={498580.921,-6934.44574,34.4131833,-0.0198403236,1.497871222e-005,-
6.133234650000001e-009,
  1.045159581e-012},
  blow={-129706.9066,-174.7984495},
  ahighe={-681807.3509999999,-594.742996,19.45009842,-0.0001820212649,
  4.05707957e-008,-4.683820930000001e-012,2.182263999e-016},
  bhigh={-165890.2452,-79.65845887},
  R=56.92661604478445);

constant IdealGases.Common.DataRecord SH(
  name="SH",
  MM=0.03307294,
  Hf=4297631.507812732,
  H0=275092.2355254779,
  Tlimit=1000,
  allow={6389.43468,-374.796092,7.54814577,-0.01288875477,1.907786343e-005,-
1.265033728e-008,
  3.23515869e-012},
  blow={17429.02395,-17.60761843},
  ahighe={1682631.601,-5177.15221,9.198168519999999,-0.002323550224,
  6.543914779999999e-007,-8.468470419999999e-011,3.86474155e-015},
  bhigh={48992.14490000001,-37.70400275},
  R=251.3980311396568);
```

```

constant IdealGases.Common.DataRecord SHminus(
  name="SHminus",
  MM=0.0330734886,
  Hf=-2617628.186946085,
  H0=261420.9859917832,
  Tlimit=1000,
  allow={38780.7076,-574.25906,6.89854446,-0.01001352412,1.486460572e-005,-
9.7503689e-009,
  2.446909813e-012},
  blow={-8735.38898,-16.15966489},
  ahigh={1198715.402,-3894.84682,7.66042233,-0.00135523759,3.34237024e-007,-
3.35072231e-011,
  9.05508478e-016},
  bhigh={13174.77387,-28.18370616},
  R=251.3938611241634);

constant IdealGases.Common.DataRecord SN(
  name="SN",
  MM=0.0460717,
  Hf=5803743.143838843,
  H0=203880.3213252387,
  Tlimit=1000,
  allow={-68354.1235,1147.567483,-2.877802574,0.0172486432,-2.058999904e-005,
  1.26136964e-008,-3.139030141e-012},
  blow={25641.43612,42.24006964},
  ahigh={-483728.446,1058.07559,3.086198804,0.000911136078,-2.764061722e-007,
  4.157370109999999e-011,-2.128351755e-015},
  bhigh={23793.45477,10.33222139},
  R=180.4680964670286);

constant IdealGases.Common.DataRecord SO(
  name="SO",
  MM=0.0480644,
  Hf=99040.16278160134,
  H0=183048.2852173334,
  Tlimit=1000,
  allow={-33427.57,640.38625,-1.006641228,0.01381512705,-1.704486364e-005,
  1.06129493e-008,-2.645796205e-012},
  blow={-3371.29219,30.93861963},
  ahigh={-1443410.557,4113.87436,-0.538369578,0.002794153269,-6.63335226e-007,
  7.838221189999999e-011,-3.56050907e-015},
  bhigh={-27088.38059,36.15358329},
  R=172.9860770133404);

constant IdealGases.Common.DataRecord S0minus(
  name="S0minus",
  MM=0.0480649486,
  Hf=-2204685.307829498,
  H0=196965.0291064703,
  Tlimit=1000,
  allow={8420.196969999999,-69.1658668,3.83766661,0.002166482062,-2.811222562e-
006,
  1.793426453e-009,-4.52179595e-013},
  blow={-13541.62531,4.316141603},
  ahigh={176715.6147,-663.398736,5.17727981,-0.0002853461125,7.21442153e-008,
  -3.67646949e-012,-2.910092894e-016},
  bhigh={-9984.43801,-3.951456757},
  R=172.9841025982081);

```

```
constant IdealGases.Common.DataRecord SOF2 (
  name="SOF2",
  MM=0.08606120640000001,
  Hf=-6796933.687882859,
  H0=146696.1192865639,
  Tlimit=1000,
  allow={61145.7965,-908.1119669999999,6.88362326,0.01192459695,-1.67100641e-005,
         1.110723646e-008,-2.910788226e-012},
  blow={-67429.3584,-11.25458745},
  ahighe={-251055.5779,-607.4439609999999,10.45046709,-0.0001796384979,
          3.96416973e-008,-4.54305524e-012,2.104969923e-016},
  bhigh={-70720.71710000001,-29.04334504},
  R=96.6114859761017);

constant IdealGases.Common.DataRecord SO2 (
  name="SO2",
  MM=0.0640638,
  Hf=-4633037.690552231,
  H0=164650.3485587805,
  Tlimit=1000,
  allow={-53108.4214,909.031167,-2.356891244,0.02204449885,-2.510781471e-005,
         1.446300484e-008,-3.36907094e-012},
  blow={-41137.52080000001,40.45512519},
  ahighe={-112764.0116,-825.226138,7.61617863,-0.000199932761,5.65563143e-008,
          -5.45431661e-012,2.918294102e-016},
  bhigh={-33513.0869,-16.55776085},
  R=129.7842463294403);

constant IdealGases.Common.DataRecord SO2minus (
  name="SO2minus",
  MM=0.0640643486,
  Hf=-6378057.030614996,
  H0=167791.1542832732,
  Tlimit=1000,
  allow={94609.01659999999,-856.269105,5.36007422,0.00760957674,-1.092005659e-005,
         7.20254773e-009,-1.85013822e-012},
  blow={-45800.9481,-3.929956604},
  ahighe={-179340.0005,-366.316876,7.27444105,-0.0001101944663,
          2.443328132e-008,-2.809735019e-012,1.305160431e-016},
  bhigh={-49730.7261,-12.61421156},
  R=129.7831349525343);

constant IdealGases.Common.DataRecord SO2CL2 (
  name="SO2CL2",
  MM=0.1349698,
  Hf=-2628759.915181026,
  H0=118757.7072797026,
  Tlimit=1000,
  allow={6821.59239,-584.9569200000001,8.59675691,0.01197228675,-1.3337607e-005,
         7.12612899999999e-009,-1.496150016e-012},
  blow={-42307.8381,-16.52458363},
  ahighe={-237663.9109,-964.7740410000001,13.71469904,-0.0002850327057,
          6.29350962e-008,-7.21784426e-012,3.34684022e-016},
  bhigh={-41900.7299,-44.81942574999999},
  R=61.60246218042851);
```

```

constant IdealGases.Common.DataRecord SO2FCL(
  name="SO2FCL",
  MM=0.1185152032,
  Hf=-4695363.843412791,
  H0=124042.9970422563,
  Tlimit=1000,
  allow={36845.613,-803.835894999999,7.52618688,0.01620141782,-1.97059527e-
005,
         1.16625033e-008,-2.759241465e-012},
  blow={-65035.9819,-12.99745646},
  ahighe={-291540.2438,-1107.134423,13.81845158,-0.000325851446,7.184741e-008,
          -8.230625200000001e-012,3.81292562e-016},
  bhigh={-65524.7269,-47.09166446},
  R=70.15531995476509);

constant IdealGases.Common.DataRecord SO2F2 (
  name="SO2F2",
  MM=0.1020606064,
  Hf=-7446555.794714541,
  H0=132180.7842991613,
  Tlimit=1000,
  allow={55331.5353,-807.463968,5.28717216,0.02294547775,-2.891025299e-005,
         1.785631907e-008,-4.4189195e-012},
  blow={-88994.28820000001,-4.124256607},
  ahighe={-340390.445,-1309.288764,13.96601541,-0.000384008815,
          8.456688780000001e-008,-9.678246870000001e-012,4.48001428e-016},
  bhigh={-89019.9414,-51.0927005},
  R=81.46602585735764);

constant IdealGases.Common.DataRecord SO3 (
  name="SO3",
  MM=0.0800632,
  Hf=-4944843.573576874,
  H0=145990.9046852986,
  Tlimit=1000,
  allow={-39528.5529,620.857257,-1.437731716,0.02764126467,-3.144958662e-005,
         1.792798e-008,-4.12638666e-012},
  blow={-51841.0617,33.91331216},
  ahighe={-216692.3781,-1301.022399,10.96287985,-0.000383710002,
          8.466889039999999e-008,-9.70539929e-012,4.49839754e-016},
  bhigh={-43982.83990000001,-36.55217314},
  R=103.8488594010732);

constant IdealGases.Common.DataRecord S2 (
  name="S2",
  MM=0.0641299999999999,
  Hf=2005301.730859192,
  H0=142399.9688133479,
  Tlimit=1000,
  allow={35280.9178,-422.215658,4.67743349,0.001724046361,-3.86220821e-006,
         3.33615634e-009,-9.93066154e-013},
  blow={16547.67715,-0.7957279032},
  ahighe={-15881.28788,631.548088,2.449628069,0.001986240565,-6.50792724e-007,
          1.002813651e-010,-5.59699005e-015},
  bhigh={10855.08427,14.58544515},
  R=129.650272883206);

```

```
constant IdealGases.Common.DataRecord S2minus (
  name="S2minus",
  MM=0.0641305486,
  Hf=-579012.4333974589,
  H0=149649.6632183807,
  Tlimit=1000,
  allow={10255.58251,-616.612824999999,8.182052840000001,-
0.00831120358000001,
         9.72027416e-006,-5.77547678e-009,1.393690695e-012},
  blow={-3063.559874,-19.06068041},
  ahighe={483020.403,-1319.171302,6.03177848,-0.0007965560690000001,
         2.28224169e-007,-2.504364698e-011,7.28055078e-016},
  bhigh={2660.25849,-9.01048031899999},
  R=129.6491637996061);

constant IdealGases.Common.DataRecord S2CL2 (
  name="S2CL2",
  MM=0.135036,
  Hf=-123937.3204182589,
  H0=122343.382505406,
  Tlimit=1000,
  allow={79749.74740000001,-1636.770024,15.83744686,-0.01186906072,
         1.413008027e-005,-8.48490393e-009,1.981743887e-012},
  blow={3276.87242,-52.93745205},
  ahighe={632881.6,-3442.7058,15.1963735,-0.002960877176,6.88828437e-007,-
7.99069252999999e-011,
         3.69246606e-015},
  bhigh={15266.72656,-54.53315445000001},
  R=61.5722622115584);

constant IdealGases.Common.DataRecord S2F2 (
  name="S2F2",
  MM=0.1021268064,
  Hf=-3930535.127357121,
  H0=134322.1479605574,
  Tlimit=1000,
  allow={125290.4041,-1941.243874,13.25382183,-0.001341525124,-2.772465632e-
006,
         3.65816929e-009,-1.299531209e-012},
  blow={-40672.0825,-45.60828617},
  ahighe={-259175.7719,-105.9653069,10.06588551,-2.274618148e-005,
         5.41234314e-009,-9.91138714e-013,9.621861440000001e-017},
  bhigh={-51476.75180000001,-23.7566614},
  R=81.41321845936034);

constant IdealGases.Common.DataRecord S2O (
  name="S2O",
  MM=0.0801294,
  Hf=-699312.5494512626,
  H0=138882.320346839,
  Tlimit=1000,
  allow={10927.0331,-95.2309986999999,3.14452543,0.011768542,-1.58026684e-005,
         1.03764504e-008,-2.70862226e-012},
  blow={-7500.4719,11.04169896},
  ahighe={-144213.979,-327.643013,7.24428611,-9.7765383e-005,2.1627127e-008,-
2.48278222e-012,
         1.15177875e-016},
  bhigh={-7438.55393,-10.85180744},
```

```

R=103.7630632452009);

constant IdealGases.Common.DataRecord S3 (
  name="S3",
  MM=0.0961949999999999,
  Hf=1504634.627579396,
  H0=124479.733873902,
  Tlimit=1000,
  alow={72453.9574,-1162.146759,9.95541368,-0.00415802622,3.24177839e-006,-
1.264648239e-009,
  1.777450535e-013},
  blow={21462.75475,-25.87525865},
  ahight={-111780.5401,-51.97908390000001,7.03876084,-1.546804022e-005,
  3.40834041e-009,-3.89686236e-013,1.800860389e-017},
  bhigh={15254.06485,-7.610045099},
  R=86.43351525547067);

constant IdealGases.Common.DataRecord S4 (
  name="S4",
  MM=0.12826,
  Hf=1057479.596132855,
  H0=111338.944331826,
  Tlimit=1000,
  alow={119866.4135,-2040.786521,15.10235054,-0.0070411043,5.350405e-006,-
2.002348038e-009,
  2.569940995e-013},
  blow={24109.09409,-55.03153238},
  ahight={-206833.3537,-96.96151430000001,10.0724321,-2.895047053e-005,
  6.38780619e-009,-7.31176743e-013,3.38226529e-017},
  bhigh={13211.0953,-23.44872237},
  R=64.82513644160301);

constant IdealGases.Common.DataRecord S5 (
  name="S5",
  MM=0.160325,
  Hf=829523.4929050366,
  H0=118842.0645563699,
  Tlimit=1000,
  alow={136137.0439,-2955.476536,26.33358816,-0.0412580653,7.05681403e-005,-
5.611095330000001e-008,
  1.6594572e-011},
  blow={26753.07046,-106.9711067},
  ahight={-4038495.16,8601.80388,7.4437809,0.001533393812,-2.532718676e-007,
  2.145210795e-011,-7.213102680000001e-016},
  bhigh={-46869.2542,11.04229196},
  R=51.8601091532824);

constant IdealGases.Common.DataRecord S6 (
  name="S6",
  MM=0.19239,
  Hf=526613.5454025677,
  H0=118442.6373512137,
  Tlimit=1000,
  alow={97803.07210000001,-2568.47013,24.67025557,-0.01619983175,
  1.712334526e-005,-9.74629674e-009,2.486964867e-012},
  blow={20378.88389,-101.4410309},
  ahight={3686845.06,-7695.0179,18.60721995,0.002290382548,-1.219834021e-006,
  -1.219834021e-006});

```

---

**1188 Modelica.Media.IdealGases.Common.SingleGasesData**

---

```
    2.060780798e-010,-1.190354318e-014},
bhigh={60324.8648,-74.90245718},
R=43.21675762773533);

constant IdealGases.Common.DataRecord S7(
  name="S7",
  MM=0.224455,
  Hf=498498.0018266468,
  H0=117058.5106145998,
  Tlimit=1000,
  allow={123365.5613,-3200.54364,30.15678887,-0.02197472483,2.487892075e-005,-
1.506049557e-008,
  3.76892599e-012},
  blow={23900.05896,-127.5978171},
  ahight={-272639.6041,-73.7321522,19.05771987,-2.394575885e-005,
  5.4426713e-009,-6.379663479999999e-013,3.008212993e-017},
  bhigh={7308.458830000001,-61.59112428},
  R=37.04293510948742);

constant IdealGases.Common.DataRecord S8(
  name="S8",
  MM=0.25652,
  Hf=394811.7963511617,
  H0=123082.4964915017,
  Tlimit=1000,
  allow={314562.5719,-6116.51016,48.7532754,-0.0624179465,
  7.421831750000001e-005,-3.72644931e-008,5.79942988e-012},
  blow={35738.9084,-228.8701977},
  ahight={-8727921.130000001,12216.27968,22.09959617,-0.00349406483,
  1.397604162e-006,-2.169815281e-010,1.212304364e-014},
  bhigh={-86581.1081999999,-64.35742508},
  R=32.4125682208015);

constant IdealGases.Common.DataRecord Sc(
  name="Sc",
  MM=0.04495591,
  Hf=8401570.761219159,
  H0=155758.3641394424,
  Tlimit=1000,
  allow={-3700.80594,169.2506026,1.842242597,0.001364835821,-1.580085847e-006,
  9.61311115e-010,-2.392381918e-013},
  blow={43852.1524,10.72781921},
  ahight={8810382.649999999,-27112.32975,34.7658866,-0.01861104581,
  5.290283900000001e-006,-6.58540806e-010,2.997850429e-014},
  bhigh={216246.0097,-222.5618519},
  R=184.9472516516738);

constant IdealGases.Common.DataRecord Scplus(
  name="Scplus",
  MM=0.0449553614,
  Hf=22625671.11739424,
  H0=159311.8546256421,
  Tlimit=1000,
  allow={-5884.93016,147.9044408,2.009576456,0.000738956697,-6.61796482e-007,
  6.84107376e-010,-2.164125532e-013},
  blow={120843.9139,10.2657062},
  ahight={1973658.531,-4954.3841,6.36062735,-0.000716878592,2.464991123e-008,
```

```

9.632373790000001e-012,-8.544709642000001e-016},
bhigh={154342.3764,-22.61925782},
R=184.9495086029939);

constant IdealGases.Common.DataRecord Scminus(
  name="Scminus",
  MM=0.0449564586,
  Hf=7842228.65810876,
  H0=137854.0079222344,
  Tlimit=1000,
  allow={0,0,2.5,0,0,0,0},
  blow={41657.4639,8.270420720000001},
  ahight={0,0,2.5,0,0,0,0},
  bhigh={41657.4639,8.270420720000001},
  R=184.9449947554365);

constant IdealGases.Common.DataRecord ScO(
  name="ScO",
  MM=0.06095531,
  Hf=-903371.5684490819,
  H0=144156.5304154798,
  Tlimit=1000,
  allow={-3136.784613,227.1981685,0.885075732,0.01052430774,-1.431954596e-005,
         9.652351069999999e-009,-2.586210633e-012},
  blow={-8550.798769999999,20.12706847},
  ahight={1224192.443,-3918.91572,8.679848809999999,-0.001995893306,
          4.06698985e-007,-1.504459913e-011,-8.4729989e-016},
  bhigh={16773.22594,-29.51656272},
  R=136.4027514584045);

constant IdealGases.Common.DataRecord ScOplus(
  name="ScOplus",
  MM=0.0609547614,
  Hf=9206991.432830053,
  H0=143993.1155238678,
  Tlimit=1000,
  allow={45885.6027,-336.711437,3.5272064,0.0039888503,-5.568985429999999e-006,
         3.65028119e-009,-9.356491500000001e-013},
  blow={68383.5079,4.3398127},
  ahight={-1814357.141,5853.42532,-3.65710121,0.00573436451,-2.092878161e-006,
          3.69564502e-010,-2.214038347e-014},
  bhigh={29553.64078,56.6088941},
  R=136.4039790991619);

constant IdealGases.Common.DataRecord ScO2(
  name="ScO2",
  MM=0.07695471000000001,
  Hf=-5375248.207679556,
  H0=139720.2718326143,
  Tlimit=1000,
  allow={47947.00960000001,-429.835859,3.4172065,0.01361338544,-2.047465023e-005,
         1.455906041e-008,-4.03808769e-012},
  blow={-48610.4953,7.67001333},
  ahight={-190390.2542,-231.558882,7.17210491,-6.85920964e-005,
          1.510998557e-008,-1.727951588e-012,7.9889805e-017},
  bhigh={-51129.7475000001,-10.90764597},

```

---

**1190 Modelica.Media.IdealGases.Common.SingleGasesData**

---

```
R=108.0437051871159);

constant IdealGases.Common.DataRecord Sc2O(
  name="Sc2O",
  MM=0.10591122,
  Hf=-217580.4225463553,
  H0=110562.827998771,
  Tlimit=1000,
  allow={25500.02615,-307.7608607,4.67121504,0.008696142249999999,-
1.283873116e-005,
  8.98896681000001e-009,-2.461994147e-012},
  blow={-2614.951944,2.92001718},
  ahigh={-135088.0396,-185.4540692,7.13829408,-5.52943214e-005,
  1.221589315e-008,-1.400490463e-012,6.48885811e-017},
  bhigh={-4240.235140000001,-9.37271095},
  R=78.50416603642182};

constant IdealGases.Common.DataRecord Sc2O2 (
  name="Sc2O2",
  MM=0.12191062,
  Hf=-4024022.02531658,
  H0=107732.8619934834,
  Tlimit=1000,
  allow={3468.44549,261.7988314,-0.3052099624,0.0307610407,-4.18347836e-005,
  2.794218425e-008,-7.41267205e-012},
  blow={-61440.1321,29.15248555},
  ahight={-297734.8677,-580.367792,10.43318377,-0.0001734165462,
  3.83608979e-008,-4.403078029999999e-012,2.042208922e-016},
  bhigh={-59648.6628,-28.55828369},
  R=68.20137572920227);

constant IdealGases.Common.DataRecord Si (
  name="Si",
  MM=0.0280855,
  Hf=16022502.71492407,
  H0=268831.1762297271,
  Tlimit=1000,
  allow={98.36140810000001,154.6544523,1.87643667,0.001320637995,-1.529720059e-
006,
  8.95056277e-010,-1.95287349e-013},
  blow={52635.1031,9.69828888},
  ahight={-616929.885,2240.683927,-0.444861932,0.001710056321,-4.10771416e-007,
  4.55888478e-011,-1.889515353e-015},
  bhigh={39535.5876,26.79668061},
  R=296.0414448736893);

constant IdealGases.Common.DataRecord Siplus (
  name="Siplus",
  MM=0.0280849514,
  Hf=44241060.89070889,
  H0=261454.7874916387,
  Tlimit=1000,
  allow={-43297.9188,679.5894490000001,0.2257046144,0.004118600490000001,-
4.23488159999999e-006,
  2.327995626e-009,-5.318388059e-013},
  blow={145203.9813,19.3465051},
  ahight={59193.9023,-48.5673095,2.556312024,-3.50339716e-005,1.190298787e-008,
```

```

-2.082923821e-012,1.471452049e-016},
bhigh={149143.1392,5.24426714},
R=296.0472276266784);

constant IdealGases.Common.DataRecord Siminus(
  name="Siminus",
  MM=0.0280860486,
  Hf=10995406.73727952,
  H0=220658.5941747605,
  Tlimit=1000,
  allow={-794.014667,5.56741842,2.499837183,-9.48139446e-005,3.17124693e-007,-
4.19132318e-010,
  2.03592438e-013},
  blow={36364.4338,5.27011984},
  ahigh={-6162070.100000001,18833.10402,-18.9930245,0.01111021657,-
2.535790208e-006,
  2.699962923e-010,-1.105062911e-014},
  bhigh={-83140.8931,159.5298253},
  R=296.0356623466072);

constant IdealGases.Common.DataRecord SiBr(
  name="SiBr",
  MM=0.1079895,
  Hf=1621985.665273013,
  H0=92926.82158913597,
  Tlimit=1000,
  allow={7370.283350000001,-505.106296,8.28059103999999,-0.0100538392,
  1.35705092e-005,-9.1154499e-009,2.447477685e-012},
  blow={21844.05475,-16.62032657},
  ahigh={1317799.94,-3669.79502,8.51168612,-0.001987602041,5.03541489e-007,-
4.49012019e-011,
  8.23567007e-016},
  bhigh={43328.5155,-24.67075472},
  R=76.99333731520194);

constant IdealGases.Common.DataRecord SiBr2(
  name="SiBr2",
  MM=0.1878935,
  Hf=-271430.3581550187,
  H0=70876.11333015778,
  Tlimit=1000,
  allow={23866.16151,-641.569347,9.34587911,-0.00479290414,5.58253847e-006,-
3.45531238e-009,
  8.80033894e-013},
  blow={-5014.75702,-17.40767596},
  ahigh={-51473.4394,-12.77719997,7.01020224,-4.29633186e-006,
  9.877651970000001e-010,-1.168133105e-013,5.54668138e-018},
  bhigh={-8319.54225,-3.61182661},
  R=44.25098260450734);

constant IdealGases.Common.DataRecord SiBr3(
  name="SiBr3",
  MM=0.2677975,
  Hf=-586263.8747561124,
  H0=65203.89473389407,
  Tlimit=1000,
  allow={39728.0686,-1066.475023,13.80836232,-0.00764111472,8.77586583e-006,-

```

---

**1192 Modelica.Media.IdealGases.Common.SingleGasesData**

---

```
5.37243085e-009,
  1.356453455e-012},
blow={-16517.95623,-37.8405537},
ahigh={-88880.24740000001,-22.67879074,10.01791167,-7.48115075e-006,
  1.70924269e-009,-2.011596627e-013,9.51553415e-018},
bhigh={-22030.26938,-15.37574366},
R=31.04760873421148);

constant IdealGases.Common.DataRecord SiBr4(
  name="SiBr4",
  MM=0.3477015,
  Hf=-1195853.33971812,
  H0=64174.93453436353,
  Tlimit=1000,
  allow={68210.8674,-1520.370776,18.1045336,-0.00973468842,1.072367466e-005,-
6.34385829e-009,
  1.5572282e-012},
blow={-46165.8521,-59.7612063},
ahigh={-127140.9281,-37.6808795,13.02910642,-1.194958251e-005,
  2.693843051e-009,-3.137193207e-013,1.471642663e-017},
bhigh={-54089.9883,-29.4038433},
R=23.9126722202809);

constant IdealGases.Common.DataRecord SiC(
  name="SiC",
  MM=0.04009620000000001,
  Hf=18329570.88202872,
  H0=229874.9258034427,
  Tlimit=1000,
  allow={-6223.330889999999,314.1790457,0.389313083,0.0118750754,-1.639277197e-
005,
  1.131808223e-008,-3.045324231e-012},
blow={86062.2773,23.10166717},
ahigh={-62688.0603000001,720.983692,2.162879732,0.002201299585,-
6.569466590000001e-007,
  9.177110259999999e-011,-4.96916674e-015},
bhigh={83212.2585,16.01675317},
R=207.3630917643068);

constant IdealGases.Common.DataRecord SiC2(
  name="SiC2",
  MM=0.05210690000000001,
  Hf=12116647.46895325,
  H0=224258.3803680511,
  Tlimit=1000,
  allow={-41196.2899,686.9747180000001,0.02361343691,0.01532939217,-
1.435588838e-005,
  6.37522874e-009,-1.059204957e-012},
blow={71308.9071,28.28850316},
ahigh={7026893.09,-24661.48437,39.1545303000001,-0.02002884068,
  6.30735369e-006,-8.84838351e-010,4.53054985e-014},
bhigh={226732.5155,-236.6622024},
R=159.5656621291998);

constant IdealGases.Common.DataRecord SiCL(
  name="SiCL",
  MM=0.0635385,
```

```

Hf=2240581.993594435,
H0=155561.100749939,
Tlimit=1000,
alow={12485.08319,-204.4170347,4.97707085,-0.0005160756260000001,
      2.931980374e-007,-3.66661523e-012,-3.39303713e-014},
blow={16865.28019,-0.226395724},
ahigh={376729.456,-1147.956509,5.73540843,-0.0005784890810000001,
      1.517304648e-007,-1.291391394e-011,1.55462842e-016},
bhigh={23062.78401,-6.098515228},
R=130.857228294656);

constant IdealGases.Common.DataRecord SiCL2(
  name="SiCL2",
  MM=0.0989915,
  Hf=-1647305.950510903,
  H0=126569.392321563,
  Tlimit=1000,
  alow={53055.5097,-1015.842786,10.32848185,-0.00621226752,6.7158254e-006,-
3.90880164e-009,
      9.46173215e-013},
  blow={-16502.18093,-26.49925756},
  ahigh={-80508.7157,-26.61025523,7.02041472,-8.335824719999999e-006,
      1.871092411e-009,-2.171552767e-013,1.015849895e-017},
  bhigh={-21812.94885,-6.638764238},
  R=83.99177707176879);

constant IdealGases.Common.DataRecord SiCL3(
  name="SiCL3",
  MM=0.1344445,
  Hf=-2501193.20611851,
  H0=116906.7310302764,
  Tlimit=1000,
  alow={86917.0845999999,-1677.556857,15.1151906,-0.008937031,
      9.09735352999999e-006,-5.01457996e-009,1.155962016e-012},
  blow={-34774.773,-50.87215506},
  ahigh={-147718.0114,-53.3520098,10.04043521,-1.63512106e-005,
      3.64194011e-009,-4.20060059e-013,1.955201587e-017},
  bhigh={-43605.7165,-20.06632444},
  R=61.84315461026669);

constant IdealGases.Common.DataRecord SiCL4(
  name="SiCL4",
  MM=0.1698975,
  Hf=-3897644.16780706,
  H0=114512.797421975,
  Tlimit=1000,
  alow={117164.6285,-2185.821461,19.08250496,-0.00963633572,
      8.836847690000002e-006,-4.3600424e-009,8.94023243e-013},
  blow={-72128.0211,-73.01532221000001},
  ahigh={-210033.0761,-84.76107710000001,13.06364027,-2.55402042e-005,
      5.65381847e-009,-6.48875267e-013,3.008025543e-017},
  bhigh={-83722.6813,-35.92584781},
  R=48.93816565870598);

constant IdealGases.Common.DataRecord SiF(
  name="SiF",
  MM=0.0470839032,

```

---

**1194 Modelica.Media.IdealGases.Common.SingleGasesData**

---

```
Hf=-535908.9685665653,
H0=200835.3886854478,
Tlimit=1000,
alow={14782.23768,13.57998649,2.324182643,0.00713285118,-1.022951215e-005,
    7.14839647e-009,-1.964689785e-012},
blow={-3995.38065,12.31128121},
ahigh={-365363.788,872.2860040000001,3.37721827,0.000750815243,-
2.308898619e-007,
    3.7667818e-011,-2.143080403e-015},
bhigh={-10116.97664,8.968132819999999},
R=176.5884184385121);

constant IdealGases.Common.DataRecord SiFCL(
  name="SiFCL",
  MM=0.0825369032,
  Hf=-4577667.701978914,
  H0=143364.7682579882,
  Tlimit=1000,
  alow={41769.5464,-592.6575380000001,6.40977071,0.00464168261,-
7.978461910000001e-006,
    6.00872837e-009,-1.720953157e-012},
  blow={-43982.9607,-6.259737448},
  ahigh={-128469.8945,-124.7550173,7.09289385,-3.70731177e-005,
    8.17517300999999e-009,-9.35643521e-013,4.32854434e-017},
  bhigh={-47233.0972,-8.337392707999999},
  R=100.7364182280103);

constant IdealGases.Common.DataRecord SiF2 (
  name="SiF2",
  MM=0.0660823064,
  Hf=-8971207.881448884,
  H0=169553.2527599551,
  Tlimit=1000,
  alow={40532.3665,-399.763756,4.05530388,0.01153423049,-1.757586754e-005,
    1.260369111e-008,-3.51710522e-012},
  blow={-70477.7046,3.87985442},
  ahigh={-167520.0202,-184.621209,7.13698635,-5.45076986e-005,
    1.199012961e-008,-1.369470187e-012,6.32488999e-017},
  bhigh={-72874.0102,-11.17933348},
  R=125.8199426283947);

constant IdealGases.Common.DataRecord SiF3 (
  name="SiF3",
  MM=0.08508070960000001,
  Hf=-11711662.74569953,
  H0=153014.1680905774,
  Tlimit=1000,
  alow={57553.6998,-717.959577,5.37939694,0.01813065142,-2.738373356e-005,
    1.94651947e-008,-5.39079056e-012},
  blow={-117763.1169,-3.44290204},
  ahigh={-290328.3702,-342.787694,10.25536335,-0.000101979669,
    2.250326132e-008,-2.577117181e-012,1.192923177e-016},
  bhigh={-121808.8967,-27.25261142},
  R=97.7245257954454);

constant IdealGases.Common.DataRecord SiF4 (
  name="SiF4",
```

```

MM=0.1040791128,
Hf=-15524536.638825,
H0=147634.9344899489,
Tlimit=1000,
alow={10696.70784,-95.69638640000001,2.513988593,0.0330978103,-4.57304408e-
005,
      3.091834452e-008,-8.26930405e-012},
blow={-195625.2529,11.33722022},
ahigh={-384589.232,-656.997991,13.4892572,-5.10047136e-005,5.34632958e-008,
      -4.95959811e-012,2.299314375e-016},
bhigh={-195728.4479,-46.7389465},
R=79.88607681521283);

constant IdealGases.Common.DataRecord SiH(
  name="SiH",
  MM=0.02909344,
  Hf=12670767.36198951,
  H0=314335.809034614,
  Tlimit=1000,
  alow={-6426.676299999999,74.17251210000001,3.9734916,-0.00414940888,
        1.022918384e-005,-8.592386360000002e-009,2.567093743e-012},
  blow={42817.45800000001,2.24693715},
  ahigh={404208.649,-2364.796524,7.62749914,-0.002496591233,1.10843641e-006,-
1.943991955e-010,
       1.136251507e-014},
  bhigh={57047.3768,-24.48054429},
  R=285.7851116952825);

constant IdealGases.Common.DataRecord SiHplus(
  name="SiHplus",
  MM=0.0290928914,
  Hf=39448508.27030551,
  H0=297469.951714734,
  Tlimit=1000,
  alow={-43071.5717,388.495824,2.546166863,-0.0006686325980000001,
        5.186277020000001e-006,-4.81856974e-009,1.44570264e-012},
  blow={134907.9707,9.027950089999999},
  ahigh={171169.2033,-1045.807287,4.83864824,0.0001277940891,-
6.883872450000001e-008,
       1.41884348e-011,-7.85531696e-016},
  bhigh={143172.145,-7.55329914},
  R=285.7905006994252);

constant IdealGases.Common.DataRecord SiHBr3(
  name="SiHBr3",
  MM=0.26880544,
  Hf=-1126919.157588478,
  H0=66306.05764526196,
  Tlimit=1000,
  alow={132281.1122,-2205.365507,17.75084194,-0.01047794569,1.286821723e-005,
        -7.743124260000001e-009,1.819060891e-012},
  blow={-28349.92682,-63.3119954},
  ahigh={219997.3526,-2024.458552,14.35781007,-0.000503701584,
        1.054049755e-007,-1.160828765e-011,5.21725768e-016},
  bhigh={-28420.86214,-45.3643717},
  R=30.9311894878318);

```

```
constant IdealGases.Common.DataRecord SiHCL(
  name="SiHCL",
  MM=0.06454644000000001,
  Hf=851261.7117225983,
  H0=165236.2701955367,
  Tlimit=1000,
  allow={59096.73050000001,-878.006403,7.99968793,-0.00466314127,8.594986e-006,
         -6.608073550000001e-009,1.868076396e-012},
  blow={9567.57034,-16.90014311},
  ahighelement1={101215.289,-1163.239598,7.37636655,0.0001874151313,-1.661351093e-007,
              3.96441404e-011,-2.64058008e-015},
  bhighelement1={11410.64087,-15.07678798},
  R=128.8137967020334);

constant IdealGases.Common.DataRecord SiHCL3 (
  name="SiHCL3",
  MM=0.13545244,
  Hf=-3663440.835764937,
  H0=119241.912511875,
  Tlimit=1000,
  allow={165956.5586,-2552.498122,17.50117375,-0.007885728660000001,
         8.092567490000001e-006,-3.9804206e-009,7.05917332e-013},
  blow={-49513.16600000001,-67.58519052999999},
  ahighelement1={172864.4072,-2116.727327,14.41802307,-0.000525524962,
                 1.098826178e-007,-1.209327059e-011,5.43218704999999e-016},
  bhighelement1={-51303.5194,-50.36360893000001},
  R=61.3829621673851);

constant IdealGases.Common.DataRecord SiHF (
  name="SiHF",
  MM=0.0480918432,
  Hf=-3382219.482076328,
  H0=212722.7887160707,
  Tlimit=1000,
  allow={21925.98671,-61.7522908,2.127372739,0.01123538312,-1.322503985e-005,
         8.30730739e-009,-2.010545971e-012},
  blow={-20169.94845,13.67184876},
  ahighelement1={4049752.24,-9295.526819999999,10.71754366,0.00186812422,
                 -1.129820372e-006,
                 1.9602501e-010,-1.144682011e-014},
  bhighelement1={41332.252,-47.363176},
  R=172.8873639844189);

constant IdealGases.Common.DataRecord SiHF3 (
  name="SiHF3",
  MM=0.0860886496,
  Hf=-13948505.47173643,
  H0=157333.5632854439,
  Tlimit=1000,
  allow={83258.7939,-817.859445999999,4.04171724,0.02726104628,-3.78783148e-005,
         2.630589305e-008,-7.2895825e-012},
  blow={-141614.6961,1.382806532},
  ahighelement1={78201.008,-2592.742261,14.75104214,-0.00065310472,1.372428937e-007,
                 -1.516371051e-011,
                 6.83249715e-016},
  bhighelement1={-133661.1887,-58.5145095},
  R=96.58035105245745);
```

```

constant IdealGases.Common.DataRecord SiHI3 (
  name="SiHI3",
  MM=0.40980685,
  Hf=-181731.9549441402,
  H0=46693.76317160145,
  Tlimit=1000,
  allow={111384.0054,-1974.457175,17.91396249,-0.01219208519,1.613169233e-005,
         -1.038096942e-008,2.613287761e-012},
  blow={-2256.377425,-59.9545513},
  ahighe={227942.4252,-1898.603142,14.28134804,-0.000477602695,
          1.003097496e-007,-1.107859145e-011,4.99033820000001e-016},
  bhigh={-1614.547674,-41.2294307},
  R=20.28875798440168);

constant IdealGases.Common.DataRecord SiH2 (
  name="SiH2",
  MM=0.03010138,
  Hf=9080398.639530813,
  H0=332812.05047742,
  Tlimit=1000,
  allow={-20638.63564,330.58622,2.099271145,0.00354253937,3.37887667e-006,-
         5.38384562e-009,
         2.081191273e-012},
  blow={30117.84298,12.8233357},
  ahighe={4624039.37,-11434.3611,12.6488087,0.00091148995,-8.766611539999999e-
         007,
         1.646297357e-010,-9.96509037000001e-015},
  bhigh={107247.5101,-66.0607807},
  R=276.2156419406685);

constant IdealGases.Common.DataRecord SiH2Br2 (
  name="SiH2Br2",
  MM=0.18990938,
  Hf=-1002436.003950937,
  H0=75152.80182579713,
  Tlimit=1000,
  allow={165152.8003,-2468.145554,16.08639427,-0.0094138409,1.407423872e-005,-
         9.24075144999999e-009,
         2.255641666e-012},
  blow={-12764.82614,-59.4537432},
  ahighe={566585.144,-3973.91962,15.67387912,-0.000994380229000002,
          2.084856022e-007,-2.299521351e-011,1.034736283e-015},
  bhigh={-2990.69018,-61.9047652},
  R=43.7812603042567);

constant IdealGases.Common.DataRecord SiH2CL2 (
  name="SiH2CL2",
  MM=0.10100738,
  Hf=-3172976.073629472,
  H0=132514.5548770793,
  Tlimit=1000,
  allow={189428.1754,-2650.960252,15.49874174,-0.00653313074999999,
         9.30609826000001e-006,-5.63969867e-009,1.213664264e-012},
  blow={-27209.12007,-60.06331649},
  ahighe={540020.458,-4062.33729,15.73118788,-0.001015038389,2.127060243e-007,
          -2.345079226e-011,1.054871984e-015},
  bhigh={-18230.98225,-65.44540499},
  R=82.31549021467541);

```

---

**1198 Modelica.Media.IdealGases.Common.SingleGasesData**

---

```
constant IdealGases.Common.DataRecord SiH2F2(
  name="SiH2F2",
  MM=0.0680981864,
  Hf=-11612291.63072102,
  H0=176042.9555286953,
  Tlimit=1000,
  allow={127218.8427,-1241.72431,4.89943589,0.02123882008,-2.7257087e-005,
         1.860857446e-008,-5.22660433e-012},
  blow={-89804.6691999999,-5.1118155},
  ahighelement1={472653.4080000001,-4388.51769,15.96032597,-0.001102993658,
                 2.315838331e-007,-2.556974559e-011,1.151499711e-015},
  bhighelement1={-73161.8446,-70.9898927},
  R=122.0953514262753);

constant IdealGases.Common.DataRecord SiH2I2(
  name="SiH2I2",
  MM=0.28391032,
  Hf=-134105.7274705618,
  H0=52542.17599416605,
  Tlimit=1000,
  allow={153996.3319,-2414.436989,16.84409354,-0.0121203758,1.808453226e-005,-
         1.207740133e-008,
         3.040799276e-012},
  blow={5073.07325,-60.9872082},
  ahighelement1={594239.083,-3924.81448,15.63850289,-0.0009805706340000001,
                 2.054871384e-007,-2.265587199e-011,1.019168663e-015},
  bhighelement1={15136.27352,-59.4081252},
  R=29.28555749576134);

constant IdealGases.Common.DataRecord SiH3(
  name="SiH3",
  MM=0.03110932,
  Hf=6569007.80859241,
  H0=330350.8080536636,
  Tlimit=1000,
  allow={4341.14282,227.7185085,0.650825035,0.01221438558,-4.34760427e-006,-
         1.774916828e-009,
         1.184191367e-012},
  blow={22599.93826,19.68347482},
  ahighelement1={605632.122,-4721.25405999999,13.29129523,-0.001256824868,
                 2.68828594e-007,-3.010741582e-011,1.370945857e-015},
  bhighelement1={49744.2064,-61.405031},
  R=267.2662726154092);

constant IdealGases.Common.DataRecord SiH3Br(
  name="SiH3Br",
  MM=0.11101332,
  Hf=-704789.2991579749,
  H0=106017.7103071956,
  Tlimit=1000,
  allow={152106.5525,-1987.744479,11.31460743,-0.001806099407,7.76926247e-006,
         -6.23347816999999e-009,1.585331846e-012},
  blow={-924.886888,-38.4646369},
  ahighelement1={927928.578,-5917.93579,16.99209298,-0.001487507364,3.123451161e-007,
                 -3.44907147e-011,1.553431675e-015},
  bhighelement1={22415.60128,-79.4861159},
  R=74.89616561327956);
```

```

constant IdealGases.Common.DataRecord SiH3CL(
  name="SiH3CL",
  MM=0.0665623199999999,
  Hf=-2130905.292964548,
  H0=171879.615974924,
  Tlimit=1000,
  allow={163338.8301,-2030.302118,10.68016138,0.0004942370140000001,
         4.30517476e-006,-3.74832727e-009,8.89350421e-013},
  blow={-8180.71498,-36.88956824},
  ahighe={908778.094,-5958.94956,17.02076016,-0.001498450667,3.146813781e-007,
          -3.47516846e-011,1.565281681e-015},
  bhigh={14937.93206,-81.27277864},
  R=124.9125931908624);

constant IdealGases.Common.DataRecord SiH3F(
  name="SiH3F",
  MM=0.0501077232,
  Hf=-7515009.183255008,
  H0=218162.4169265787,
  Tlimit=1000,
  allow={126783.629,-1177.593516,4.49918896,0.01658543011,-1.678598921e-005,
         1.019153931e-008,-2.803920232e-012},
  blow={-40103.9297,-4.48108092},
  ahighe={862431.317,-6103.37057,17.12733597,-0.001540724131,3.23962872e-007,-
3.58105689e-011,
          1.61415138e-015},
  bhigh={-12614.87616,-83.99227209999999},
  R=165.9319455967618);

constant IdealGases.Common.DataRecord SiH3I(
  name="SiH3I",
  MM=0.15801379,
  Hf=-13239.35081868488,
  H0=76437.81596530277,
  Tlimit=1000,
  allow={145938.4799,-1982.982746,11.96516737,-0.00393750595,1.087991733e-005,
         -8.43313592e-009,2.196078188e-012},
  blow={8063.20831,-40.6455332},
  ahighe={936114.918,-5859.72029,16.95376698,-0.00147349713,3.094475885e-007,-
3.41746247e-011,
          1.539334807e-015},
  bhigh={31274.99231,-78.01700080000001},
  R=52.61864803065607);

constant IdealGases.Common.DataRecord SiH4(
  name="SiH4",
  MM=0.03211726,
  Hf=1080415.950800286,
  H0=328016.8046713823,
  Tlimit=1000,
  allow={78729.9329,-552.608705,2.498944303,0.01442118274,-8.467107309999999e-
006,
         2.726164641e-009,-5.43675437e-013},
  blow={6269.66906,4.96546183},
  ahighe={1290378.74,-7813.39978,18.28851664,-0.001975620946,4.15650215e-007,-
4.59674561e-011,
          2.072777131e-015},
  bhigh={47668.8795,-98.0169746},

```

---

**1200 Modelica.Media.IdealGases.Common.SingleGasesData**

---

```
R=258.8786216507884);

constant IdealGases.Common.DataRecord SiI(
  name="SiI",
  MM=0.15498997,
  Hf=1696583.462787947,
  H0=64140.40856966423,
  Tlimit=1000,
  allow={94920.8134,-1573.960062,12.37908109,-0.0159386482,1.692049005e-005,-
9.21555831999999e-009,
  2.046186848e-012},
  blow={37797.3567,-40.6640785},
  ahight={852940.2790000001,-2442.039471,7.72297752,-0.001985601243,
  6.54286662999999e-007,-9.49415829e-011,4.89004137e-015},
  bhigh={45781.6094,-17.33975081},
  R=53.64522620399243};

constant IdealGases.Common.DataRecord SiI2(
  name="SiI2",
  MM=0.28189444,
  Hf=328016.4021681308,
  H0=48927.9426724415,
  Tlimit=1000,
  allow={11633.52403,-451.448217,8.707978130000001,-0.00358066493,
  4.25371443e-006,-2.673478266e-009,6.89155898e-013},
  blow={11262.47142,-11.55818335},
  ahight={10021.68907,14.33017148,6.70133055,0.00039607699,-2.073866165e-007,
  4.5534571e-011,-3.01679828e-015},
  bhigh={9007.368100000002,0.2887336913},
  R=29.49498400890774);

constant IdealGases.Common.DataRecord SiN(
  name="SiN",
  MM=0.0420922,
  Hf=9590100.707494499,
  H0=207547.6454069876,
  Tlimit=1000,
  allow={-14646.72152,137.4993497,3.67850858,-0.0061584992,2.309417067e-005,-
2.294461481e-008,
  7.39566568e-012},
  blow={46732.1299,6.494564115},
  ahight={-2932685.132,5853.68859,1.321451677,0.001258329284,-3.77388636e-007,
  6.88776104e-011,-4.18984259e-015},
  bhigh={6527.14881,25.53145732},
  R=197.5299936805394);

constant IdealGases.Common.DataRecord SiO(
  name="SiO",
  MM=0.0440849,
  Hf=-2242092.371764482,
  H0=197689.1180426858,
  Tlimit=1000,
  allow={-47227.7105,806.3137640000001,-1.636976133,0.01454275546,-
1.723202046e-005,
  1.04239734e-008,-2.559365273e-012},
  blow={-16665.85903,33.557957},
  ahight={-176513.4162,-31.9917709,4.47744193,4.59176471e-006,3.55814315e-008,
```

```

-1.327012559e-011, 1.613253297e-015},
bhigh={-13508.4236, -0.838695733},
R=188.601357834542);

constant IdealGases.Common.DataRecord SiO2(
  name="SiO2",
  MM=0.0600843,
  Hf=-5360359.977564856,
  H0=175462.1922865041,
  Tlimit=1000,
  allow={-33629.4878, 473.407892, 0.2309770671, 0.01850230806, -2.242786671e-005,
         1.364981554e-008, -3.35193503e-012},
  blow={-42264.8749, 22.95803206},
  ahigh={-146403.1193, -626.144106, 7.96456371, -0.0001854119096, 4.09521467e-008,
         -4.69720676e-012, 2.17805428e-016},
  bhigh={-37918.3477, -20.45285414},
  R=138.3801092797952);

constant IdealGases.Common.DataRecord SiS(
  name="SiS",
  MM=0.0601505,
  Hf=1798728.306497868,
  H0=148495.9726020565,
  Tlimit=1000,
  allow={35994.4929, -423.9723299999999, 4.65401442, 0.001588470782, -3.31025436e-
006,
         2.706096479e-009, -8.113517820000001e-013},
  blow={14115.15571, -1.183201858},
  ahigh={-2102323.897, 6228.83618, -3.004120882, 0.004495499930000001, -
1.368821364e-006,
         1.998097253e-010, -9.882035800000001e-015},
  bhigh={-27955.38166, 54.05828786},
  R=138.2278119051379);

constant IdealGases.Common.DataRecord SiS2(
  name="SiS2",
  MM=0.09221550000000001,
  Hf=76155.49446676533,
  H0=132333.6857686614,
  Tlimit=1000,
  allow={43977.4566, -743.001312, 7.94158836, 0.00203808373, -4.6463376e-006,
         3.84801436e-009, -1.157260149e-012},
  blow={2801.077853, -17.36888325},
  ahigh={-126648.9071, -97.7423522, 7.57280368, -2.905413928e-005,
         6.40558784e-009, -7.329273340000001e-013, 3.38981978e-017},
  bhigh={-1244.672263, -13.52710463},
  R=90.1634974597546);

constant IdealGases.Common.DataRecord Si2(
  name="Si2",
  MM=0.056171,
  Hf=10329093.73164088,
  H0=183299.6386035499,
  Tlimit=1000,
  allow={12375.96221, -102.4904376, 4.35484852, 0.001281063335, -2.531991623e-006,
         2.265694244e-009, -7.001290140000001e-013},
  blow={69069.4285, 3.2511252},

```

---

**1202 Modelica.Media.IdealGases.Common.SingleGasesData**

---

```
ahigh={1370060.657,-4207.06004,9.337432890000001,-0.002749217168,
        9.586345959999999e-007,-1.372449748e-010,6.765028100000001e-015},
bhigh={95108.84539999999,-31.6838519},
R=148.0207224368446;

constant IdealGases.Common.DataRecord Si2C(
  name="Si2C",
  MM=0.06818170000000001,
  Hf=8126724.678322773,
  H0=169874.7024494842,
  Tlimit=1000,
  allow={-4553.3662,131.4796415,2.469106923,0.0127652068,-1.656910776e-005,
         1.065289663e-008,-2.739192976e-012},
  blow={64700.4992,14.6883898},
  ahight=-125382.9442,-341.427779,7.25436533,-0.0001017635503,
  2.250902158e-008,-2.584074852e-012,1.198884876e-016},
  bhigh={66080.0938,-11.46216579},
  R=121.9458007060545);

constant IdealGases.Common.DataRecord Si2F6(
  name="Si2F6",
  MM=0.1701614192,
  Hf=-14006054.31715863,
  H0=156876.2891465118,
  Tlimit=1000,
  allow={15807.6745,-297.5292881,7.5125915,0.0455279623,-6.42187684e-005,
         4.357408730000001e-008,-1.166762282e-011},
  blow={-288670.7444,-7.72178477},
  ahight=-918630.314,157.3948683,21.41256129,-0.0002681898667,7.21228011e-008,
  -9.18425266e-012,4.53723836e-016},
  bhigh={-296617.6753,-78.4902247},
  R=48.8622628977227);

constant IdealGases.Common.DataRecord Si2N(
  name="Si2N",
  MM=0.0701777,
  Hf=5663907.480581439,
  H0=168460.7503523199,
  Tlimit=1000,
  allow={23182.71938,-367.484379,5.4096586,0.00817638939999999,-1.220312934e-
005,
        8.5806091e-009,-2.354927059e-012},
  blow={48092.7803,-3.042205654},
  ahight=-280502.6986,250.2366876,7.0924561,0.0002853120795,-9.79210181e-008,
  1.552154244e-011,-8.03626045e-016},
  bhigh={43420.1479,-10.0957091},
  R=118.4774080655251);

constant IdealGases.Common.DataRecord Si3(
  name="Si3",
  MM=0.0842565,
  Hf=7451856.901247975,
  H0=148591.4202465092,
  Tlimit=1000,
  allow={-11142.08177,157.5785843,2.486135003,0.01631637255,-2.208240021e-005,
         1.372008287e-008,-3.2623307e-012},
  blow={73282.53850000001,15.88081347},
```

```

ahigh={-1699395.561,4697.81538,2.618198124,0.001959082075,-2.581160603e-007,
       6.10344486e-012,6.08630924e-016},
bhigh={42779.1681,25.86540384},
R=98.6804816245631;

constant IdealGases.Common.DataRecord Sn(
  name="Sn",
  MM=0.11871,
  Hf=2537275.713924691,
  H0=52352.05964114227,
  Tlimit=1000,
  allow={-124869.2263,1618.84119,-4.60239735,0.01045433308,2.99826555e-006,-
1.068699386e-008,
        4.32342131e-012},
  blow={27483.64008,48.0506723},
  ahigh={-5145695.64,11405.75108,-4.17963206,0.002236390679,-3.60321977e-007,
         2.440237836e-011,-2.937628285e-016},
  bhigh={-42150.1357,59.8145093000001},
  R=70.0401988038076);

constant IdealGases.Common.DataRecord Snplus(
  name="Snplus",
  MM=0.1187094514,
  Hf=8558290.144705359,
  H0=52206.6939650603,
  Tlimit=1000,
  allow={-5571.29778,122.2323189,1.566361415,0.00339706141,-6.31292229e-006,
         5.58545208e-009,-1.68902134e-012},
  blow={120902.412,11.62634765},
  ahigh={4622916.850000001,-11859.58712,12.37026473,-0.002773624217,
         3.09851349e-007,-5.362951439999999e-012,-8.663474691e-016},
  bhigh={199432.2977,-68.3710828},
  R=70.04052248530566);

constant IdealGases.Common.DataRecord Snminus(
  name="Snminus",
  MM=0.1187105486,
  Hf=1512047.169496444,
  H0=54674.57674608034,
  Tlimit=1000,
  allow={272279.6,-3369.9358,17.43906405,-0.02810527618,2.790684767e-005,-
1.442400675e-008,
        3.070159757e-012},
  blow={37532.2775,-80.0785697},
  ahigh={-64477.1322,743.05551,1.921380256,0.0002361623997,-5.28066836e-008,
         6.10233938e-012,-2.843422486e-016},
  bhigh={16457.39414,12.65436865},
  R=70.03987512530121);

constant IdealGases.Common.DataRecord SnBr(
  name="SnBr",
  MM=0.198614,
  Hf=380859.4560302899,
  H0=50279.03370356571,
  Tlimit=1000,
  allow={25641.93542,-525.912233,7.68741069,-0.00965420542000001,
         1.482863248e-005,-9.94223698e-009,2.475051091e-012},

```

---

**1204 Modelica.Media.IdealGases.Common.SingleGasesData**

---

```
blow={10204.83889,-11.26550396},
ahigh={1815578.541,-6728.26458,13.32464516,-0.0046081023,1.236439537e-006,-
1.528863587e-010,
6.971513500000001e-015},
bhigh={49255.8309,-56.4096278},
R=41.86246689558642);

constant IdealGases.Common.DataRecord SnBr2(
  name="SnBr2",
  MM=0.278518,
  Hf=-427170.2080296426,
  H0=52403.82309222384,
  Tlimit=1000,
  allow={-4995.75306,-136.6498896,7.54356385,-0.001181772507,1.442445852e-006,
  -9.254217260000001e-010,2.423731634e-013},
  blow={-15755.06969,-3.59558402},
  ahigh={-19408.94091,-2.335607932,7.00196926,-8.622241869999999e-007,
  2.039869794e-010,-2.464780359e-014,1.189850894e-018},
  bhigh={-16448.28122,-0.434935475},
  R=29.85254812974386);

constant IdealGases.Common.DataRecord SnBr3(
  name="SnBr3",
  MM=0.358422,
  Hf=-442820.0445285167,
  H0=53792.60759663191,
  Tlimit=1000,
  allow={-8050.41647,-289.7962523,11.14443355,-0.002475383807,3.01005135e-006,
  -1.92569532e-009,5.032605089999999e-013},
  blow={-20700.66649,-18.95965383},
  ahigh={-38930.0453,-4.99084417,10.00418437,-1.82499691e-006,4.30564666e-010,
  -5.19188793e-014,2.502455595e-018},
  bhigh={-22172.99473,-12.29854098},
  R=23.19743765728666);

constant IdealGases.Common.DataRecord SnBr4(
  name="SnBr4",
  MM=0.4383260000000001,
  Hf=-739670.8887905348,
  H0=57133.31629882781,
  Tlimit=1000,
  allow={-5832.84092,-427.019847,14.67466525,-0.00360426323,
  4.366686289999999e-006,-2.78587513e-009,7.26508787e-013},
  blow={-40829.1645,-34.4710436},
  ahigh={-51768.1292,-7.42612695,13.00619502,-2.692587978e-006,6.3367598e-010,
  -7.62708719e-014,3.671112e-018},
  bhigh={-43001.6634,-24.71455118},
  R=18.96869453329257);

constant IdealGases.Common.DataRecord SnCl(
  name="SnCl",
  MM=0.154163,
  Hf=224819.2886749739,
  H0=62557.91597205554,
  Tlimit=1000,
  allow={33515.4973,-683.220862,8.39391773,-0.01164434373,1.812159499e-005,-
  1.26199917e-008,
```

```

    3.29521347e-012},
blow={6051.79089,-16.67834578},
ahigh={774160.568,-3602.04848,9.749059819999999,-0.002596198909,
6.36771216e-007,-6.75429557e-011,2.484167602e-015},
bhigh={24424.47662,-32.25454098},
R=53.93299300091461);

constant IdealGases.Common.DataRecord SnCL2 (
  name="SnCL2",
  MM=0.189616,
  Hf=-1068729.869842207,
  H0=72154.86562315417,
  Tlimit=1000,
  alow={9524.6659,-432.984564,8.64472028,-0.00345840961,4.11785203e-006,-
2.592639269e-009,
6.6923636e-013},
blow={-24329.21127,-12.99143851},
ahigh={-39035.88490000001,-7.94796268,7.00648762,-2.776594714e-006,
6.461341420000001e-010,-7.1190674e-015,3.68813118e-018},
bhigh={-26545.29653,-3.368081174},
R=43.84900008438107);

constant IdealGases.Common.DataRecord SnCL3 (
  name="SnCL3",
  MM=0.225069,
  Hf=-1299034.76267278,
  H0=77994.05959950059,
  Tlimit=1000,
  alow={17600.76813,-814.280548,13.06326984,-0.00639559619,7.57462457e-006,-
4.74971146e-009,
1.222182656e-012},
blow={-34134.3096,-34.5767967},
ahigh={-74829.66399999999,-15.28523914,10.01240305,-5.28558124e-006,
1.226095111e-009,-1.459895108e-013,6.96887978e-018},
bhigh={-38308.7827,-16.63063137},
R=36.94188004567489);

constant IdealGases.Common.DataRecord SnCL4 (
  name="SnCL4",
  MM=0.260522,
  Hf=-1836566.466555608,
  H0=86265.38641650225,
  Tlimit=1000,
  alow={42459.6787,-1280.574318,17.68445328,-0.00957477952,1.115620323e-005,-
6.90722927e-009,
1.759652856e-012},
blow={-55040.09039999999,-58.27352999},
ahigh={-107833.6862,-25.54305872,13.02040391,-8.5951259e-006,
1.976570888e-009,-2.337929994e-013,1.110287624e-017},
bhigh={-61635.6203,-30.72702622},
R=31.91466363685217);

constant IdealGases.Common.DataRecord SnF (
  name="SnF",
  MM=0.1377084032,
  Hf=-689987.5954701362,
  H0=66343.88162014502,

```

---

**1206 Modelica.Media.IdealGases.Common.SingleGasesData**

---

```
Tlimit=1000,
alow={60985.03520000001,-986.0306290000001,8.994731209999999,-0.01223012156,
      1.840583578e-005,-1.267926607e-008,3.29133102e-012},
blow={-7882.61448,-22.21921734},
ahigh={25897.15476,-1484.478214,7.37851893,-0.00132424444,2.847634043e-007,
      -2.360371345e-011,6.11678371e-016},
bhigh={-4776.68897,-16.81202516},
R=60.3773757213968);

constant IdealGases.Common.DataRecord SnF2(
  name="SnF2",
  MM=0.1567068064,
  Hf=-3260591.947077035,
  H0=78127.10424810239,
  Tlimit=1000,
  alow={70423.9185,-1137.505955,10.08659352,-0.00472285593,4.13570292e-006,-
1.921143972e-009,
      3.64076028e-013},
  blow={-57566.7082,-25.71566624},
  ahigh={-102775.8772,-44.5175162,7.03326876,-1.329882797e-005,
      2.934271938e-009,-3.35841592e-013,1.553358669e-017},
  bhigh={-63621.2588,-6.81008452},
  R=53.05750395280852);

constant IdealGases.Common.DataRecord SnF3(
  name="SnF3",
  MM=0.1757052096,
  Hf=-3680200.948350254,
  H0=84230.4051979572,
  Tlimit=1000,
  alow={109555.5529,-1950.557979,15.41407832,-0.008546518770000001,
      7.80055182e-006,-3.82594222e-009,7.78774776e-013},
  blow={-70567.9492,-54.9463948},
  ahigh={-182931.9689,-75.5735868,10.05671455,-2.275132668e-005,
      5.03469582e-009,-5.77653096e-013,2.67721417e-017},
  bhigh={-80918.97590000001,-21.91673972},
  R=47.32057756812237);

constant IdealGases.Common.DataRecord SnF4(
  name="SnF4",
  MM=0.1947036128,
  Hf=-5263213.672632991,
  H0=96567.20144845716,
  Tlimit=1000,
  alow={139044.1119,-2168.784371,16.91802917,-0.002318608529,-2.045847976e-
006,
      3.4300879e-009,-1.28922886e-012},
  blow={-115356.3396,-63.7060475},
  ahigh={-263569.2764,-144.5469969,13.10690681,-4.23775579e-005,
      9.28687195e-009,-1.057062694e-012,4.86710868e-017},
  bhigh={-127153.9967,-38.039219},
  R=42.70322404618472);

constant IdealGases.Common.DataRecord SnI(
  name="SnI",
  MM=0.24561447,
  Hf=703236.3443407874,
```

```

H0=41435.35191554471,
Tlimit=1000,
alow={26702.12235,-500.602635,7.49067315,-0.008746346550000001,
      1.289688397e-005,-8.21134492999999e-009,1.936120475e-012},
blow={21772.49172,-9.35480684999999},
ahigh={-55549.91850000001,-1530.457453,7.93260139,-0.002108885464,
       7.8098036e-007,-1.348104887e-010,8.207184700000001e-015},
bhigh={27448.64953,-16.26822892},
R=33.85171891542058);

constant IdealGases.Common.DataRecord SnI2(
  name="SnI2",
  MM=0.37251894,
  Hf=-21654.5687583026,
  H0=39953.03701873522,
  Tlimit=1000,
  alow={-5739.25489,-94.3502445,7.37795666,-0.0008258348939999999,
        1.01168704e-006,-6.5084516e-010,1.7081866e-013},
  blow={-2622.6445,-0.960581496},
  ahigh={-15594.5339,-1.594789414,7.00135116,-5.93532424e-007,
        1.407432564e-010,-1.703457321e-014,8.23361171e-019},
  bhigh={-3100.608319,1.235038509},
  R=22.31959534728624);

constant IdealGases.Common.DataRecord SnI3(
  name="SnI3",
  MM=0.49942341,
  Hf=-16053.59268200904,
  H0=40665.61477364467,
  Tlimit=1000,
  alow={-10699.78099,-138.769354,10.55637633,-0.001216435658,1.490873499e-006,
        -9.59449962999999e-010,2.518810438e-013},
  blow={-3314.22988,-12.49558209},
  ahigh={-25175.3487,-2.34828289,10.0019913,-8.75276483999999e-007,
        2.07646498e-010,-2.514060336e-014,1.215482266e-018},
  bhigh={-4017.05561,-9.26390846},
  R=16.64814230474298);

constant IdealGases.Common.DataRecord SnI4(
  name="SnI4",
  MM=0.6263278800000001,
  Hf=-189763.2961828236,
  H0=42702.80447997939,
  Tlimit=1000,
  alow={-12831.46793,-145.9885093,13.58634885,-0.001283554646,
        1.574557147e-006,-1.013990168e-009,2.66337043e-013},
  blow={-17511.78045,-23.67973168},
  ahigh={-28018.40706,-2.474699366,13.00210215,-9.25096233999999e-007,
        2.196515817e-010,-2.661071614e-014,1.287167546e-018},
  bhigh={-18250.83832,-20.27484281},
  R=13.27495113262402);

constant IdealGases.Common.DataRecord SnO(
  name="SnO",
  MM=0.1347094,
  Hf=162654.2319986579,
  H0=65875.92996479831,

```

---

**1208 Modelica.Media.IdealGases.Common.SingleGasesData**

---

```
Tlimit=1000,
alow={25477.88022,-229.3583066,3.42159353,0.00471330004,-7.36597132e-006,
      5.37973887e-009,-1.519277855e-012},
blow={2853.047679,6.67396527},
ahigh={-2555955.548,7870.0535,-5.43976873,0.00630781918,-2.06887137e-006,
       3.30707488e-010,-1.848169716e-014},
bhigh={-48416.6391,71.75952460000001},
R=61.72154281735352);

constant IdealGases.Common.DataRecord SnO2(
  name="SnO2",
  MM=0.1507088,
  Hf=77501.64555752551,
  H0=78268.02416315438,
  Tlimit=1000,
  alow={47076.7408,-611.456867,6.11743391,0.00751401466,-1.238371787e-005,
        9.22091599e-009,-2.632417597e-012},
  blow={2981.068066,-8.14492641},
  ahigh={-157355.8926,-142.7316788,7.60570076,-4.19800853e-005,
        9.218742430000002e-009,-1.051372983e-012,4.84957927e-017},
  bhigh={-523.453187,-14.28360169},
  R=55.16912084762137);

constant IdealGases.Common.DataRecord SnS(
  name="SnS",
  MM=0.150775,
  Hf=736853.0061349694,
  H0=61688.68844304427,
  Tlimit=1000,
  alow={27248.77813,-492.412638,6.22165473,-0.00340940696,3.98491368e-006,-
2.470702874e-009,
        6.33934146e-013},
  blow={14524.95307,-6.791285229},
  ahigh={-1797985.129,6141.73632,-4.01950488,0.005993829179999999,-
2.188735204e-006,
        3.899542e-010,-2.40656363e-014},
  bhigh={-26113.49815,62.56820705},
  R=55.14489802686122);

constant IdealGases.Common.DataRecord SnS2(
  name="SnS2",
  MM=0.18284,
  Hf=818451.0774447605,
  H0=74573.91708597681,
  Tlimit=1000,
  alow={38771.4541,-848.695394,10.4208399,-0.005686902670000001,
        6.37382977e-006,-3.8251581e-009,9.50248096e-013},
  blow={20060.27952,-27.23724769},
  ahigh={-67640.84050000001,-19.70189679,7.51533387,-6.33255918e-006,
        1.434207625e-009,-1.676383081e-013,7.88691190000001e-018},
  bhigh={15650.53629,-9.922744838},
  R=45.47403194049442);

constant IdealGases.Common.DataRecord Sn2(
  name="Sn2",
  MM=0.23742,
  Hf=1774676.678460113,
```

```

H0=47903.02838850982,
Tlimit=1000,
alow={-132966.6654,1639.179787,-1.902868972,0.01290431677,-1.120542242e-005,
      4.50299386e-009,-6.05623777e-013},
blow={40974.5271,44.35408029999999},
ahigh={-4275783.04,12364.21656,-8.43833617,0.00708805629,-1.712012737e-006,
       1.805262952e-010,-7.05865680999999e-015},
bhigh={-30014.1872,100.5138538},
R=35.0200994019038);

constant IdealGases.Common.DataRecord Sr(
  name="Sr",
  MM=0.08762,
  Hf=1831773.567678612,
  H0=70730.74640493038,
  Tlimit=1000,
  alow={4.19064984,-0.0630443758,2.500373027,-1.115455943e-006,
        1.785248643e-009,-1.456209589e-012,4.750132981e-016},
  blow={18558.52648,5.55577284},
  ahigh={14894144.1,-43753.3505,51.3726628,-0.02592566025,6.58299e-006,-
6.94961179999999e-010,
        2.417779662e-014},
  bhigh={297754.5522,-345.489077},
  R=94.89239899566309);

constant IdealGases.Common.DataRecord Srplus(
  name="Srplus",
  MM=0.08761945140000001,
  Hf=8173599.772162005,
  H0=70731.18926193139,
  Tlimit=1000,
  alow={11.27287678,-0.134695187,2.500651495,-1.635163061e-006,
        2.249493149e-009,-1.610827539e-012,4.698612333e-016},
  blow={85389.81180000001,6.24725924},
  ahigh={3145095.058,-9514.75688999999,13.50086948,-0.00605971712,
        1.594068746e-006,-1.718800946e-010,6.322256169e-015},
  bhigh={145799.1907,-72.3641693},
  R=94.8929931328011);

constant IdealGases.Common.DataRecord SrBr(
  name="SrBr",
  MM=0.167524,
  Hf=-381543.4087056183,
  H0=60332.38222583033,
  Tlimit=1000,
  alow={-754.9812420000001,-73.5953201,4.82803581,-0.000716058465,
        1.028527433e-006,-6.88837072e-010,1.880257878e-013},
  blow={-8686.168450000001,4.15115309},
  ahigh={3009976.114,-9193.19224,15.26033745,-0.00600925395,1.692611858e-006,
        -2.039438429e-010,8.807775840000001e-015},
  bhigh={49228.06129999999,-70.6825383},
  R=49.63152742293642);

constant IdealGases.Common.DataRecord SrBr2(
  name="SrBr2",
  MM=0.247428,
  Hf=-1643814.972436426,

```

## 1210 Modelica.Media.IdealGases.Common.SingleGasesData

---

```
H0=66017.78699257966,
Tlimit=1000,
alow={-4632.45866,-72.4493864,7.80021716,-0.000677212531,8.53778024e-007,-
5.63235174e-010,
1.510568795e-013},
blow={-50822.3833,-5.28507476},
ahigh={-11861.73059,-1.404383013,7.50121557,-5.415983190000001e-007,
1.296969275e-010,-1.580911652e-014,7.68158227e-019},
bhigh={-51185.75210000001,-3.55203711},
R=33.60360185589343);

constant IdealGases.Common.DataRecord SrCL(
  name="SrCL",
  MM=0.123073,
  Hf=-1038959.658089102,
  H0=79628.49690833896,
  Tlimit=1000,
  alow={3555.34632,-172.9385697,5.21345102,-0.001542910668,2.025835568e-006,-
1.327076633e-009,
3.54717596e-013},
  blow={-15882.88066,0.4818144784},
  ahigh={2070217.324,-6199.89173,11.50315072,-0.00367568546,9.46173235e-007,-
9.203765750000001e-011,
2.707242192e-015},
  bhigh={22714.71632,-45.61058514},
  R=67.55723838697359);

constant IdealGases.Common.DataRecord SrCLplus(
  name="SrCLplus",
  MM=0.1230724514,
  Hf=3316031.795560708,
  H0=77862.77018936505,
  Tlimit=1000,
  alow={1689.043529,-174.9284676,5.06532864,-0.0009857327839999999,
1.071655138e-006,-5.98812712999999e-010,1.395650932e-013},
  blow={48611.9026,0.2492001711},
  ahigh={-21324.75783,-4.05794827,4.50314939,3.82620238e-005,2.67695151e-009,
5.667610500000001e-014,1.613288103e-018},
  bhigh={47693.2603,3.637722496},
  R=67.55753952586014);

constant IdealGases.Common.DataRecord SrCL2(
  name="SrCL2",
  MM=0.158526,
  Hf=-3058261.118050036,
  H0=90938.38234737518,
  Tlimit=1000,
  alow={1913.170011,-268.1156951,8.0813343,-0.002393329295,2.976397494e-006,-
1.943902862e-009,
5.17430771e-013},
  blow={-59101.1185,-8.45782754300001},
  ahigh={-25927.16099,-5.26378585,7.00447524,-1.970397814e-006,4.6796712e-010,
-5.67021588e-014,2.742892387e-018},
  bhigh={-60453.9953,-2.192202775},
  R=52.44863303180551);

constant IdealGases.Common.DataRecord SrF(
```

```

name="SrF",
MM=0.1066184032,
Hf=-2847095.612851947,
H0=87068.57091628248,
Tlimit=1000,
alow={27250.08615,-485.001972,6.1126972,-0.003008469209,3.3684658e-006,-
1.987813072e-009,
        4.88416109e-013},
blow={-35368.9039,-6.68373918},
ahigh={870234.0489999999,-2589.81877,7.20026422,-0.001173446296,
        1.996165066e-007,1.103842667e-011,-2.343183859e-015},
bhigh={-21336.71699,-16.67390309},
R=77.98346017622593);

constant IdealGases.Common.DataRecord SrFplus(
name="SrFplus",
MM=0.1066178546,
Hf=1964664.481252842,
H0=85415.44973087461,
Tlimit=1000,
alow={40096.242,-594.494608,6.18233033,-0.00271615975,2.641369127e-006,-
1.383123292e-009,
        3.052935853e-013},
blow={26971.49351,-8.313659449999999},
ahigh={-47520.8206,-25.26802345,4.51929528,2.55642675e-005,3.38254183e-009,
        -1.511394636e-013,9.53749118e-018},
bhigh={23838.61417,1.892197325},
R=77.98386143853246);

constant IdealGases.Common.DataRecord SrF2(
name="SrF2",
MM=0.1256168064,
Hf=-6247522.266256246,
H0=105635.5545112792,
Tlimit=1000,
alow={41707.5965,-850.013241000001,10.02806064,-0.00609848519,
        7.05626133e-006,-4.35937431e-009,1.11130831e-012},
blow={-92178.59849999999,-23.25100053},
ahigh={-61262.6543,-19.64060822,7.01555775,-6.51252678e-006,
        1.490473613e-009,-1.75643704e-013,8.31709384999999e-018},
bhigh={-96567.0125,-5.40230876},
R=66.1891687767028);

constant IdealGases.Common.DataRecord SrH(
name="SrH",
MM=0.0886279399999999,
Hf=2473566.518639608,
H0=98423.87174969881,
Tlimit=1000,
alow={-43177.9689,767.64297,-1.594398638,0.01500711077,-1.826296898e-005,
        1.136757659e-008,-2.855588359e-012},
blow={21796.85693,33.2467423},
ahigh={-226982.5428,1345.209458,1.152194421,0.003109555289,-1.188080968e-
006,
        2.073091147e-010,-1.300839287e-014},
bhigh={17355.47062,21.12916944},
R=93.81321511026886);

```

---

**1212 Modelica.Media.IdealGases.Common.SingleGasesData**

---

```
constant IdealGases.Common.DataRecord SrI(
  name="SrI",
  MM=0.21452447,
  Hf=-36600.37477309698,
  H0=47934.50369554578,
  Tlimit=1000,
  allow={-2467.498501,-25.35495809,4.60329848,-0.0001613676212,
         2.694294049e-007,-1.564795762e-010,3.82648228e-014},
  blow={-2175.538291,6.41148188},
  ahigh={2201136.857,-7082.2307,13.32597928,-0.00530586442,1.63607115e-006,-
2.205540591e-010,
        1.078496824e-014},
  bhigh={42190.77170000001,-55.308332},
  R=38.75768577822381);

constant IdealGases.Common.DataRecord SrI2(
  name="SrI2",
  MM=0.34142894,
  Hf=-814866.0216090645,
  H0=48916.93715242767,
  Tlimit=1000,
  allow={-4391.84969,-37.5726735,7.6569935,-0.00035617213,
         4.508689010000001e-007,-2.983230473e-010,8.01865866e-014},
  blow={-35533.0346,-2.598281578},
  ahigh={-8090.381920000001,-0.734826323,7.50064006,-2.863515042e-007,
         6.87655697e-011,-8.398807359999999e-015,4.08696816e-019},
  bhigh={-35721.0642,-1.693119087},
  R=24.35198375392549);

constant IdealGases.Common.DataRecord SrO(
  name="SrO",
  MM=0.1036194,
  Hf=-137114.1986925228,
  H0=87243.40229725321,
  Tlimit=1000,
  allow={42248.9167,-591.3517869999999,5.964363,-0.002105712736,
         2.028373972e-006,-1.197885492e-009,3.46628992e-013},
  blow={101.8053721,-7.51142747},
  ahigh={-51729330.5,151082.1,-161.4513515,0.08516051790000001,-2.091669402e-
005,
        2.452307103e-009,-1.108841292e-013},
  bhigh={-968782.9519999999,1197.846971},
  R=80.24049550566787);

constant IdealGases.Common.DataRecord SrOH(
  name="SrOH",
  MM=0.10462734,
  Hf=-1855019.213907187,
  H0=105581.0651403352,
  Tlimit=1000,
  allow={46400.06559999999,-993.868554,9.78494896,-0.00577437762,
         5.126164110000001e-006,-1.833418356e-009,1.92601341e-013},
  blow={-20227.21196,-27.40670981},
  ahigh={2545648.379,-7416.19598,13.81956204,-0.003096481934,7.72389421e-007,
         -7.921280320000001e-011,2.740829111e-015},
  bhigh={22313.02058,-61.4878113},
  R=79.46748909032763);
```

```

constant IdealGases.Common.DataRecord SrOHplus(
  name="SrOHplus",
  MM=0.1046267914,
  Hf=2964533.279188374,
  H0=106135.0429599431,
  Tlimit=1000,
  allow={38573.6694,-903.024233,9.451103249999999,-0.00504251628,
         4.190101890000001e-006,-1.196920603e-009,1.606318743e-014},
  blow={39950.67720000001,-26.0148611},
  ahigh={867969.566,-2340.609538,7.97682975,0.0001023752569,-6.28603233e-008,
         1.024851502e-011,-5.72368392e-016},
  bhigh={50731.5772,-20.27046917},
  R=79.46790577007029);

constant IdealGases.Common.DataRecord Sr_OH_2(
  name="Sr_OH_2",
  MM=0.12163468,
  Hf=-4905628.666100819,
  H0=139860.7781925352,
  Tlimit=1000,
  allow={72050.90300000001,-1789.465226,17.29429139,-0.01169280781,
         1.106702894e-005,-4.49014609e-009,6.603997460000001e-013},
  blow={-66053.94040000001,-63.88681219999999},
  ahigh={1750013.518,-4681.6015,13.95548402,0.000203501956,-1.253496115e-007,
         2.044549432e-011,-1.142004855e-015},
  bhigh={-44271.7024,-50.5145685},
  R=68.35609712624722);

constant IdealGases.Common.DataRecord SrS(
  name="SrS",
  MM=0.119685,
  Hf=871882.9845009816,
  H0=79676.77653841332,
  Tlimit=1000,
  allow={12564.40807,-315.0436559,5.72172251,-0.002630479718,3.38469975e-006,-
2.296256846e-009,
         6.512086680000001e-013},
  blow={12772.9516,-3.693874943},
  ahigh={-13794329.47,39213.3158,-36.0286638,0.01818513386,-3.27585816e-006,
         2.324312746e-010,-3.6145557e-015},
  bhigh={-241024.6179,299.1679642},
  R=69.46962443079751);

constant IdealGases.Common.DataRecord Sr2(
  name="Sr2",
  MM=0.17524,
  Hf=1755137.736818078,
  H0=64854.6964163433,
  Tlimit=1000,
  allow={-110885.9753,592.760401,8.43968855,-0.02652801112,4.43582597e-005,-
3.39991549e-008,
         9.96554667999999e-012},
  blow={31576.15033,-7.03396665},
  ahigh={209844.5682,103.865013,2.30933233,0.0001330517507,-4.42643339e-008,
         6.71141915e-012,-3.374128e-016},
  bhigh={36366.4269,21.68088517},
  R=47.44619949783154);

```

---

**1214 Modelica.Media.IdealGases.Common.SingleGasesData**

---

```
constant IdealGases.Common.DataRecord Ta (
  name="Ta",
  MM=0.1809479,
  Hf=4324552.194305654,
  H0=34262.00580388057,
  Tlimit=1000,
  allow={-11509.07339, 47.8073043, 3.18558839, -0.00536652816, 1.288379705e-005, -1.045798666e-008,
         3.050617695e-012},
  blow={92997.97630000001, 5.33605661},
  ahighelement1={1689726.898, -5986.85466, 9.565039670000001, -0.002511649459,
                 6.44303117e-007, -7.189237249999999e-011, 3.11335207e-015},
  bhighelement1={130671.0983, -43.3509627},
  R=45.94953575034582);

constant IdealGases.Common.DataRecord Taplus (
  name="Taplus",
  MM=0.1809473514,
  Hf=8564255.420209482,
  H0=35025.85117142533,
  Tlimit=1000,
  allow={286971.2865, -3084.920994, 14.30679704, -0.01984772164, 1.951445133e-005,
         -8.97094603e-009, 1.501974665e-012},
  blow={201382.8712, -63.0642783},
  ahighelement1={3656142.13, -12540.73524, 18.65022579, -0.007943274660000001,
                 2.151786937e-006, -2.816764844e-010, 1.413722944e-014},
  bhighelement1={263687.6455, -106.5864286},
  R=45.94967506111836);

constant IdealGases.Common.DataRecord Taminus (
  name="Taminus",
  MM=0.1809484486,
  Hf=4119789.004922146,
  H0=35458.32555980256,
  Tlimit=1000,
  allow={187398.2301, -1681.268679, 6.48004015, -0.0002254355782, 2.028434876e-006,
         -4.67980143e-009, 1.997027996e-012},
  blow={97934.9601999999, -20.49618155},
  ahighelement1={-4235467.48, 11010.56361, -4.73691107, 0.002503562129, -4.82185169e-007,
                 4.88640803e-011, -2.030547835e-015},
  bhighelement1={15746.99408, 64.918027},
  R=45.94939644041801);

constant IdealGases.Common.DataRecord TaCL5 (
  name="TaCL5",
  MM=0.3582129,
  Hf=-2135140.86176126,
  H0=75008.74479953123,
  Tlimit=1000,
  allow={63064.8329, -1771.221442, 22.55083356, -0.01360238041, 1.613009806e-005, -1.016307761e-008,
         2.631917453e-012},
  blow={-87927.3710999999, -80.97864282000001},
  ahighelement1={-142644.4545, -38.9434032, 16.0314237, -1.333676355e-005,
                 3.084307611e-009, -3.66393006e-013, 1.745841795e-017},
  bhighelement1={-97012.9717, -42.56986372},
  R=23.21097872243015);
```

```

constant IdealGases.Common.DataRecord TaO (
  name="TaO",
  MM=0.1969473,
  Hf=1231470.068388853,
  H0=44509.90188745923,
  Tlimit=1000,
  allow={-13957.38049, 394.523699, -0.0630169641, 0.01308595441, -1.86233973e-005,
    1.366155583e-008, -3.92455123e-012},
  blow={26451.95419, 27.41671469},
  ahighe={6106591.12, -19841.35246, 28.15797433, -0.01330045882, 3.87680749e-006, -
  5.06887276999999e-010,
    2.443666035e-014},
  bhigh={152581.4433, -165.8307106},
  R=42.21673513675993);

constant IdealGases.Common.DataRecord TaO2 (
  name="TaO2",
  MM=0.2129467,
  Hf=-815518.6532592428,
  H0=50243.08430231603,
  Tlimit=1000,
  allow={15163.56303, 70.58414759999999, 0.691851699, 0.02020605733, -2.928317413e-
  005,
    2.081417374e-008, -5.751106690000001e-012},
  blow={-22121.99468, 24.846625},
  ahighe={1297565.964, -4149.217430000001, 10.41680373, -0.001045478838,
    2.754904271e-007, -3.29704313e-011, 1.356249074e-015},
  bhigh={3418.98567, -33.9264279},
  R=39.04485019021192);

constant IdealGases.Common.DataRecord Ti (
  name="Ti",
  MM=0.047867,
  Hf=9881546.785885891,
  H0=157501.8488729187,
  Tlimit=1000,
  allow={-45701.79399999999, 660.809202, 0.429525749, 0.00361502991, -3.54979281e-
  006,
    1.759952494e-009, -3.052720871e-013},
  blow={52709.4793, 20.26149738},
  ahighe={-170478.6714, 1073.852803, 1.181955014, 0.0002245246352,
    3.091697848e-007, -5.74002728e-011, 2.927371014e-015},
  bhigh={49780.69910000001, 17.40431368},
  R=173.699458917417);

constant IdealGases.Common.DataRecord Tiplus (
  name="Tiplus",
  MM=0.0478664514,
  Hf=23766625.59531204,
  H0=165038.9525219745,
  Tlimit=1000,
  allow={170745.7044, -1727.524602, 9.615885329999999, -0.0108965506,
    8.20180965e-006, -2.871464413e-009, 3.420382976e-013},
  blow={144789.7558, -34.6314366},
  ahighe={-768546.308, 2545.8681, 0.342386278, 0.000709990136, 2.706231875e-008, -
  2.3716601e-011,
    1.895443077e-015},
  bhigh={119882.1489, 24.8479915},

```

---

**1216 Modelica.Media.IdealGases.Common.SingleGasesData**

---

```
R=173.7014496963525);

constant IdealGases.Common.DataRecord Timinus(
  name="Timinus",
  MM=0.0478675486,
  Hf=9593217.689865155,
  H0=157990.4177503671,
  Tlimit=1000,
  alow={-3006.48499,204.0911689,1.822638976,0.001245254812,-1.309239865e-006,
    7.37214322e-010,-1.724319779e-013},
  blow={53467.7205,12.05926588},
  ahigh={23411.1764,2.580413872,2.497754577,9.76907204e-007,-2.280024955e-010,
    2.717202291e-014,-1.295670294e-018},
  bhigh={54546.62179999999,7.99982395},
  R=173.6974681841134);

constant IdealGases.Common.DataRecord TiCL(
  name="TiCL",
  MM=0.0833199999999999,
  Hf=1810500.792126741,
  H0=116186.1017762842,
  Tlimit=1000,
  alow={-17141.77839,310.3259047,0.892093598,0.01383436793,-1.885370354e-005,
    1.212085912e-008,-3.032162427e-012},
  blow={15580.66453,22.44778384},
  ahigh={-963322.542,2868.829781,1.973820676,0.001752668011,-4.25543436e-007,
    5.0491034e-011,-2.304720715e-015},
  bhigh={-1811.011561,23.19450287},
  R=99.78963034085456);

constant IdealGases.Common.DataRecord TiCL2(
  name="TiCL2",
  MM=0.118773,
  Hf=-1997339.462672493,
  H0=114043.343184057,
  Tlimit=1000,
  alow={13243.54656,-576.282932,9.64004729,-0.00447512095,
    5.398794869999999e-006,-3.52580049e-009,9.74600455e-013},
  blow={-27920.66683,-22.18347625},
  ahigh={-3190012.95,10157.63826,-4.84079477,0.00682200634,-1.70075327e-006,
    2.118106588e-010,-1.041245457e-014},
  bhigh={-94809.82980000001,78.23195948},
  R=70.00304783073595);

constant IdealGases.Common.DataRecord TiCL3(
  name="TiCL3",
  MM=0.154226,
  Hf=-3496946.04022668,
  H0=99691.01837563056,
  Tlimit=1000,
  alow={124328.2268,-2320.728253,17.79895274,-0.01160889348,1.006643154e-005,
    -4.81194069e-009,1.006186828e-012},
  blow={-56096.0251,-67.32754065},
  ahigh={-94087.92310000002,-125.5351109,10.66923972,-0.0002637456897,
    7.95916935e-008,-1.176740054e-011,6.41622348e-016},
  bhigh={-67691.1468,-23.69164216},
  R=53.91096183522882);
```

```

constant IdealGases.Common.DataRecord TiCL4 (
  name="TiCL4",
  MM=0.189679,
  Hf=-4023429.056458543,
  H0=113947.9910796662,
  Tlimit=1000,
  allow={81871.96800000001,-1758.32385,18.92512121,-0.01133495876,
    1.251952037e-005,-7.42268132999999e-009,1.825440187e-012},
  blow={-86729.2310999999,-67.69594291},
  ahighe={-143256.2278,-43.1677485,13.03337752,-1.371365738e-005,
    3.093403483e-009,-3.60424516e-013,1.691384523e-017},
  bhigh={-95889.4699999999,-32.47541011},
  R=43.83443607357694);

constant IdealGases.Common.DataRecord TiO(
  name="TiO",
  MM=0.0638664,
  Hf=775112.0307391679,
  H0=150205.9768516779,
  Tlimit=1000,
  allow={-11681.5246,454.256565,-0.1139144613,0.01275432333,-1.727656935e-005,
    1.187369403e-008,-3.23657937e-012},
  blow={2924.306353,27.02903947},
  ahighe={2330644.03,-7415.79386,12.81799311,-0.004344555950000001,
    1.186303111e-006,-1.367644275e-010,5.70321225e-015},
  bhigh={51448.4136,-57.9399424},
  R=130.1853869953528);

constant IdealGases.Common.DataRecord TiOplus(
  name="TiOplus",
  MM=0.0638658513999999,
  Hf=10730632.97485454,
  H0=144257.2329036547,
  Tlimit=1000,
  allow={36912.5625,-149.2825538,2.624977257,0.00581862713,-7.52966211e-006,
    4.74415435e-009,-1.186916989e-012},
  blow={82415.5116999999,10.95428726},
  ahighe={342132.953,-2161.85106,8.02517566,-0.002708700692,1.004583805e-006,-
    1.50576685e-010,
    8.04565811e-015},
  bhigh={93626.4699,-22.61587887},
  R=130.1865052722056);

constant IdealGases.Common.DataRecord TiOCL(
  name="TiOCL",
  MM=0.0993194,
  Hf=-2459358.393224285,
  H0=122259.1054718413,
  Tlimit=1000,
  allow={35458.5651,-629.528714,7.3701763,0.00330201914,-6.12145875e-006,
    4.73588025e-009,-1.374690357e-012},
  blow={-27970.87903,-12.94470159},
  ahighe={-127305.9873,-111.111885,7.58288808,-3.3128718e-005,7.31401409e-009,
    -8.37882226e-013,3.87927506e-017},
  bhigh={-31393.56857,-12.47104274},
  R=83.71448075602551);

```

---

**1218 Modelica.Media.IdealGases.Common.SingleGasesData**

---

```
constant IdealGases.Common.DataRecord TiOCL2(
  name="TiOCL2",
  MM=0.1347724,
  Hf=-4047950.470571125,
  H0=123986.8845549979,
  Tlimit=1000,
  allow={32890.5246,-738.918859,10.55976857,0.001577742341,-3.9071821e-006,
         3.29057653e-009,-9.949471750000001e-013},
  blow={-64484.107,-24.17469934},
  ahighe={-132295.1633,-96.91212050000002,10.07240769,-2.897248531e-005,
          6.40177504e-009,-7.3384526e-013,3.39928267e-017},
  bhigh={-68474.09640000001,-19.75043469},
  R=61.69269078832164);

constant IdealGases.Common.DataRecord TiO2(
  name="TiO2",
  MM=0.0798657999999999,
  Hf=-3824290.246889157,
  H0=142134.5557172156,
  Tlimit=1000,
  allow={-1710.545601,272.1435528,0.596137896,0.01925463599,-2.665500165e-005,
         1.811109197e-008,-4.87671047e-012},
  blow={-39122.4177,24.08605889},
  ahighe={154629.9764,-1046.25688,7.78898583,-0.0001546805714,-7.05993595e-008,
          3.100244802e-011,-2.49472543e-015},
  bhigh={-32663.3675,-15.9153466},
  R=104.1055370383819);

constant IdealGases.Common.DataRecord U(
  name="U",
  MM=0.23802891,
  Hf=2247626.139194605,
  H0=27304.64127235637,
  Tlimit=1000,
  allow={69657.3775,-1070.351517,8.075842310000001,-0.01060034069,
         9.25654801e-006,-3.21989976e-009,4.058048809e-013},
  blow={68665.137,-22.40521678},
  ahighe={-4092498.96,12748.88349,-12.18707506,0.00725810568,-7.78777507e-007,
          -3.84435385e-011,7.066508567e-015},
  bhigh={-16993.72664,115.5026301},
  R=34.93051327252644);

constant IdealGases.Common.DataRecord UF(
  name="UF",
  MM=0.2570273132,
  Hf=-191616.7444884609,
  H0=36521.65555143032,
  Tlimit=1000,
  allow={172553.3463,-1698.985561,5.90292643,0.01841977309,-4.633174310000001e-005,
         4.37665720000001e-008,-1.451876566e-011},
  blow={2086.455917,-11.86592944},
  ahighe={-5325439.37,15339.67086,-11.75044398,0.00964721093,-2.49877819e-006,
          3.155572896e-010,-1.544694839e-014},
  bhigh={-105719.6525,122.0828149},
  R=32.34859321557893);
```

```

constant IdealGases.Common.DataRecord UFplus(
  name="UFplus",
  MM=0.2570267646,
  Hf=2167318.908857323,
  H0=36958.66854482439,
  Tlimit=1000,
  allow={1622640.597,-19173.26611,87.5946122,-0.1699915707,0.0001861251683,-
1.036176702e-007,
         2.324287412e-011},
  blow={161670.9643,-480.6891249},
  ahight={539509.184,-2923.962095,10.08511948,-0.002425945123,5.92872548e-007,
         -6.79584554e-011,3.152666445e-015},
  bhigh={82672.91220000001,-33.07089704},
  R=32.34866226067727);

constant IdealGases.Common.DataRecord UFminus (
  name="UFminus",
  MM=0.2570278618,
  Hf=-605689.9509250013,
  H0=35622.62447300179,
  Tlimit=1000,
  allow={3692.32786,-149.1979606,4.03398852,0.002857429445,-5.17170256e-006,
         3.98537911e-009,-9.47035544e-013},
  blow={-19152.81695,6.059243838},
  ahight={-4311143.56,19748.17938,-27.60304763,0.02276806261,-6.88874172e-006,
         9.483511059999999e-010,-4.9106881e-014},
  bhigh={-138080.129,224.0758537},
  R=32.34852417077533);

constant IdealGases.Common.DataRecord UF2 (
  name="UF2",
  MM=0.2760257164,
  Hf=-1938358.21523476,
  H0=54894.85616638001,
  Tlimit=1000,
  allow={-38824.9202,445.493086,4.71800919,0.00164891148,
         8.696243010000001e-006,-1.207439921e-008,4.53679391e-012},
  blow={-68553.29330000001,11.59725423},
  ahight={-471677.682,322.423686,8.21409062999999,-0.000140741378,
         6.54117805e-009,1.818915497e-012,-2.287963719e-016},
  bhigh={-69952.3505,-9.648963156000001},
  R=30.12209191389676);

constant IdealGases.Common.DataRecord UF2plus (
  name="UF2plus",
  MM=0.2760251678,
  Hf=255214.1442806506,
  H0=52303.66170979283,
  Tlimit=1000,
  allow={5439.01016,-150.7096256,6.87348519,0.001530973969,-3.31669431e-006,
         2.949453335e-009,-8.28170654e-013},
  blow={7256.01491,-2.205491833},
  ahight={-3779149.61,12987.37504,-10.27930641,0.01048838051,-2.651047812e-006,
         3.116973593e-010,-1.417216291e-014},
  bhigh={-74334.402,118.9131079},
  R=30.12215178155214);

```

---

**1220 Modelica.Media.IdealGases.Common.SingleGasesData**

---

```
constant IdealGases.Common.DataRecord UF2minus(
  name="UF2minus",
  MM=0.2760262650000001,
  Hf=-2457132.820313313,
  H0=47652.49060628342,
  Tlimit=1000,
  allow={806049.1730000001,-9404.254659999999,44.4839277,-0.0686855877,
         6.777809920000001e-005,-3.33882768e-008,6.60485162e-012},
  blow={-36033.0613,-226.1594249},
  ahighelement1={10382318.55,-28401.13518,34.9853337,-0.01053216396,1.92462047e-006,-
1.69874957e-010,
               5.70985889e-015},
  bhigh={100157.2794,-209.4039075},
  R=30.12203204647934);

constant IdealGases.Common.DataRecord UF3(
  name="UF3",
  MM=0.2950241196,
  Hf=-3596175.792130048,
  H0=63191.657093246,
  Tlimit=1000,
  allow={30710.27207,-634.078947,11.14030799,-0.0002937554125,-1.865390371e-
006,
         2.37181947e-009,-7.44675407e-013},
  blow={-127183.8141,-23.48962529},
  ahighelement1={-3828876.42,12968.91783,-7.26555429,0.01048289752,-2.649840687e-006,
               3.115594509e-010,-1.4165794e-014},
  bhigh={-211361.5913,105.4464687},
  R=28.18234662058458);

constant IdealGases.Common.DataRecord UF3plus(
  name="UF3plus",
  MM=0.295023571,
  Hf=-965161.6277127904,
  H0=63245.35336873134,
  Tlimit=1000,
  allow={-398415.589,3480.89244,-2.519603507,0.02283749282,-2.16884478e-005,
         1.1126579e-008,-2.350882012e-012},
  blow={-55508.892,58.53815975},
  ahighelement1={-1196316.388,3972.78507,4.53590711,0.00413341615,-1.097960689e-006,
               1.319641252e-010,-6.10112918e-015},
  bhigh={-61995.4174,21.37623045},
  R=28.18239902600868);

constant IdealGases.Common.DataRecord UF3minus(
  name="UF3minus",
  MM=0.2950246682,
  Hf=-4021496.840378447,
  H0=65615.96736336911,
  Tlimit=1000,
  allow={-231518.73,2534.197822,-3.18071459,0.02932741422,-2.85691226e-005,
         1.345804152e-008,-2.465442912e-012},
  blow={-158038.7746,59.73605435},
  ahighelement1={-191005.3149,-506.918131,11.81482258,-0.0001711879404,-9.38483918e-
008,
               2.242009537e-011,-1.42819528e-015},
  bhigh={-143862.8681,-27.74680411},
  R=28.18229421535538);
```

```

constant IdealGases.Common.DataRecord UF4 (
  name="UF4",
  MM=0.3140225228,
  Hf=-5114784.136114202,
  H0=69716.59804778818,
  Tlimit=1000,
  allow={-50078.5238,-969.349355,20.69596355,-0.02564852637,4.19586012e-005,-
3.20101339e-008,
         9.422076860000001e-012},
  blow={-193162.6334,-71.72362935},
  ahigh={-1230291.173,3876.46379,7.62162591,0.004094525220000001,-
1.088530094e-006,
        1.308042117e-010,-6.04439087e-015},
  bhigh={-221414.292,6.270300658},
  R=26.47731100897964);

constant IdealGases.Common.DataRecord UF4plus (
  name="UF4plus",
  MM=0.3140219742,
  Hf=-2042976.405184323,
  H0=64687.5622374837,
  Tlimit=1000,
  allow={38489.5633,-1343.282151,16.46334809,-0.00719607986,1.147165041e-005,-
8.70587798e-009,
         2.501076195e-012},
  blow={-74050.482,-52.95285534999999},
  ahigh={-1617630.159,3765.93477,9.16613853,0.002426641067,-
5.695320299999999e-007,
        6.021096599999999e-011,-2.438874329e-015},
  bhigh={-106191.0645,-4.162641042},
  R=26.47735726514645);

constant IdealGases.Common.DataRecord UF4minus (
  name="UF4minus",
  MM=0.3140230714,
  Hf=-5503846.766081914,
  H0=69666.13918674002,
  Tlimit=1000,
  allow={72031.7969,-1374.343862,16.40877729,-0.00436306627,2.460407959e-006,-
1.498144731e-010,
         -1.239417075e-013},
  blow={-204517.4023,-51.72250795},
  ahigh={-4639200.11,16268.24426,-9.59529105,0.01441556535,-
4.008173850000001e-006,
        5.140230790000001e-010,-2.511687821e-014},
  bhigh={-312514.0933,127.4038784},
  R=26.47726475297446);

constant IdealGases.Common.DataRecord UF5 (
  name="UF5",
  MM=0.333020926,
  Hf=-5854959.048429288,
  H0=70875.07167642673,
  Tlimit=1000,
  allow={161576.9193,-2976.611537,25.3754242,-0.0193010567,2.561852107e-005,-
1.747402896e-008,
         4.73205938e-012},
  blow={-223908.9616,-102.4264836},

```

---

**1222 Modelica.Media.IdealGases.Common.SingleGasesData**

---

```
ahigh={-1695373.88,3735.42535,12.1874804,0.002418608203,-5.67852616e-007,
       6.00276561e-011,-2.430740809e-015},
bhigh={-264510.7342,-18.83137475},
R=24.96681544870847);

constant IdealGases.Common.DataRecord UF5plus(
  name="UF5plus",
  MM=0.3330203774,
  Hf=-2563258.286668436,
  H0=70903.04258360378,
  Tlimit=1000,
  allow={162651.9713,-2915.370389,24.36168077,-0.01382199656,1.339849041e-005,
         -7.07857269e-009,1.57340404e-012},
  blow={-92264.0485,-98.30243935},
  ahight={-262847.8263,-117.4047323,16.08988291,-3.66551671e-005,
          8.221494970000001e-009,-9.53716052e-013,4.46014508e-017},
  bhigh={-107629.3931,-47.63374165000001},
  R=24.96685657770803);

constant IdealGases.Common.DataRecord UF5minus(
  name="UF5minus",
  MM=0.3330214746,
  Hf=-6874725.540597317,
  H0=75649.66502613644,
  Tlimit=1000,
  allow={-314213.0934,2020.611008,7.50849182,0.01640034396,-1.562795472e-005,
         7.99024601e-009,-1.671674534e-012},
  blow={-290764.7496,4.690020575},
  ahight={-1331518.187,3905.51686,10.58727667,0.00411250727,-1.09327793e-006,
          1.314215436e-010,-6.07577827e-015},
  bhigh={-304950.6955,-7.977511665},
  R=24.96677431984442);

constant IdealGases.Common.DataRecord UF6(
  name="UF6",
  MM=0.3520193292,
  Hf=-6103760.418165129,
  H0=75630.36967459797,
  Tlimit=1000,
  allow={191567.406,-3426.39161,28.0039589,-0.01326392086,1.107123764e-005,-
4.82175410999999e-009,
         8.32513245e-013},
  blow={-246104.5194,-121.1612947},
  ahight={-340902.792,-145.8950046,19.10902533,-4.358451129999999e-005,
          9.617604659999999e-009,-1.100915551e-012,5.09264508e-017},
  bhigh={-264364.5561,-65.79130395999999},
  R=23.61936209268818);

constant IdealGases.Common.DataRecord UF6minus(
  name="UF6minus",
  MM=0.3520198778,
  Hf=-7645324.368668365,
  H0=78775.65657174375,
  Tlimit=1000,
  allow={156923.0858,-3027.27994,27.01442427,-0.0136860992,8.29857616e-006,-
2.582889593e-009,
         2.790817248e-013},
```

```

blow={-313530.7376,-112.8934794},
ahigh={2699676.848,-6574.45967,22.41731997,0.000605779971,-4.51749171e-007,
    7.82182046999999e-011,-4.56033472e-015},
bhigh={-285566.7984,-91.82896336},
R=23.61932528345421);

constant IdealGases.Common.DataRecord UO (
  name="UO",
  MM=0.25402831,
  Hf=120020.4378795419,
  H0=37548.13390680748,
  Tlimit=1000,
  allow={1007249.615,-12871.90666,59.93123920000001,-0.1003445164,
    8.529345550000001e-005,-2.634105066e-008,-5.350649790000001e-013},
  blow={66274.4252,-322.6786779},
  ahigh={-2458660.003,3942.65216,5.01603072,-0.0005450639469999999,
    2.096321351e-007,-2.654652526e-011,1.361108472e-015},
  bhigh={-26651.49645,5.580598689},
  R=32.73049369969828);

constant IdealGases.Common.DataRecord UOplus (
  name="UOplus",
  MM=0.2540277614,
  Hf=2287041.084793814,
  H0=34737.25450859325,
  Tlimit=1000,
  allow={15628.34007,-122.4921454,3.32414616,0.002467496097,-
6.85185673999999e-007,
    5.55207319e-011,-5.89301203e-014},
  blow={69529.99740000001,9.583869879},
  ahigh={-106371.4748,-1793.716133,8.02444904999999,-0.001565116442,
    5.06603391e-007,-7.37397747e-011,4.07476643e-015},
  bhigh={77890.8584,-21.1528425},
  R=32.73056438468461);

constant IdealGases.Common.DataRecord UOF (
  name="UOF",
  MM=0.2730267132,
  Hf=-1985823.063411511,
  H0=51297.01718871954,
  Tlimit=1000,
  allow={386.124847,51.62286899999999,4.88896252,0.00726563356,-1.134176875e-
005,
    8.50904175e-009,-2.354655485e-012},
  blow={-67198.1338,8.269521875000001},
  ahigh={-3805936.01,12952.07909,-10.25373631,0.01047841671,-2.648894949e-006,
    3.114551623e-010,-1.416111895e-014},
  bhigh={-147897.6701,118.4570601},
  R=30.45296155292104);

constant IdealGases.Common.DataRecord UOF2 (
  name="UOF2",
  MM=0.2920251164,
  Hf=-3819912.060139495,
  H0=61009.94400630945,
  Tlimit=1000,
  allow={-91066.1831,-101.4150169,13.71057013,-0.01659931694,3.082352584e-005,
    1.111111111111111e-005,-1.111111111111111e-005});

```

---

**1224 Modelica.Media.IdealGases.Common.SingleGasesData**

---

```
-2.490530206e-008, 7.580617639999999e-012},  
blow={-137468.6016, -34.70029665},  
ahigh={-1198944.06, 3852.17341, 4.63910975, 0.00408773844, -1.087067018e-006,  
1.306398384e-010, -6.03690152e-015},  
bhigh={-161263.7583, 20.18237189},  
R=28.47177017682031);  
  
constant IdealGases.Common.DataRecord UOF3 (  
  name="UOF3",  
  MM=0.3110235196,  
  Hf=-4856989.72522334,  
  H0=63748.10826364272,  
  Tlimit=1000,  
  allow={124925.6374, -2111.719498, 18.44931536, -0.01041493583, 1.469002962e-005,  
-1.049770051e-008, 2.922353602e-012},  
  blow={-174384.4024, -66.49285435},  
  ahigh={-1658296.049, 3714.43468, 9.20242554, 0.002412862293, -5.66624172e-007,  
5.98906211e-011, -2.42453481e-015},  
  bhigh={-210549.1714, -5.670532802},  
  R=26.73261498260018);  
  
constant IdealGases.Common.DataRecord UOF4 (  
  name="UOF4",  
  MM=0.3300219228,  
  Hf=-5410584.593442651,  
  H0=68912.79769248105,  
  Tlimit=1000,  
  allow={143716.4405, -2496.730363, 21.04827038, -0.00469256133,  
8.741718649999999e-007, 1.517555623e-009, -7.76879974e-013},  
  blow={-206128.4178, -82.46219615},  
  ahigh={-299619.0288, -167.7276501, 16.12494871, -4.98441446e-005,  
1.098253497e-008, -1.255803231e-012, 5.80448098e-017},  
  bhigh={-219545.926, -50.24484645},  
  R=25.1936960110336);  
  
constant IdealGases.Common.DataRecord UO2 (  
  name="UO2",  
  MM=0.27002771,  
  Hf=-1769521.73908374,  
  H0=49847.09532218007,  
  Tlimit=1000,  
  allow={-112965.0727, 427.073027, 8.41369401, -0.00976428, 2.199903691e-005, -  
1.907665954e-008,  
6.02988991e-012},  
  blow={-62514.433, -13.00704863},  
  ahigh={-1190542.635, 3832.18635, 2.153236312, 0.00408235191, -1.085924847e-006,  
1.305134065e-010, -6.03121575e-015},  
  bhigh={-83676.1801, 25.90742388},  
  R=30.79118065327443);  
  
constant IdealGases.Common.DataRecord UO2plus (  
  name="UO2plus",  
  MM=0.2700271614,  
  Hf=190699.0346194115,  
  H0=42871.68350020659,  
  Tlimit=1000,  
  allow={44880.2338, -648.923642, 6.87424356, 0.002281949316, 1.859218752e-007, -
```

```

1.849509036e-009,
  8.14996533e-013},
blow={7891.74023,-10.28257233},
ahigh={-1589957.42,3657.74607,3.74678542,0.002394401448,-5.62410144e-007,
  5.93945164e-011,-2.401047056e-015},
bhigh={-20485.58817,14.64015614},
R=30.79124321009879);

constant IdealGases.Common.DataRecord UO2minus(
  name="UO2minus",
  MM=0.2700282586,
  Hf=-2124592.529590901,
  H0=51021.98959261074,
  Tlimit=1000,
  allow={63419.7131,-719.61185,8.30824162,0.000962645545,-3.58373398e-006,
    3.47355458e-009,-1.02279312e-012},
  blow={-67181.69869999999,-16.0603012},
  ahigh={-273825.164,3228.53442,-0.2805114576,0.006607310710000001,-
2.070291015e-006,
  2.851708013e-010,-1.466578192e-014},
  bhigh={-88934.3618,41.45760316},
  R=30.79111809670427);

constant IdealGases.Common.DataRecord UO2F(
  name="UO2F",
  MM=0.2890261132,
  Hf=-3452749.75659189,
  H0=54679.34307051395,
  Tlimit=1000,
  allow={83217.92390000001,-1327.082127,12.4012539,-0.00408415101,
    7.33159348e-006,-5.98740488e-009,1.787869209e-012},
  blow={-115752.6888,-34.15298625},
  ahigh={-1614277.202,3712.77186,6.20330882,0.002412618055,-
5.66588019999999e-007,
  5.98882352e-011,-2.424485602e-015},
  bhigh={-147835.7384,8.736139132},
  R=28.76719998738163);

constant IdealGases.Common.DataRecord UO2F2(
  name="UO2F2",
  MM=0.3080245164,
  Hf=-4396507.2742502,
  H0=61904.07576271744,
  Tlimit=1000,
  allow={99598.74299999999,-1530.010701,13.58992177,0.0053706558,-1.141444341e-
005,
  9.301197039999999e-009,-2.781220538e-012},
  blow={-158031.0734,-41.96260275},
  ahigh={-263655.2036,-203.2835065,13.15092994,-6.00648683e-005,
    1.321137913e-008,-1.508668985e-012,6.96618275e-017},
  bhigh={-166444.4818,-35.66258175},
  R=26.99289036202185);

constant IdealGases.Common.DataRecord UO3(
  name="UO3",
  MM=0.28602711,
  Hf=-2794278.524857312,

```

---

**1226 Modelica.Media.IdealGases.Common.SingleGasesData**

---

```
H0=52948.63483394984,
Tlimit=1000,
alow={66376.7703999999,-758.264657999999,7.11284471,0.01322149697,-
2.106191042e-005,
    1.545856318e-008,-4.37856629e-012},
blow={-94133.6934999999,-8.588030428},
ahigh={-1097362.721,2808.784061,5.96612147,0.002861871152,-1.05284381e-006,
    1.849985929e-010,-1.102849619e-014},
bhigh={-117336.0017,6.672828312},
R=29.06882497956226);

constant IdealGases.Common.DataRecord UO3minus(
  name="UO3minus",
  MM=0.2860276586,
  Hf=-4563036.051786915,
  H0=53935.6301258063,
  Tlimit=1000,
  alow={100014.3902,-1448.443754,12.17532415,-0.002944956748,
      5.67522799999999e-006,-4.91671042e-009,1.525185713e-012},
  blow={-151925.3376,-34.71731375},
  ahigh={-1632938.634,3667.49541,6.24003065,0.002396952522,-5.62949023e-007,
      5.94541313e-011,-2.403730982e-015},
  bhigh={-184592.7385,7.032059012},
  R=29.06876922566257);

constant IdealGases.Common.DataRecord V(
  name="V",
  MM=0.0509415,
  Hf=10154138.84553851,
  H0=155218.5153558494,
  Tlimit=1000,
  alow={-55353.7602,559.333851,2.675543482,-0.00624304963,1.565902337e-005,-
1.372845314e-008,
    4.16838881e-012},
  blow={58206.6436,9.524567490000001},
  ahigh={1200390.3,-5027.0053,10.58830594,-0.005044326100000001,
    1.488547375e-006,-1.785922508e-010,8.113013866e-015},
  bhigh={91707.4090999999,-47.6833632},
  R=163.2160811911703);

constant IdealGases.Common.DataRecord Vplus(
  name="Vplus",
  MM=0.0509409514,
  Hf=23041293.47297585,
  H0=155038.5452753833,
  Tlimit=1000,
  alow={75688.3446,-841.527382,7.55923271,-0.01441722656,2.038356397e-005,-
1.289073883e-008,
    3.065656561e-012},
  blow={144447.8191,-19.91067645},
  ahigh={2347072.054,-9021.197190000001,14.77349798,-0.00689189688,
    1.968884877e-006,-2.539798544e-010,1.226783122e-014},
  bhigh={195835.1444,-78.5559293},
  R=163.2178389192786);

constant IdealGases.Common.DataRecord Vminus(
  name="Vminus",
```

```

MM=0.0509420486,
Hf=9037446.974600077,
H0=154651.4758733122,
Tlimit=1000,
alow={-3799.27356, 231.3840448, 1.72560819, 0.001429275357, -1.506038188e-006,
      8.491815170000001e-010, -1.987980413e-013},
blow={53474.0292, 12.61900982},
ahigh={26001.0043, 2.096334097, 2.498006548, 8.991278839999999e-007, -
2.139749508e-010,
      2.581021334e-014, -1.240812796e-018},
bhigh={54700.0708, 7.97790024},
R=163.2143235009202);

constant IdealGases.Common.DataRecord VCL4 (
  name="VCL4",
  MM=0.1927535,
  Hf=-2734365.290383832,
  H0=113010.129517752,
  Tlimit=1000,
  alow={77198.3471, -1702.85404, 18.82697965, -0.01130896832, 1.266950765e-005, -
7.625019339999999e-009,
      1.907582465e-012},
  blow={-58637.4854, -65.59171105999999},
  ahigh={-1717776.251, 4550.441049999999, 8.164464799999999, 0.002224875998, -
4.094111780000001e-007,
      3.2717875e-011, -8.8586908e-016},
  bhigh={-96906.20060000001, 4.348293098},
  R=43.13525824433798);

constant IdealGases.Common.DataRecord VN (
  name="VN",
  MM=0.0649482,
  Hf=8052571.125912651,
  H0=135488.6509556847,
  Tlimit=1000,
  alow={-15817.37285, 486.816542, -1.045200388, 0.01685261043, -2.334616543e-005,
      1.58871005e-008, -4.27908088e-012},
  blow={59814.814, 31.44995263},
  ahigh={1018619.667, -3932.30557, 9.856823609999999, -0.00328922171,
      1.005181767e-006, -1.243211436e-010, 5.48677656e-015},
  bhigh={85573.18339999999, -35.52830762},
  R=128.0169735265951);

constant IdealGases.Common.DataRecord VO (
  name="VO",
  MM=0.0669409,
  Hf=2219610.223346266,
  H0=131057.4850353073,
  Tlimit=1000,
  alow={-13116.19784, 374.781697, 0.0930083486, 0.01244977714, -1.688540028e-005,
      1.142443381e-008, -3.04058932e-012},
  blow={15237.8992, 25.36811755},
  ahigh={2986190.283, -10113.44974, 17.18161749, -0.00787670503, 2.562279547e-006,
      -3.54740035e-010, 1.770268056e-014},
  bhigh={79612.5488, -87.89993010000001},
  R=124.2061579691937);

```

---

**1228 Modelica.Media.IdealGases.Common.SingleGasesData**

---

```
constant IdealGases.Common.DataRecord VO2 (
  name="VO2",
  MM=0.0829403,
  Hf=-2805604.211704081,
  H0=128073.4335419573,
  Tlimit=1000,
  allow={-6678.58586, 391.159758, -1.028549847, 0.02401523419, -3.33737881e-005,
         2.283623543e-008, -6.1991431e-012},
  blow={-30746.09276, 32.7791238},
  ahighelement1={121063.2401, -1627.832993, 9.252713099999999, -0.001572703139,
                 5.23143016e-007, -6.476071140000001e-011, 2.847226026e-015},
  bhighelement1={-20973.06345, -25.47380687},
  R=100.2464664342907);

constant IdealGases.Common.DataRecord V4010 (
  name="V4010",
  MM=0.36376,
  Hf=-7766561.705520123,
  H0=101154.6981526281,
  Tlimit=1000,
  allow={338573.916, -5353.92926, 29.93220131, 0.0581883652, -9.69634016e-005,
         7.27475699e-008, -2.091628357e-011},
  blow={-318934.937, -146.6398396},
  ahighelement1={-1360273.27, -1341.69286, 40.9913859, -0.000393226553, 8.62849412e-008,
                 9.83594409e-012,
                 4.53563729e-016},
  bhighelement1={-348461.612, -190.7911348},
  R=22.85702661095228);

constant IdealGases.Common.DataRecord W (
  name="W",
  MM=0.18384,
  Hf=4630349.902088773,
  H0=33814.87162750217,
  Tlimit=1000,
  allow={159522.3922, -2673.843928, 20.60469727, -0.0625231523, 0.0001105654838,
         -8.45351161e-008,
         2.336187771e-011},
  blow={113964.8616, -90.118369},
  ahighelement1={-8048745.96, 14657.00424, -0.2508531501, -0.002596486992,
                 1.409225475e-006, -2.233011706e-010, 1.262640862e-014},
  bhighelement1={-3091.130919, 39.5582219},
  R=45.22667536988686);

constant IdealGases.Common.DataRecord Wplus (
  name="Wplus",
  MM=0.1838394514,
  Hf=8854687.895353457,
  H0=33840.75046255279,
  Tlimit=1000,
  allow={-196928.4929, 2670.137332, -11.31686913, 0.0330818373, -3.6290355e-005,
         2.066142971e-008, -4.808285562e-012},
  blow={182095.0862, 85.5210448},
  ahighelement1={6387743.399999999, -20618.11463, 27.59291576, -0.01244535845,
                 3.27120049e-006, -4.065463720000001e-010, 1.912595872e-014},
  bhighelement1={324517.443, -171.6919194},
  R=45.22681033196339);
```

```

constant IdealGases.Common.DataRecord Wminus (
  name="Wminus",
  MM=0.1838405486,
  Hf=4168783.948026143,
  H0=33710.8872182728,
  Tlimit=1000,
  allow={0,0,2.5,0,0,0,0},
  blow={91429.8137,8.46116967},
  ahight={0,0,2.5,0,0,0,0},
  bhigh={91429.8137,8.46116967},
  R=45.22654040861582);

constant IdealGases.Common.DataRecord WCL6 (
  name="WCL6",
  MM=0.396558,
  Hf=-1244993.166195109,
  H0=77645.5272620903,
  Tlimit=1000,
  allow={33393.9167,-1697.36608,25.60612352,-0.01425247881,1.739682668e-005,-
1.120537864e-008,
  2.95152289e-012},
  blow={-56730.59239999999,-97.41466299000001},
  ahight={-151673.5473,-34.5137925,19.02867176,-1.242486969e-005,
  2.917675103e-009,-3.50598144e-013,1.685355887e-017},
  bhigh={-65357.79790000001,-58.94914959000001},
  R=20.96659757210799);

constant IdealGases.Common.DataRecord WO (
  name="WO",
  MM=0.1998394,
  Hf=2010292.820134568,
  H0=46332.76020644578,
  Tlimit=1000,
  allow={-19337.58411,493.669084,-0.411614822000001,0.01307976507,-
1.689145619e-005,
  1.092748066e-008,-2.820593541e-012},
  blow={45110.179,30.02592661},
  ahight={1262156.956,-4177.263120000001,9.35828647,-0.00288761222,
  8.89393396e-007,-8.955318699999999e-011,2.504359614e-015},
  bhigh={73133.8064,-31.4488229},
  R=41.60576943285459);

constant IdealGases.Common.DataRecord WOCL4 (
  name="WOCL4",
  MM=0.3416514000000001,
  Hf=-1678591.101924359,
  H0=65187.43666790184,
  Tlimit=1000,
  allow={26588.12693,-933.6913239999999,13.21377385,0.01270930811,-
1.964716509e-005,
  1.405106974e-008,-3.89938214e-012},
  blow={-67922.90790000001,-35.94869683},
  ahight={-304902.8285,-353.942586,16.26432703,-0.0001058245461,
  2.340548846e-008,-2.685852021e-012,1.245412684e-016},
  bhigh={-72725.46739999999,-49.91761253},
  R=24.33612740940034);

```

---

**1230 Modelica.Media.IdealGases.Common.SingleGasesData**

---

```
constant IdealGases.Common.DataRecord WO2 (
  name="WO2",
  MM=0.2158388,
  Hf=134645.5966211821,
  H0=49642.82603498537,
  Tlimit=1000,
  allow={3120.918919,241.3883468,-0.3024119184,0.02324333417,-3.37881268e-005,
         2.380934019e-008,-6.421592880000001e-012},
  blow={1442.005265,29.57655179},
  ahight=-753740.6680000001,3204.64305,0.6701965600000001,0.00448823887,-
8.85832845999999e-007,
  6.24517175e-011,-9.046613900000001e-016},
  bhigh={-17694.2023,34.0572401},
  R=38.52167450893908);

constant IdealGases.Common.DataRecord WO2CL2 (
  name="WO2CL2",
  MM=0.2867448,
  Hf=-2341915.180327594,
  H0=68018.03206195892,
  Tlimit=1000,
  allow={430.015448,-243.8547856,7.97630699,0.01715005722,-2.487975176e-005,
         1.737644163e-008,-4.7719284e-012},
  blow={-82328.124,-7.844771421},
  ahight=-240479.173,-306.8379644,13.22814502,-9.10022071e-005,
  2.00658102e-008,-2.296837607e-012,1.062829386e-016},
  bhigh={-83674.9305999999,-34.91556626},
  R=28.99606897840868);

constant IdealGases.Common.DataRecord WO3 (
  name="WO3",
  MM=0.2318382,
  Hf=-1379087.712896321,
  H0=57229.32200129228,
  Tlimit=1000,
  allow={7262.46146,34.2939109,1.573061955,0.02754971099,-3.99482731e-005,
         2.809371537e-008,-7.77754573e-012},
  blow={-40017.3327,18.57409963},
  ahight=1732203.64,-6284.719779999999,16.81358864,-0.00347208936,
  8.079935799999999e-007,-6.589378720000001e-011,1.142928957e-015},
  bhigh={-2473.075565,-74.0747409},
  R=35.86325290655294);

constant IdealGases.Common.DataRecord WO3minus (
  name="WO3minus",
  MM=0.2318387486,
  Hf=-2805725.811271913,
  H0=59155.9395606572,
  Tlimit=1000,
  allow={163099.9684,-1967.609682,12.01970914,0.001681260779,-
5.859360060000001e-006,
         5.02861633e-009,-1.434012198e-012},
  blow={-70092.03989999999,-39.3690104},
  ahight=463783.251,-1347.089334,9.773637559999999,0.001071395459,-
3.84240366e-007,
  5.7375319e-011,-3.157605982e-015},
  bhigh={-72395.1557,-23.91116343},
  R=35.86316804334235);
```

```

constant IdealGases.Common.DataRecord Xe (
  name="Xe",
  MM=0.131293,
  Hf=0,
  H0=47203.03443443291,
  Tlimit=1000,
  alow={0,0,2.5,0,0,0,0},
  blow={-745.375,6.164454205},
  ahighe={4025.22668,-12.09507521,2.514153347,-8.248102080000001e-006,
    2.530232618e-009,-3.89233323e-013,2.360439138e-017},
  bhigh={-668.5800730000001,6.063710715},
  R=63.3276107637117);

constant IdealGases.Common.DataRecord Xeplus (
  name="Xeplus",
  MM=0.1312924514,
  Hf=8961309.042935578,
  H0=47203.23167033196,
  Tlimit=1000,
  alow={100.292362,-1.218753648,2.506016493,-1.547411334e-005,
    2.191372741e-008,-1.623684074e-011,4.929132670000001e-015},
  blow={140766.5368,7.516712465},
  ahighe={-12416.83887,-150.0654643,2.964678293,-0.000469339666,
    1.959138719e-007,-3.037761925e-011,1.637361082e-015},
  bhigh={141496.6808,4.565685735},
  R=63.32787537547646);

constant IdealGases.Common.DataRecord Zn (
  name="Zn",
  MM=0.06539,
  Hf=1994188.713870623,
  H0=94776.38782688484,
  Tlimit=1000,
  alow={0,0,2.5,0,0,0,0},
  blow={14938.05072,5.11886101},
  ahighe={-175559.1489,498.413924,1.969386292,0.0002608808787,-5.62719508e-008,
    2.723336049e-012,4.266685808e-016},
  bhigh={11737.73458,8.961085649999999},
  R=127.1520415965744);

constant IdealGases.Common.DataRecord Znplus (
  name="Znplus",
  MM=0.06538945139999999,
  Hf=15950586.91683717,
  H0=94777.18297541796,
  Tlimit=1000,
  alow={0.000409834494,-4.34358162e-006,2.500000019,-4.389036810000001e-011,
    5.5639692e-014,-3.63822826e-017,9.607881994999999e-021},
  blow={124697.9918,5.8119955},
  ahighe={-343617.946,956.7355239999999,1.511478952,0.000461346796,-
    8.786800980000001e-008,
    7.558567779999999e-013,1.168827311e-015},
  bhigh={118532.1933,13.0074267},
  R=127.1531083681794);

constant IdealGases.Common.DataRecord Zr (
  name="Zr",

```

---

**1232 Modelica.Media.IdealGases.Common.SingleGasesData**

---

```
MM=0.091224,
Hf=6569747.116986759,
H0=74712.91546084364,
Tlimit=1000,
alow={67158.9996,-943.5981740000001,6.35975618,-0.0009790119730000001,-
7.60822415e-006,
9.30871743e-009,-3.124675586e-012},
blow={75880.1946999999,-16.65770522},
ahigh={6006771.84,-15669.60605,17.9698235,-0.00676340965,1.733678968e-006,-
2.064699786e-010,
9.33409261000001e-015},
bhigh={173463.6249,-105.1117377},
R=91.1434710163992);

constant IdealGases.Common.DataRecord Zrplus(
name="Zrplus",
MM=0.0912234514,
Hf=13661468.32721131,
H0=81907.57842779889,
Tlimit=1000,
alow={173984.2193,-2224.598466,14.00787829,-0.02378785396,2.641058912e-005,
-1.442565487e-008,3.135982142e-012},
blow={159821.0714,-58.1672881},
ahigh={729813.716,-2017.117556,5.0374983,-0.000550337195,1.023753499e-007,-
1.261537793e-011,
7.092401041999999e-016},
bhigh={162088.4945,-9.820640859999999},
R=91.14401913541281);

constant IdealGases.Common.DataRecord Zrminus(
name="Zrminus",
MM=0.09122454860000001,
Hf=6061442.961198715,
H0=84949.69960311757,
Tlimit=1000,
alow={30466.62367,-807.427721,9.21300611,-0.01614342054,1.908653551e-005,-
1.138888914e-008,
2.745019116e-012},
blow={69030.3403,-28.62644371},
ahigh={84718.61159999999,317.543934,2.251491246,0.0001018645389,-
2.285208242e-008,
2.647293502e-012,-1.235813604e-016},
bhigh={64223.376,10.81261057},
R=91.14292290397806);

constant IdealGases.Common.DataRecord ZrN(
name="ZrN",
MM=0.1052307,
Hf=6779124.342991162,
H0=84223.2352345846,
Tlimit=1000,
alow={22591.09156,-180.2590198,3.130058377,0.00542770928,-
8.264614840000001e-006,
5.95843822e-009,-1.670274535e-012},
blow={85788.81490000001,8.47072477999999},
ahigh={-72557.9089,-81.267966,4.5599244,1.29116165e-005,
5.203830769999999e-009,-5.92743667e-013,2.731548854e-017},
bhigh={84686.48390000001,1.493633264},
```

```

R=79.01184730311593);

constant IdealGases.Common.DataRecord ZrO(
  name="ZrO",
  MM=0.1072234,
  Hf=782690.2336616821,
  H0=83658.11940304076,
  Tlimit=1000,
  allow={-509176.14,8652.77009,-52.9474015,0.1728961761,-0.0002457230895,
    1.672135156e-007,-4.423012380000001e-011},
  blow={-30951.29818,313.2576719},
  ahigh={464809.831,344.231447,4.81577918,-0.000466063314,2.140489079e-007,-
2.054364483e-011,
  4.08466776e-016},
  bhigh={7317.3407,1.502933548},
  R=77.54344667302101);

constant IdealGases.Common.DataRecord ZrOplus(
  name="ZrOplus",
  MM=0.1072228514,
  Hf=6720713.957808437,
  H0=88265.83024446596,
  Tlimit=1000,
  allow={10329.11549,73.0466122,2.606345299,0.005099300290000001,-6.25033876e-
006,
  3.79746002e-009,-9.198416899999999e-013},
  blow={85332.33779999999,12.67311019},
  ahigh={-493716.656,669.990076,4.57535347,-0.00069232379,4.28060094e-007,-
7.54766022e-011,
  4.41427986e-015},
  bhigh={80188.80660000001,2.928943704},
  R=77.5438434199298);

constant IdealGases.Common.DataRecord ZrO2(
  name="ZrO2",
  MM=0.1232228,
  Hf=-2572922.681516732,
  H0=97451.63232778349,
  Tlimit=1000,
  allow={36376.49,-262.0658297,3.69286687,0.01214524156,-1.822445342e-005,
  1.299215204e-008,-3.61590011e-012},
  blow={-38019.9088,8.290857600000001},
  ahigh={2854363.887,-8738.58988999999,16.21315114,-0.004417922830000001,
  8.959096920000001e-007,-4.054667109999999e-011,-2.147732083e-015},
  bhigh={15275.10023,-74.8199549},
  R=67.47511012572349);

```

---

## Modelica.Media.IdealGases.MixtureGases

**Medium models consisting of mixtures of ideal gases**

## Information

### Package Content

Name	Description
<a href="#">CombustionAir</a>	Air as mixture of N <sub>2</sub> and O <sub>2</sub>
<a href="#">AirSteam</a>	air and steam mixture (no condensation!, pseudo-mixture)
<a href="#">FlueGasLambdaOnePlus</a>	simple flue gas for over0stochiometric O <sub>2</sub> -fuel ratios
<a href="#">FlueGasSixComponents</a>	simplest flue gas for over-and understochiometric combustion of hydrocarbons
<a href="#">SimpleNaturalGas</a>	simple natural gas mix with 6 components
<a href="#">SimpleNaturalGasFixedComposition</a>	Same as SimpleNaturalGas but with fixed composition

### Types and constants

```
package simpleMoistAir = AirSteam(reference_X={0.03, 0.97})  
"moist air without condensation";
```

---

## Modelica.Media.IdealGases.MixtureGases.CombustionAir

Air as mixture of N<sub>2</sub> and O<sub>2</sub>

## Information

## Modelica.Media.IdealGases.MixtureGases.AirSteam

air and steam mixture (no condensation!, pseudo-mixture)

## Information

## Modelica.Media.IdealGases.MixtureGases.FlueGasLambdaOnePlus

simple flue gas for over0stochiometric O<sub>2</sub>-fuel ratios

## Information

## Modelica.Media.IdealGases.MixtureGases.FlueGasSixComponents

simplest flue gas for over-and understochiometric combustion of hydrocarbons

## Information

**Modelica.Media.IdealGases.MixtureGases.SimpleNaturalGas**

simple natural gas mix with 6 components

**Information****Modelica.Media.IdealGases.MixtureGases.SimpleNaturalGasFixedComposition**

Same as SimpleNaturalGas but with fixed composition

**Information****Modelica.Media.IdealGases.SingleGases**

Media models of ideal gases from NASA tables

**Information**

This package contains the data records for the 1241 ideal gases from

McBride B.J., Zehe M.J., and Gordon S. (2002): **NASA Glenn Coefficients for Calculating Thermodynamic Properties of Individual Species**. NASA report TP-2002-211556

The record entries are defined in **IdealGases.DataRecord**. Medium models for the following gases are provided:

Ag	BaOH+	C2H4O_ethyleng_o	DF	In2I4	Nb	ScO2
Ag+	Ba_OH_2	CH3CHO Ethanal	DOCl	In2I6	Nb+	Sc2O
Ag-	BaS	CH3COOH	DO2	In2O	Nb-	Sc2O2
Air	Ba2	OHCH2COOH	DO2-	K	NbCl5	Si
Al	Be	C2H5	D2	K+	NbO	Si+
Al+	Be+	C2H5Br	D2+	K-	NbOC13	Si-
Al-	Be++	C2H6	D2-	KAlF4	NbO2	SiBr
AlBr	BeBr	CH3N2CH3	D2O	KBO2	Ne	SiBr2
AlBr2	BeBr2	C2H5OH	D2O2	KBr	Ne+	SiBr3
AlBr3	BeCl	CH3OCH3	D2S	KCN	Ni	SiBr4
AlC	BeCl2	CH3O2CH3	e-	KCl	Ni+	SiC
AlC2	BeF	CCN	F	KF	Ni-	SiC2
AlC1	BeF2	CNC	F+	KH	NiCl	SiCl
AlC1+	BeH	OCCN	F-	KI	NiCl2	SiCl2
AlC12	BeH+	C2N2	FCN	Kli	NiO	SiCl3
AlC13	BeH2	C2O	FCO	KNO2	NiS	SiCl4
AlF	BeI	C3	FO	KNO3	O	SiF
AlF+	BeI2	C3H3_1_propynl	FO2_FOO	KNa	O+	SiFC1
AlFC1	BeN	C3H3_2_propynl	FO2_OF0	KO	O-	SiF2
AlFC12	BeO	C3H4_allene	F2	KOH	OD	SiF3
AlF2	BeOH	C3H4_propyne	F2O	K2	OD-	SiF4
AlF2-	BeOH+	C3H4_cyclo	F2O2	K2+	OH	SiH
AlF2C1	Be_OH_2	C3H5_allyl	FS2F	K2Br2	OH+	SiH+
AlF3	BeS	C3H6_propylene	Fe	K2CO3	OH-	SiHBr3
AlF4-	Be2	C3H6_cyclo	Fe+	K2C2N2	O2	SiHCl
AlH	Be2C14	C3H6O_propylox	Fe_CO_5	K2Cl2	O2+	SiHCl3
AlHC1	Be2F4	C3H6O_acetone	FeCl	K2F2	O2-	SiHF
AlHC12	Be2O	C3H6O_propanal	FeCl2	K2I2	O3	SiHF3
AlHF	Be2OF2	C3H7_n_propyl	FeCl3	K2O	P	SiHI3

AlHFC1	Be2O2	C3H7_i_propyl	FeO	K2O+	P+	SiH2
AlHF2	Be3O3	C3H8	Fe_OH_2	K2O2	P-	SiH2Br2
AlH2	Be4O4	C3H8O_1propanol	Fe2C14	K2O2H2	PC1	SiH2Cl2
AlH2Cl	Br	C3H8O_2propanol	Fe2C16	K2SO4	PC12	SiH2F2
AlH2F	Br+	CNCOCN	Ga	Kr	PC12-	SiH2I2
AlH3	Br-	C3O2	Ga+	Kr+	PC13	SiH3
AlI	BrCl	C4	GaBr	li	PC15	SiH3Br
AlI2	BrF	C4H2_butadiyne	GaBr2	li+	PF	SiH3Cl
AlI3	BrF3	C4H4_1_3-cyclo	GaBr3	li-	PF+	SiH3F
AlN	BrF5	C4H6_butadiene	GaCl	lia1F4	PF-	SiH3I
AlO	BrO	C4H6_1butyne	GaCl2	libO2	PFC1	SiH4
AlO+	OBrO	C4H6_2butyne	GaCl3	libr	PFC1-	SiI
AlO-	BrOO	C4H6_cyclo	GaF	licl	PFC12	SiI2
AlOC1	Bro3	C4H8_1_butene	GaF2	lif	PFC14	SiN
AlOC12	Br2	C4H8_cis2_buten	GaF3	liH	PF2	SiO
AlOF	BrBrO	C4H8_isobutene	GaH	liI	PF2-	SiO2
AlOF2	BrOBr	C4H8_cyclo	GaI	lin	PF2C1	Sis
AlOF2-	C	C4H9_n_butyl	GaI2	linO2	PF2C13	Sis2
AlOH	C+	C4H9_i_butyl	GaI3	linO3	PF3	Si2
AlOHC1	C-	C4H9_s_butyl	GaO	lio	PF3C12	Si2C
AlOHC12	CBr	C4H9_t_butyl	GaOH	lioF	PF4C1	Si2F6
AlOHF	CBr2	C4H10_n_butane	Ga2Br2	lioH	PF5	Si2N
AlOHF2	CBr3	C4H10_isobutane	Ga2Br4	lion	PH	Si3
AlO2	CBr4	C4N2	Ga2Br6	li2	PH2	Sn
AlO2-	CC1	C5	Ga2C12	li2+	PH2-	Sn+
Al_OH_2	CC12	C5H6_1_3cyclo	Ga2C14	li2Br2	PH3	Sn-
Al_OH_2C1	CC12Br2	C5H8_cyclo	Ga2C16	li2F2	PN	SnBr
Al_OH_2F	CC13	C5H10_1_pentene	Ga2F2	li2I2	PO	SnBr2
Al_OH_3	CC13Br	C5H10_cyclo	Ga2F4	li2O	PO-	SnBr3
AlS	CC14	C5H11_pentyl	Ga2F6	li2O+	POC13	SnBr4
AlS2	CF	C5H11_t_pentyl	Ga2I2	li2O2	POFC12	SnCl
Al2	CF+	C5H12_n_pentane	Ga2I4	li2O2H2	POF2C1	SnC12
Al2Br6	CFBr3	C5H12_i_pentane	Ga2I6	li2SO4	POF3	SnC13
Al2C2	CFC1	CH3C_CH3_2CH3	Ga2O	li3+	PO2	SnC14
Al2C16	CFC1Br2	C6D5_phenyl	Ge	li3Br3	PO2-	SnF
Al2F6	CFC12	C6D6	Ge+	li3C13	PS	SnF2
Al2I6	CFC12Br	C6H2	Ge-	li3F3	P2	SnF3
Al2O	CFC13	C6H5_phenyl	GeBr	li3I3	P2O3	SnF4
Al2O+	CF2	C6H5O_phenoxy	GeBr2	Mg	P2O4	SnI
Al2O2	CF2+	C6H6	GeBr3	Mg+	P2O5	SnI2
Al2O2+	CF2Br2	C6H5OH_phenol	GeBr4	MgBr	P3	SnI3
Al2O3	CF2C1	C6H10_cyclo	GeCl	MgBr2	P3O6	SnI4
Al2S	CF2C1Br	C6H12_1_hexene	GeCl2	MgCl	P4	SnO
Al2S2	CF2C12	C6H12_cyclo	GeCl3	MgCl+	P4O6	SnO2
Ar	CF3	C6H13_n_hexyl	GeCl4	MgCl2	P4O7	SnS
Ar+	CF3+	C6H14_n_hexane	GeF	MgF	P4O8	SnS2
B	CF3Br	C7H7_benzyl	GeF2	MgF+	P4O9	Sn2
B+	CF3C1	C7H8	GeF3	MgF2	P4O10	Sr
B-	CF4	C7H8O_cresol_mx	GeF4	MgF2+	Pb	Sr+
BBr	CH+	C7H14_1_heptene	GeH4	MgH	Pb+	SrBr
BBr2	CHBr3	C7H15_n_heptyl	GeI	MgI	Pb-	SrBr2
BBr3	CHCl	C7H16_n_heptane	GeO	MgI2	PbBr	SrCl
BC	CHClBr2	C7H16_2_methylh	GeO2	MgN	PbBr2	SrCl+
BC2	CHCl2	C8H8_styrene	GeS	MgO	PbBr3	SrCl2
BC1	CHCl2Br	C8H10_ethylbenz	GeS2	MgOH	PbBr4	SrF
BC1+	CHCl3	C8H16_1_octene	Ge2	MgOH+	PbCl	SrF+
BC1OH	CHF	C8H17_n_octyl	H	Mg_OH_2	PbCl2	SrF2
BC1_OH_2	CHFBr2	C8H18_n_octane	H+	MgS	PbCl3	SrH
BC12	CHFC1	C8H18_isooctane	H-	Mg2	PbCl4	SrI

BC12+	CHFC1Br	C9H19_n_nonyl	HALO	Mg2F4	PbF	SrI2
BC12OH	CHFC12	C10H8_naphthale	HALO2	Mn	PbF2	SrO
BF	CHF2	C10H21_n_decyl	HBO	Mn+	PbF3	SrOH
BFC1	CHF2Br	C12H9_o_bipheny	HBO+	Mo	PbF4	SrOH+
BFC12	CHF2Cl	C12H10_biphenyl	HBO2	Mo+	PbI	Sr_OH_2
BFOH	CHF3	Ca	HBS	Mo-	PbI2	SrS
BF_OH_2	CHI3	Ca+	HBS+	MoO	PbI3	Sr2
BF2	CH2	CaBr	HCN	MoO2	PbI4	Ta
BF2+	CH2Br2	CaBr2	HCO	MoO3	PbO	Ta+
BF2-	CH2Cl	CaCl	HCO+	MoO3-	PbO2	Ta-
BF2C1	CH2ClBr	CaCl+	HCCN	Mo2O6	PbS	TaCl5
BF2OH	CH2Cl2	CaCl2	HCCO	Mo3O9	PbS2	TaO
BF3	CH2F	CaF	HC1	Mo4O12	Rb	TaO2
BF4-	CH2FBr	CaF+	HD	Mo5O15	Rb+	Ti
BH	CH2FC1	CaF2	HD+	N	Rb-	Ti+
BHC1	CH2F2	CaH	HDO	N+	RbBO2	Ti-
BHC12	CH2I2	CaI	HDO2	N-	RbBr	TiCl
BHF	CH3	CaI2	HF	NCO	RbCl	TiCl2
BHFC1	CH3Br	CaO	HI	ND	RbF	TiCl3
BHF2	CH3Cl	CaO+	HNC	ND2	RbH	TiCl4
BH2	CH3F	CaOH	HNCO	ND3	RbI	TiO
BH2C1	CH3I	CaOH+	HNO	NF	RbK	TiO+
BH2F	CH2OH	Ca_OH_2	HNO2	NF2	Rbli	TiOC1
BH3	CH2OH+	CaS	HNO3	NF3	RbNO2	TiOC12
BH3NH3	CH3O	Ca2	HOCl	NH	RbNO3	TiO2
BH4	CH4	Cd	HOF	NH+	RbNa	U
BI	CH3OH	Cd+	HO2	NHF	RbO	UF
BI2	CH3OOH	Cl	HO2-	NHF2	RbOH	UF+
BI3	CI	Cl+	HPO	NH2	Rb2Br2	UF-
BN	CI2	Cl-	HSO3F	NH2F	Rb2C12	UF2
BO	CI3	ClCN	H2	NH3	Rb2F2	UF2+
BO-	CI4	ClF	H2+	NH2OH	Rb2I2	UF2-
BOC1	CN	ClF3	H2-	NH4+	Rb2O	UF3
BOC12	CN+	ClF5	HBOH	NO	Rb2O2	UF3+
BOF	CN-	ClO	HCOOH	NOCl	Rb2O2H2	UF3-
BOF2	CNN	ClO2	H2F2	NOF	Rb2SO4	UF4
BOH	CO	Cl2	H2O	NOF3	Rn	UF4+
BO2	CO+	Cl2O	H2O+	NO2	Rn+	UF4-
BO2-	COCl	Co	H2O2	NO2-	S	UF5
B_OH_2	COC12	Co+	H2S	NO2Cl	S+	UF5+
BS	COFC1	Co-	H2SO4	NO2F	S-	UF5-
BS2	COF2	Cr	H2BOH	NO3	SC1	UF6
B2	COHCl	Cr+	HB_OH_2	NO3-	SC12	UF6-
B2C	COHF	Cr-	H3BO3	NO3F	SC12+	UO
B2C14	COS	CrN	H3B3O3	N2	SD	UO+
B2F4	CO2	CrO	H3B3O6	N2+	SF	UOF
B2H	CO2+	CrO2	H3F3	N2-	SF+	UOF2
B2H2	COOH	CrO3	H3O+	NCN	SF-	UOF3
B2H3	CP	CrO3-	H4F4	N2D2_cis	SF2	UOF4
B2H3_db	CS	Cs	H5F5	N2F2	SF2+	UO2
B2H4	CS2	Cs+	H6F6	N2F4	SF2-	UO2+
B2H4_db	C2	Cs-	H7F7	N2H2	SF3	UO2-
B2H5	C2+	CsBO2	He	NH2NO2	SF3+	UO2F
B2H5_db	C2-	CsBr	He+	N2H4	SF3-	UO2F2
B2H6	C2Cl	CsCl	Hg	N2O	SF4	UO3
B2O	C2Cl2	CsF	Hg+	N2O+	SF4+	UO3-
B2O2	C2C13	CsH	HgBr2	N2O3	SF4-	V
B2O3	C2C14	CsI	I	N2O4	SF5	V+
B2_OH_4	C2C16	Csli	I+	N2O5	SF5+	V-

B2S	C2F	CsNO2	I-	N3	SF5-	VC14
B2S2	C2FC1	CsNO3	IF5	N3H	SF6	VN
B2S3	C2FC13	CsNa	IF7	Na	SF6-	VO
B3H7_C2v	C2F2	CsO	I2	Na+	SH	VO2
B3H7_Cs	C2F2C12	CsOH	In	Na-	SH-	V4O10
B3H9	C2F3	CsRb	In+	NaAlF4	SN	W
B3N3H6	C2F3C1	Cs2	InBr	NaBO2	SO	W+
B3O3C13	C2F4	Cs2Br2	InBr2	NaBr	SO-	W-
B3O3FC12	C2F6	Cs2CO3	InBr3	NaCN	SOF2	WC16
B3O3F2C1	C2H	Cs2Cl2	InCl	NaCl	SO2	WO
B3O3F3	C2HC1	Cs2F2	InCl2	NaF	SO2-	WOC14
B4H4	C2HC13	Cs2I2	InCl3	NaH	SO2C12	WO2
B4H10	C2HF	Cs2O	InF	NaI	SO2FC1	WO2C12
B4H12	C2HFC12	Cs2O+	InF2	Nali	SO2F2	WO3
B5H9	C2HF2C1	Cs2O2	InF3	NaNO2	SO3	WO3-
Ba	C2HF3	Cs2O2H2	InH	NaNO3	S2	Xe
Ba+	C2H2_vinylidene	Cs2SO4	InI	Nao	S2-	Xe+
BaBr	C2H2C12	Cu	InI2	NaOH	S2C12	Zn
BaBr2	C2H2FC1	Cu+	InI3	NaOH+	S2F2	Zn+
BaCl	C2H2F2	Cu-	InO	Na2	S2O	Zr
BaCl+	CH2CO_ketene	CuCl	InOH	Na2Br2	S3	Zr+
BaCl2	O_CH_2O	CuF	In2Br2	Na2C12	S4	Zr-
BaF	HO_CO_2OH	CuF2	In2Br4	Na2F2	S5	ZrN
BaF+	C2H3_vinyl	CuO	In2Br6	Na2I2	S6	ZrO
BaF2	CH2Br-COOH	Cu2	In2C12	Na2O	S7	ZrO+
BaH	C2H3C1	Cu3Cl3	In2C14	Na2O+	S8	ZrO2
BaI	CH2C1-COOH	D	In2C16	Na2O2	Sc	
BaI2	C2H3F	D+	In2F2	Na2O2H2	Sc+	
BaO	CH3CN	D-	In2F4	Na2SO4	Sc-	
BaO+	CH3CO_acetyl	DBr	In2F6	Na3C13	ScO	
BaOH	C2H4	DCl	In2I2	Na3F3	ScO+	

## Package Content

Name	Description
<input type="checkbox"/> Ag	Ideal gas "Ag" from NASA Glenn coefficients
<input type="checkbox"/> Agplus	Ideal gas "Ag+" from NASA Glenn coefficients
<input type="checkbox"/> Agminus	Ideal gas "Ag-" from NASA Glenn coefficients
<input type="checkbox"/> Air	Ideal gas "Air" from NASA Glenn coefficients
<input type="checkbox"/> Al	Ideal gas "Al" from NASA Glenn coefficients
<input type="checkbox"/> Alplus	Ideal gas "Al+" from NASA Glenn coefficients
<input type="checkbox"/> Alminus	Ideal gas "Al-" from NASA Glenn coefficients
<input type="checkbox"/> ALBr	Ideal gas "ALBr" from NASA Glenn coefficients
<input type="checkbox"/> ALBr2	Ideal gas "ALBr2" from NASA Glenn coefficients
<input type="checkbox"/> ALBr3	Ideal gas "ALBr3" from NASA Glenn coefficients
<input type="checkbox"/> ALC	Ideal gas "ALC" from NASA Glenn coefficients
<input type="checkbox"/> ALC2	Ideal gas "ALC2" from NASA Glenn coefficients
<input type="checkbox"/> ALCL	Ideal gas "ALCl" from NASA Glenn coefficients
<input type="checkbox"/> ALCLplus	Ideal gas "ALCl+" from NASA Glenn coefficients

<input type="checkbox"/> ALCL2	Ideal gas "AlCl2" from NASA Glenn coefficients
<input type="checkbox"/> ALCL3	Ideal gas "AlCl3" from NASA Glenn coefficients
<input type="checkbox"/> ALF	Ideal gas "AlF" from NASA Glenn coefficients
<input type="checkbox"/> ALFplus	Ideal gas "AlF+" from NASA Glenn coefficients
<input type="checkbox"/> ALFCL	Ideal gas "AlFCI" from NASA Glenn coefficients
<input type="checkbox"/> ALFCL2	Ideal gas "AlFCI2" from NASA Glenn coefficients
<input type="checkbox"/> ALF2	Ideal gas "AlF2" from NASA Glenn coefficients
<input type="checkbox"/> ALF2minus	Ideal gas "AlF2-" from NASA Glenn coefficients
<input type="checkbox"/> ALF2CL	Ideal gas "AlF2CI" from NASA Glenn coefficients
<input type="checkbox"/> ALF3	Ideal gas "AlF3" from NASA Glenn coefficients
<input type="checkbox"/> ALF4minus	Ideal gas "AlF4-" from NASA Glenn coefficients
<input type="checkbox"/> ALH	Ideal gas "AlH" from NASA Glenn coefficients
<input type="checkbox"/> ALHCL	Ideal gas "AlHCl" from NASA Glenn coefficients
<input type="checkbox"/> ALHCL2	Ideal gas "AlHCl2" from NASA Glenn coefficients
<input type="checkbox"/> ALHF	Ideal gas "AlHF" from NASA Glenn coefficients
<input type="checkbox"/> ALHFCL	Ideal gas "AlHFCl" from NASA Glenn coefficients
<input type="checkbox"/> ALHF2	Ideal gas "AlHF2" from NASA Glenn coefficients
<input type="checkbox"/> ALH2	Ideal gas "AlH2" from NASA Glenn coefficients
<input type="checkbox"/> ALH2CL	Ideal gas "AlH2CI" from NASA Glenn coefficients
<input type="checkbox"/> ALH2F	Ideal gas "AlH2F" from NASA Glenn coefficients
<input type="checkbox"/> ALH3	Ideal gas "AlH3" from NASA Glenn coefficients
<input type="checkbox"/> ALI	Ideal gas "AlI" from NASA Glenn coefficients
<input type="checkbox"/> ALI2	Ideal gas "AlI2" from NASA Glenn coefficients
<input type="checkbox"/> ALI3	Ideal gas "AlI3" from NASA Glenn coefficients
<input type="checkbox"/> ALN	Ideal gas "AlN" from NASA Glenn coefficients
<input type="checkbox"/> ALO	Ideal gas "AlO" from NASA Glenn coefficients
<input type="checkbox"/> ALOplus	Ideal gas "AlO+" from NASA Glenn coefficients
<input type="checkbox"/> ALOminus	Ideal gas "AlO-" from NASA Glenn coefficients
<input type="checkbox"/> ALOCL	Ideal gas "AlOCI" from NASA Glenn coefficients
<input type="checkbox"/> ALOCL2	Ideal gas "AlOCI2" from NASA Glenn coefficients
<input type="checkbox"/> ALOF	Ideal gas "AlOF" from NASA Glenn coefficients
<input type="checkbox"/> ALOF2	Ideal gas "AlOF2" from NASA Glenn coefficients
<input type="checkbox"/> ALOF2minus	Ideal gas "AlOF2-" from NASA Glenn coefficients
<input type="checkbox"/> ALOH	Ideal gas "AlOH" from NASA Glenn coefficients
<input type="checkbox"/> ALOHCL	Ideal gas "AlOHCi" from NASA Glenn coefficients
<input type="checkbox"/> ALOHCL2	Ideal gas "AlOHCi2" from NASA Glenn coefficients
<input type="checkbox"/> ALOHF	Ideal gas "AlOHF" from NASA Glenn coefficients

<input type="checkbox"/> ALOHF2	Ideal gas "ALOHF2" from NASA Glenn coefficients
<input type="checkbox"/> ALO2	Ideal gas "AIO2" from NASA Glenn coefficients
<input type="checkbox"/> ALO2minus	Ideal gas "AIO2-" from NASA Glenn coefficients
<input type="checkbox"/> AL_OH_2	Ideal gas "Al_OH_2" from NASA Glenn coefficients
<input type="checkbox"/> AL_OH_2CL	Ideal gas "Al_OH_2Cl" from NASA Glenn coefficients
<input type="checkbox"/> AL_OH_2F	Ideal gas "Al_OH_2F" from NASA Glenn coefficients
<input type="checkbox"/> AL_OH_3	Ideal gas "Al_OH_3" from NASA Glenn coefficients
<input type="checkbox"/> ALS	Ideal gas "AIS" from NASA Glenn coefficients
<input type="checkbox"/> ALS2	Ideal gas "AIS2" from NASA Glenn coefficients
<input type="checkbox"/> AL2	Ideal gas "Al2" from NASA Glenn coefficients
<input type="checkbox"/> AL2Br6	Ideal gas "Al2Br6" from NASA Glenn coefficients
<input type="checkbox"/> AL2C2	Ideal gas "Al2C2" from NASA Glenn coefficients
<input type="checkbox"/> AL2CL6	Ideal gas "Al2Cl6" from NASA Glenn coefficients
<input type="checkbox"/> AL2F6	Ideal gas "Al2F6" from NASA Glenn coefficients
<input type="checkbox"/> AL2I6	Ideal gas "Al2I6" from NASA Glenn coefficients
<input type="checkbox"/> AL2O	Ideal gas "Al2O" from NASA Glenn coefficients
<input type="checkbox"/> AL2Oplus	Ideal gas "Al2O+" from NASA Glenn coefficients
<input type="checkbox"/> AL2O2	Ideal gas "Al2O2" from NASA Glenn coefficients
<input type="checkbox"/> AL2O2plus	Ideal gas "Al2O2+" from NASA Glenn coefficients
<input type="checkbox"/> AL2O3	Ideal gas "Al2O3" from NASA Glenn coefficients
<input type="checkbox"/> AL2S	Ideal gas "Al2S" from NASA Glenn coefficients
<input type="checkbox"/> AL2S2	Ideal gas "Al2S2" from NASA Glenn coefficients
<input type="checkbox"/> Ar	Ideal gas "Ar" from NASA Glenn coefficients
<input type="checkbox"/> Arplus	Ideal gas "Ar+" from NASA Glenn coefficients
<input type="checkbox"/> B	Ideal gas "B" from NASA Glenn coefficients
<input type="checkbox"/> Bplus	Ideal gas "B+" from NASA Glenn coefficients
<input type="checkbox"/> Bminus	Ideal gas "B-" from NASA Glenn coefficients
<input type="checkbox"/> BBr	Ideal gas "BBr" from NASA Glenn coefficients
<input type="checkbox"/> BBr2	Ideal gas "BBr2" from NASA Glenn coefficients
<input type="checkbox"/> BBr3	Ideal gas "BBr3" from NASA Glenn coefficients
<input type="checkbox"/> BC	Ideal gas "BC" from NASA Glenn coefficients
<input type="checkbox"/> BC2	Ideal gas "BC2" from NASA Glenn coefficients
<input type="checkbox"/> BCL	Ideal gas "BCl" from NASA Glenn coefficients
<input type="checkbox"/> BCLplus	Ideal gas "BCl+" from NASA Glenn coefficients
<input type="checkbox"/> BCLOH	Ideal gas "BCIOH" from NASA Glenn coefficients
<input type="checkbox"/> BCL_OH_2	Ideal gas "BCl_OH_2" from NASA Glenn coefficients
<input type="checkbox"/> BCL2	Ideal gas "BCl2" from NASA Glenn coefficients

<input type="checkbox"/> BCL2plus	Ideal gas "BCI2+" from NASA Glenn coefficients
<input type="checkbox"/> BCL2OH	Ideal gas "BCI2OH" from NASA Glenn coefficients
<input type="checkbox"/> BF	Ideal gas "BF" from NASA Glenn coefficients
<input type="checkbox"/> BFCL	Ideal gas "BFCI" from NASA Glenn coefficients
<input type="checkbox"/> BFCL2	Ideal gas "BFCI2" from NASA Glenn coefficients
<input type="checkbox"/> BFOH	Ideal gas "BFOH" from NASA Glenn coefficients
<input type="checkbox"/> BF_OH_2	Ideal gas "BF_OH_2" from NASA Glenn coefficients
<input type="checkbox"/> BF2	Ideal gas "BF2" from NASA Glenn coefficients
<input type="checkbox"/> BF2plus	Ideal gas "BF2+" from NASA Glenn coefficients
<input type="checkbox"/> BF2minus	Ideal gas "BF2-" from NASA Glenn coefficients
<input type="checkbox"/> BF2CL	Ideal gas "BF2CI" from NASA Glenn coefficients
<input type="checkbox"/> BF2OH	Ideal gas "BF2OH" from NASA Glenn coefficients
<input type="checkbox"/> BF3	Ideal gas "BF3" from NASA Glenn coefficients
<input type="checkbox"/> BF4minus	Ideal gas "BF4-" from NASA Glenn coefficients
<input type="checkbox"/> BH	Ideal gas "BH" from NASA Glenn coefficients
<input type="checkbox"/> BHCL	Ideal gas "BHCI" from NASA Glenn coefficients
<input type="checkbox"/> BHCL2	Ideal gas "BHCi2" from NASA Glenn coefficients
<input type="checkbox"/> BHF	Ideal gas "BHF" from NASA Glenn coefficients
<input type="checkbox"/> BHFCI	Ideal gas "BHFCI" from NASA Glenn coefficients
<input type="checkbox"/> BHF2	Ideal gas "BHF2" from NASA Glenn coefficients
<input type="checkbox"/> BH2	Ideal gas "BH2" from NASA Glenn coefficients
<input type="checkbox"/> BH2CL	Ideal gas "BH2CI" from NASA Glenn coefficients
<input type="checkbox"/> BH2F	Ideal gas "BH2F" from NASA Glenn coefficients
<input type="checkbox"/> BH3	Ideal gas "BH3" from NASA Glenn coefficients
<input type="checkbox"/> BH3NH3	Ideal gas "BH3NH3" from NASA Glenn coefficients
<input type="checkbox"/> BH4	Ideal gas "BH4" from NASA Glenn coefficients
<input type="checkbox"/> BI	Ideal gas "BI" from NASA Glenn coefficients
<input type="checkbox"/> BI2	Ideal gas "BI2" from NASA Glenn coefficients
<input type="checkbox"/> BI3	Ideal gas "BI3" from NASA Glenn coefficients
<input type="checkbox"/> BN	Ideal gas "BN" from NASA Glenn coefficients
<input type="checkbox"/> BO	Ideal gas "BO" from NASA Glenn coefficients
<input type="checkbox"/> BOminus	Ideal gas "BO-" from NASA Glenn coefficients
<input type="checkbox"/> BOCL	Ideal gas "BOCI" from NASA Glenn coefficients
<input type="checkbox"/> BOCL2	Ideal gas "BOCI2" from NASA Glenn coefficients
<input type="checkbox"/> BOF	Ideal gas "BOF" from NASA Glenn coefficients
<input type="checkbox"/> BOF2	Ideal gas "BOF2" from NASA Glenn coefficients
<input type="checkbox"/> BOH	Ideal gas "BOH" from NASA Glenn coefficients

<input type="checkbox"/> BO2	Ideal gas "BO2" from NASA Glenn coefficients
<input type="checkbox"/> BO2minus	Ideal gas "BO2-" from NASA Glenn coefficients
<input type="checkbox"/> B_OH_2	Ideal gas "B_OH_2" from NASA Glenn coefficients
<input type="checkbox"/> BS	Ideal gas "BS" from NASA Glenn coefficients
<input type="checkbox"/> BS2	Ideal gas "BS2" from NASA Glenn coefficients
<input type="checkbox"/> B2	Ideal gas "B2" from NASA Glenn coefficients
<input type="checkbox"/> B2C	Ideal gas "B2C" from NASA Glenn coefficients
<input type="checkbox"/> B2Cl4	Ideal gas "B2Cl4" from NASA Glenn coefficients
<input type="checkbox"/> B2F4	Ideal gas "B2F4" from NASA Glenn coefficients
<input type="checkbox"/> B2H	Ideal gas "B2H" from NASA Glenn coefficients
<input type="checkbox"/> B2H2	Ideal gas "B2H2" from NASA Glenn coefficients
<input type="checkbox"/> B2H3	Ideal gas "B2H3" from NASA Glenn coefficients
<input type="checkbox"/> B2H3_db	Ideal gas "B2H3_db" from NASA Glenn coefficients
<input type="checkbox"/> B2H4	Ideal gas "B2H4" from NASA Glenn coefficients
<input type="checkbox"/> B2H4_db	Ideal gas "B2H4_db" from NASA Glenn coefficients
<input type="checkbox"/> B2H5	Ideal gas "B2H5" from NASA Glenn coefficients
<input type="checkbox"/> B2H5_db	Ideal gas "B2H5_db" from NASA Glenn coefficients
<input type="checkbox"/> B2H6	Ideal gas "B2H6" from NASA Glenn coefficients
<input type="checkbox"/> B2O	Ideal gas "B2O" from NASA Glenn coefficients
<input type="checkbox"/> B2O2	Ideal gas "B2O2" from NASA Glenn coefficients
<input type="checkbox"/> B2O3	Ideal gas "B2O3" from NASA Glenn coefficients
<input type="checkbox"/> B2_OH_4	Ideal gas "B2_OH_4" from NASA Glenn coefficients
<input type="checkbox"/> B2S	Ideal gas "B2S" from NASA Glenn coefficients
<input type="checkbox"/> B2S2	Ideal gas "B2S2" from NASA Glenn coefficients
<input type="checkbox"/> B2S3	Ideal gas "B2S3" from NASA Glenn coefficients
<input type="checkbox"/> B3H7_C2v	Ideal gas "B3H7_C2v" from NASA Glenn coefficients
<input type="checkbox"/> B3H7_Cs	Ideal gas "B3H7_Cs" from NASA Glenn coefficients
<input type="checkbox"/> B3H9	Ideal gas "B3H9" from NASA Glenn coefficients
<input type="checkbox"/> B3N3H6	Ideal gas "B3N3H6" from NASA Glenn coefficients
<input type="checkbox"/> B3O3CL3	Ideal gas "B3O3Cl3" from NASA Glenn coefficients
<input type="checkbox"/> B3O3FCL2	Ideal gas "B3O3FCI2" from NASA Glenn coefficients
<input type="checkbox"/> B3O3F2CL	Ideal gas "B3O3F2Cl" from NASA Glenn coefficients
<input type="checkbox"/> B3O3F3	Ideal gas "B3O3F3" from NASA Glenn coefficients
<input type="checkbox"/> B4H4	Ideal gas "B4H4" from NASA Glenn coefficients
<input type="checkbox"/> B4H10	Ideal gas "B4H10" from NASA Glenn coefficients
<input type="checkbox"/> B4H12	Ideal gas "B4H12" from NASA Glenn coefficients
<input type="checkbox"/> B5H9	Ideal gas "B5H9" from NASA Glenn coefficients

<input type="checkbox"/> Ba	Ideal gas "Ba" from NASA Glenn coefficients
<input type="checkbox"/> Baplus	Ideal gas "Ba+" from NASA Glenn coefficients
<input type="checkbox"/> BaBr	Ideal gas "BaBr" from NASA Glenn coefficients
<input type="checkbox"/> BaBr2	Ideal gas "BaBr2" from NASA Glenn coefficients
<input type="checkbox"/> BaCL	Ideal gas "BaCl" from NASA Glenn coefficients
<input type="checkbox"/> BaCLplus	Ideal gas "BaCl+" from NASA Glenn coefficients
<input type="checkbox"/> BaCL2	Ideal gas "BaCl2" from NASA Glenn coefficients
<input type="checkbox"/> BaF	Ideal gas "BaF" from NASA Glenn coefficients
<input type="checkbox"/> BaFplus	Ideal gas "BaF+" from NASA Glenn coefficients
<input type="checkbox"/> BaF2	Ideal gas "BaF2" from NASA Glenn coefficients
<input type="checkbox"/> BaH	Ideal gas "BaH" from NASA Glenn coefficients
<input type="checkbox"/> Bal	Ideal gas "Bal" from NASA Glenn coefficients
<input type="checkbox"/> Bal2	Ideal gas "Bal2" from NASA Glenn coefficients
<input type="checkbox"/> BaO	Ideal gas "BaO" from NASA Glenn coefficients
<input type="checkbox"/> BaOplus	Ideal gas "BaO+" from NASA Glenn coefficients
<input type="checkbox"/> BaOH	Ideal gas "BaOH" from NASA Glenn coefficients
<input type="checkbox"/> BaOHplus	Ideal gas "BaOH+" from NASA Glenn coefficients
<input type="checkbox"/> Ba_OH_2	Ideal gas "Ba_OH_2" from NASA Glenn coefficients
<input type="checkbox"/> BaS	Ideal gas "BaS" from NASA Glenn coefficients
<input type="checkbox"/> Ba2	Ideal gas "Ba2" from NASA Glenn coefficients
<input type="checkbox"/> Be	Ideal gas "Be" from NASA Glenn coefficients
<input type="checkbox"/> Beplus	Ideal gas "Be+" from NASA Glenn coefficients
<input type="checkbox"/> Beplusplus	Ideal gas "Be++" from NASA Glenn coefficients
<input type="checkbox"/> BeBr	Ideal gas "BeBr" from NASA Glenn coefficients
<input type="checkbox"/> BeBr2	Ideal gas "BeBr2" from NASA Glenn coefficients
<input type="checkbox"/> BeCL	Ideal gas "BeCl" from NASA Glenn coefficients
<input type="checkbox"/> BeCL2	Ideal gas "BeCl2" from NASA Glenn coefficients
<input type="checkbox"/> BeF	Ideal gas "BeF" from NASA Glenn coefficients
<input type="checkbox"/> BeF2	Ideal gas "BeF2" from NASA Glenn coefficients
<input type="checkbox"/> BeH	Ideal gas "BeH" from NASA Glenn coefficients
<input type="checkbox"/> BeHplus	Ideal gas "BeH+" from NASA Glenn coefficients
<input type="checkbox"/> BeH2	Ideal gas "BeH2" from NASA Glenn coefficients
<input type="checkbox"/> Bel	Ideal gas "Bel" from NASA Glenn coefficients
<input type="checkbox"/> Bel2	Ideal gas "Bel2" from NASA Glenn coefficients
<input type="checkbox"/> BeN	Ideal gas "BeN" from NASA Glenn coefficients
<input type="checkbox"/> BeO	Ideal gas "BeO" from NASA Glenn coefficients
<input type="checkbox"/> BeOH	Ideal gas "BeOH" from NASA Glenn coefficients

<input type="checkbox"/> BeOHplus	Ideal gas "BeOH+" from NASA Glenn coefficients
<input type="checkbox"/> Be_OH_2	Ideal gas "Be_OH_2" from NASA Glenn coefficients
<input type="checkbox"/> BeS	Ideal gas "BeS" from NASA Glenn coefficients
<input type="checkbox"/> Be2	Ideal gas "Be2" from NASA Glenn coefficients
<input type="checkbox"/> Be2Cl4	Ideal gas "Be2Cl4" from NASA Glenn coefficients
<input type="checkbox"/> Be2F4	Ideal gas "Be2F4" from NASA Glenn coefficients
<input type="checkbox"/> Be2O	Ideal gas "Be2O" from NASA Glenn coefficients
<input type="checkbox"/> Be2OF2	Ideal gas "Be2OF2" from NASA Glenn coefficients
<input type="checkbox"/> Be2O2	Ideal gas "Be2O2" from NASA Glenn coefficients
<input type="checkbox"/> Be3O3	Ideal gas "Be3O3" from NASA Glenn coefficients
<input type="checkbox"/> Be4O4	Ideal gas "Be4O4" from NASA Glenn coefficients
<input type="checkbox"/> Br	Ideal gas "Br" from NASA Glenn coefficients
<input type="checkbox"/> Brplus	Ideal gas "Br+" from NASA Glenn coefficients
<input type="checkbox"/> Brminus	Ideal gas "Br-" from NASA Glenn coefficients
<input type="checkbox"/> BrCl	Ideal gas "BrCl" from NASA Glenn coefficients
<input type="checkbox"/> BrF	Ideal gas "BrF" from NASA Glenn coefficients
<input type="checkbox"/> BrF3	Ideal gas "BrF3" from NASA Glenn coefficients
<input type="checkbox"/> BrF5	Ideal gas "BrF5" from NASA Glenn coefficients
<input type="checkbox"/> BrO	Ideal gas "BrO" from NASA Glenn coefficients
<input type="checkbox"/> OBrO	Ideal gas "OBrO" from NASA Glenn coefficients
<input type="checkbox"/> BrOO	Ideal gas "BrOO" from NASA Glenn coefficients
<input type="checkbox"/> BrO3	Ideal gas "BrO3" from NASA Glenn coefficients
<input type="checkbox"/> Br2	Ideal gas "Br2" from NASA Glenn coefficients
<input type="checkbox"/> BrBrO	Ideal gas "BrBrO" from NASA Glenn coefficients
<input type="checkbox"/> BrOBr	Ideal gas "BrOBr" from NASA Glenn coefficients
<input type="checkbox"/> C	Ideal gas "C" from NASA Glenn coefficients
<input type="checkbox"/> Cplus	Ideal gas "C+" from NASA Glenn coefficients
<input type="checkbox"/> Cminus	Ideal gas "C-" from NASA Glenn coefficients
<input type="checkbox"/> CBr	Ideal gas "CBr" from NASA Glenn coefficients
<input type="checkbox"/> CBr2	Ideal gas "CBr2" from NASA Glenn coefficients
<input type="checkbox"/> CBr3	Ideal gas "CBr3" from NASA Glenn coefficients
<input type="checkbox"/> CBr4	Ideal gas "CBr4" from NASA Glenn coefficients
<input type="checkbox"/> CCl	Ideal gas "CCl" from NASA Glenn coefficients
<input type="checkbox"/> CCl2	Ideal gas "CCl2" from NASA Glenn coefficients
<input type="checkbox"/> CCl2Br2	Ideal gas "CCl2Br2" from NASA Glenn coefficients
<input type="checkbox"/> CCl3	Ideal gas "CCl3" from NASA Glenn coefficients
<input type="checkbox"/> CCl3Br	Ideal gas "CCl3Br" from NASA Glenn coefficients

<input type="checkbox"/> CCL4	Ideal gas "CCl4" from NASA Glenn coefficients
<input type="checkbox"/> CF	Ideal gas "CF" from NASA Glenn coefficients
<input type="checkbox"/> CFplus	Ideal gas "CF+" from NASA Glenn coefficients
<input type="checkbox"/> CFBr3	Ideal gas "CFBr3" from NASA Glenn coefficients
<input type="checkbox"/> CFCL	Ideal gas "CFCI" from NASA Glenn coefficients
<input type="checkbox"/> CFCLBr2	Ideal gas "CFCIBr2" from NASA Glenn coefficients
<input type="checkbox"/> CFCL2	Ideal gas "CFCI2" from NASA Glenn coefficients
<input type="checkbox"/> CFCL2Br	Ideal gas "CFCI2Br" from NASA Glenn coefficients
<input type="checkbox"/> CFCL3	Ideal gas "CFCI3" from NASA Glenn coefficients
<input type="checkbox"/> CF2	Ideal gas "CF2" from NASA Glenn coefficients
<input type="checkbox"/> CF2plus	Ideal gas "CF2+" from NASA Glenn coefficients
<input type="checkbox"/> CF2Br2	Ideal gas "CF2Br2" from NASA Glenn coefficients
<input type="checkbox"/> CF2CL	Ideal gas "CF2Cl" from NASA Glenn coefficients
<input type="checkbox"/> CF2CLBr	Ideal gas "CF2ClBr" from NASA Glenn coefficients
<input type="checkbox"/> CF2CL2	Ideal gas "CF2Cl2" from NASA Glenn coefficients
<input type="checkbox"/> CF3	Ideal gas "CF3" from NASA Glenn coefficients
<input type="checkbox"/> CF3plus	Ideal gas "CF3+" from NASA Glenn coefficients
<input type="checkbox"/> CF3Br	Ideal gas "CF3Br" from NASA Glenn coefficients
<input type="checkbox"/> CF3CL	Ideal gas "CF3Cl" from NASA Glenn coefficients
<input type="checkbox"/> CF4	Ideal gas "CF4" from NASA Glenn coefficients
<input type="checkbox"/> CHplus	Ideal gas "CH+" from NASA Glenn coefficients
<input type="checkbox"/> CHBr3	Ideal gas "CHBr3" from NASA Glenn coefficients
<input type="checkbox"/> CHCL	Ideal gas "CHCl" from NASA Glenn coefficients
<input type="checkbox"/> CHCLBr2	Ideal gas "CHClBr2" from NASA Glenn coefficients
<input type="checkbox"/> CHCL2	Ideal gas "CHCl2" from NASA Glenn coefficients
<input type="checkbox"/> CHCL2Br	Ideal gas "CHCl2Br" from NASA Glenn coefficients
<input type="checkbox"/> CHCL3	Ideal gas "CHCl3" from NASA Glenn coefficients
<input type="checkbox"/> CHF	Ideal gas "CHF" from NASA Glenn coefficients
<input type="checkbox"/> CHFBr2	Ideal gas "CHFBr2" from NASA Glenn coefficients
<input type="checkbox"/> CHFCL	Ideal gas "CHFCI" from NASA Glenn coefficients
<input type="checkbox"/> CHFCLBr	Ideal gas "CHFClBr" from NASA Glenn coefficients
<input type="checkbox"/> CHFCL2	Ideal gas "CHFCI2" from NASA Glenn coefficients
<input type="checkbox"/> CHF2	Ideal gas "CHF2" from NASA Glenn coefficients
<input type="checkbox"/> CHF2Br	Ideal gas "CHF2Br" from NASA Glenn coefficients
<input type="checkbox"/> CHF2CL	Ideal gas "CHF2Cl" from NASA Glenn coefficients
<input type="checkbox"/> CHF3	Ideal gas "CHF3" from NASA Glenn coefficients
<input type="checkbox"/> CHI3	Ideal gas "CHI3" from NASA Glenn coefficients

<input type="checkbox"/> CH2	Ideal gas "CH2" from NASA Glenn coefficients
<input type="checkbox"/> CH2Br2	Ideal gas "CH2Br2" from NASA Glenn coefficients
<input type="checkbox"/> CH2CL	Ideal gas "CH2Cl" from NASA Glenn coefficients
<input type="checkbox"/> CH2CLBr	Ideal gas "CH2ClBr" from NASA Glenn coefficients
<input type="checkbox"/> CH2CL2	Ideal gas "CH2Cl2" from NASA Glenn coefficients
<input type="checkbox"/> CH2F	Ideal gas "CH2F" from NASA Glenn coefficients
<input type="checkbox"/> CH2FBr	Ideal gas "CH2FBr" from NASA Glenn coefficients
<input type="checkbox"/> CH2FCL	Ideal gas "CH2FCI" from NASA Glenn coefficients
<input type="checkbox"/> CH2F2	Ideal gas "CH2F2" from NASA Glenn coefficients
<input type="checkbox"/> CH2I2	Ideal gas "CH2I2" from NASA Glenn coefficients
<input type="checkbox"/> CH3	Ideal gas "CH3" from NASA Glenn coefficients
<input type="checkbox"/> CH3Br	Ideal gas "CH3Br" from NASA Glenn coefficients
<input type="checkbox"/> CH3CL	Ideal gas "CH3Cl" from NASA Glenn coefficients
<input type="checkbox"/> CH3F	Ideal gas "CH3F" from NASA Glenn coefficients
<input type="checkbox"/> CH3I	Ideal gas "CH3I" from NASA Glenn coefficients
<input type="checkbox"/> CH2OH	Ideal gas "CH2OH" from NASA Glenn coefficients
<input type="checkbox"/> CH2OHplus	Ideal gas "CH2OH+" from NASA Glenn coefficients
<input type="checkbox"/> CH3O	Ideal gas "CH3O" from NASA Glenn coefficients
<input type="checkbox"/> CH4	Ideal gas "CH4" from NASA Glenn coefficients
<input type="checkbox"/> CH3OH	Ideal gas "CH3OH" from NASA Glenn coefficients
<input type="checkbox"/> CH3OOH	Ideal gas "CH3OOH" from NASA Glenn coefficients
<input type="checkbox"/> CI	Ideal gas "CI" from NASA Glenn coefficients
<input type="checkbox"/> CI2	Ideal gas "CI2" from NASA Glenn coefficients
<input type="checkbox"/> CI3	Ideal gas "CI3" from NASA Glenn coefficients
<input type="checkbox"/> CI4	Ideal gas "CI4" from NASA Glenn coefficients
<input type="checkbox"/> CN	Ideal gas "CN" from NASA Glenn coefficients
<input type="checkbox"/> CNplus	Ideal gas "CN+" from NASA Glenn coefficients
<input type="checkbox"/> CNminus	Ideal gas "CN-" from NASA Glenn coefficients
<input type="checkbox"/> CNN	Ideal gas "CNN" from NASA Glenn coefficients
<input type="checkbox"/> CO	Ideal gas "CO" from NASA Glenn coefficients
<input type="checkbox"/> COplus	Ideal gas "CO+" from NASA Glenn coefficients
<input type="checkbox"/> COCL	Ideal gas "COCl" from NASA Glenn coefficients
<input type="checkbox"/> COCL2	Ideal gas "COCl2" from NASA Glenn coefficients
<input type="checkbox"/> COFCL	Ideal gas "COFCI" from NASA Glenn coefficients
<input type="checkbox"/> COF2	Ideal gas "COF2" from NASA Glenn coefficients
<input type="checkbox"/> COHCL	Ideal gas "COHCl" from NASA Glenn coefficients
<input type="checkbox"/> COHF	Ideal gas "COHF" from NASA Glenn coefficients

<input type="checkbox"/> COS	Ideal gas "COS" from NASA Glenn coefficients
<input type="checkbox"/> CO2	Ideal gas "CO2" from NASA Glenn coefficients
<input type="checkbox"/> CO2plus	Ideal gas "CO2+" from NASA Glenn coefficients
<input type="checkbox"/> COOH	Ideal gas "COOH" from NASA Glenn coefficients
<input type="checkbox"/> CP	Ideal gas "CP" from NASA Glenn coefficients
<input type="checkbox"/> CS	Ideal gas "CS" from NASA Glenn coefficients
<input type="checkbox"/> CS2	Ideal gas "CS2" from NASA Glenn coefficients
<input type="checkbox"/> C2	Ideal gas "C2" from NASA Glenn coefficients
<input type="checkbox"/> C2plus	Ideal gas "C2+" from NASA Glenn coefficients
<input type="checkbox"/> C2minus	Ideal gas "C2-" from NASA Glenn coefficients
<input type="checkbox"/> C2CL	Ideal gas "C2Cl" from NASA Glenn coefficients
<input type="checkbox"/> C2CL2	Ideal gas "C2Cl2" from NASA Glenn coefficients
<input type="checkbox"/> C2CL3	Ideal gas "C2Cl3" from NASA Glenn coefficients
<input type="checkbox"/> C2CL4	Ideal gas "C2Cl4" from NASA Glenn coefficients
<input type="checkbox"/> C2CL6	Ideal gas "C2Cl6" from NASA Glenn coefficients
<input type="checkbox"/> C2F	Ideal gas "C2F" from NASA Glenn coefficients
<input type="checkbox"/> C2FCL	Ideal gas "C2FCI" from NASA Glenn coefficients
<input type="checkbox"/> C2FCL3	Ideal gas "C2FCI3" from NASA Glenn coefficients
<input type="checkbox"/> C2F2	Ideal gas "C2F2" from NASA Glenn coefficients
<input type="checkbox"/> C2F2CL2	Ideal gas "C2F2Cl2" from NASA Glenn coefficients
<input type="checkbox"/> C2F3	Ideal gas "C2F3" from NASA Glenn coefficients
<input type="checkbox"/> C2F3CL	Ideal gas "C2F3Cl" from NASA Glenn coefficients
<input type="checkbox"/> C2F4	Ideal gas "C2F4" from NASA Glenn coefficients
<input type="checkbox"/> C2F6	Ideal gas "C2F6" from NASA Glenn coefficients
<input type="checkbox"/> C2H	Ideal gas "C2H" from NASA Glenn coefficients
<input type="checkbox"/> C2HCL	Ideal gas "C2HCl" from NASA Glenn coefficients
<input type="checkbox"/> C2HCL3	Ideal gas "C2HCl3" from NASA Glenn coefficients
<input type="checkbox"/> C2HF	Ideal gas "C2HF" from NASA Glenn coefficients
<input type="checkbox"/> C2HFCL2	Ideal gas "C2HFCI2" from NASA Glenn coefficients
<input type="checkbox"/> C2HF2CL	Ideal gas "C2HF2Cl" from NASA Glenn coefficients
<input type="checkbox"/> C2HF3	Ideal gas "C2HF3" from NASA Glenn coefficients
<input type="checkbox"/> C2H2_vinylidene	Ideal gas "C2H2_vinylidene" from NASA Glenn coefficients
<input type="checkbox"/> C2H2CL2	Ideal gas "C2H2Cl2" from NASA Glenn coefficients
<input type="checkbox"/> C2H2FCL	Ideal gas "C2H2FCI" from NASA Glenn coefficients
<input type="checkbox"/> C2H2F2	Ideal gas "C2H2F2" from NASA Glenn coefficients
<input type="checkbox"/> CH2CO_ketene	Ideal gas "CH2CO_ketene" from NASA Glenn coefficients
<input type="checkbox"/> O_CH_2O	Ideal gas "O_CH_2O" from NASA Glenn coefficients

<input type="checkbox"/> HO_CO_2OH	Ideal gas "HO_CO_2OH" from NASA Glenn coefficients
<input type="checkbox"/> C2H3_vinyl	Ideal gas "C2H3_vinyl" from NASA Glenn coefficients
<input type="checkbox"/> CH2BrminusCOOH	Ideal gas "CH2Br-COOH" from NASA Glenn coefficients
<input type="checkbox"/> C2H3CL	Ideal gas "C2H3Cl" from NASA Glenn coefficients
<input type="checkbox"/> CH2CLminusCOOH	Ideal gas "CH2Cl-COOH" from NASA Glenn coefficients
<input type="checkbox"/> C2H3F	Ideal gas "C2H3F" from NASA Glenn coefficients
<input type="checkbox"/> CH3CN	Ideal gas "CH3CN" from NASA Glenn coefficients
<input type="checkbox"/> CH3CO_acetyl	Ideal gas "CH3CO_acetyl" from NASA Glenn coefficients
<input type="checkbox"/> C2H4	Ideal gas "C2H4" from NASA Glenn coefficients
<input type="checkbox"/> C2H4O_ethylen_o	Ideal gas "C2H4O_ethylen_o" from NASA Glenn coefficients
<input type="checkbox"/> CH3CHO_ethanal	Ideal gas "CH3CHO_ethanal" from NASA Glenn coefficients
<input type="checkbox"/> CH3COOH	Ideal gas "CH3COOH" from NASA Glenn coefficients
<input type="checkbox"/> OHCH2COOH	Ideal gas "OHCH2COOH" from NASA Glenn coefficients
<input type="checkbox"/> C2H5	Ideal gas "C2H5" from NASA Glenn coefficients
<input type="checkbox"/> C2H5Br	Ideal gas "C2H5Br" from NASA Glenn coefficients
<input type="checkbox"/> C2H6	Ideal gas "C2H6" from NASA Glenn coefficients
<input type="checkbox"/> CH3N2CH3	Ideal gas "CH3N2CH3" from NASA Glenn coefficients
<input type="checkbox"/> C2H5OH	Ideal gas "C2H5OH" from NASA Glenn coefficients
<input type="checkbox"/> CH3OCH3	Ideal gas "CH3OCH3" from NASA Glenn coefficients
<input type="checkbox"/> CH3O2CH3	Ideal gas "CH3O2CH3" from NASA Glenn coefficients
<input type="checkbox"/> CCN	Ideal gas "CCN" from NASA Glenn coefficients
<input type="checkbox"/> CNC	Ideal gas "CNC" from NASA Glenn coefficients
<input type="checkbox"/> OCCN	Ideal gas "OCCN" from NASA Glenn coefficients
<input type="checkbox"/> C2N2	Ideal gas "C2N2" from NASA Glenn coefficients
<input type="checkbox"/> C2O	Ideal gas "C2O" from NASA Glenn coefficients
<input type="checkbox"/> C3	Ideal gas "C3" from NASA Glenn coefficients
<input type="checkbox"/> C3H3_1_propynl	Ideal gas "C3H3_1_propynl" from NASA Glenn coefficients
<input type="checkbox"/> C3H3_2_propynl	Ideal gas "C3H3_2_propynl" from NASA Glenn coefficients
<input type="checkbox"/> C3H4_allene	Ideal gas "C3H4_allene" from NASA Glenn coefficients
<input type="checkbox"/> C3H4_propyne	Ideal gas "C3H4_propyne" from NASA Glenn coefficients
<input type="checkbox"/> C3H4_cyclo	Ideal gas "C3H4_cyclo" from NASA Glenn coefficients
<input type="checkbox"/> C3H5_allyl	Ideal gas "C3H5_allyl" from NASA Glenn coefficients
<input type="checkbox"/> C3H6_propylene	Ideal gas "C3H6_propylene" from NASA Glenn coefficients
<input type="checkbox"/> C3H6_cyclo	Ideal gas "C3H6_cyclo" from NASA Glenn coefficients
<input type="checkbox"/> C3H6O_propylox	Ideal gas "C3H6O_propylox" from NASA Glenn coefficients
<input type="checkbox"/> C3H6O_acetone	Ideal gas "C3H6O_acetone" from NASA Glenn coefficients
<input type="checkbox"/> C3H6O_propanal	Ideal gas "C3H6O_propanal" from NASA Glenn coefficients

<input type="checkbox"/> C3H7_n_propyl	Ideal gas "C3H7_n_propyl" from NASA Glenn coefficients
<input type="checkbox"/> C3H7_i_propyl	Ideal gas "C3H7_i_propyl" from NASA Glenn coefficients
<input type="checkbox"/> C3H8	Ideal gas "C3H8" from NASA Glenn coefficients
<input type="checkbox"/> C3H8O_1propanol	Ideal gas "C3H8O_1propanol" from NASA Glenn coefficients
<input type="checkbox"/> C3H8O_2propanol	Ideal gas "C3H8O_2propanol" from NASA Glenn coefficients
<input type="checkbox"/> CNCOCN	Ideal gas "CNCOCN" from NASA Glenn coefficients
<input type="checkbox"/> C3O2	Ideal gas "C3O2" from NASA Glenn coefficients
<input type="checkbox"/> C4	Ideal gas "C4" from NASA Glenn coefficients
<input type="checkbox"/> C4H2_butadiyne	Ideal gas "C4H2_butadiyne" from NASA Glenn coefficients
<input type="checkbox"/> C4H4_1_3minuscyclo	Ideal gas "C4H4_1_3-cyclo" from NASA Glenn coefficients
<input type="checkbox"/> C4H6_butadiene	Ideal gas "C4H6_butadiene" from NASA Glenn coefficients
<input type="checkbox"/> C4H6_1butyne	Ideal gas "C4H6_1butyne" from NASA Glenn coefficients
<input type="checkbox"/> C4H6_2butyne	Ideal gas "C4H6_2butyne" from NASA Glenn coefficients
<input type="checkbox"/> C4H6_cyclo	Ideal gas "C4H6_cyclo" from NASA Glenn coefficients
<input type="checkbox"/> C4H8_1_butene	Ideal gas "C4H8_1_butene" from NASA Glenn coefficients
<input type="checkbox"/> C4H8_cis2_buten	Ideal gas "C4H8_cis2_buten" from NASA Glenn coefficients
<input type="checkbox"/> C4H8_isobutene	Ideal gas "C4H8_isobutene" from NASA Glenn coefficients
<input type="checkbox"/> C4H8_cyclo	Ideal gas "C4H8_cyclo" from NASA Glenn coefficients
<input type="checkbox"/> C4H9_n_butyl	Ideal gas "C4H9_n_butyl" from NASA Glenn coefficients
<input type="checkbox"/> C4H9_i_butyl	Ideal gas "C4H9_i_butyl" from NASA Glenn coefficients
<input type="checkbox"/> C4H9_s_butyl	Ideal gas "C4H9_s_butyl" from NASA Glenn coefficients
<input type="checkbox"/> C4H9_t_butyl	Ideal gas "C4H9_t_butyl" from NASA Glenn coefficients
<input type="checkbox"/> C4H10_n_butane	Ideal gas "C4H10_n_butane" from NASA Glenn coefficients
<input type="checkbox"/> C4H10_isobutane	Ideal gas "C4H10_isobutane" from NASA Glenn coefficients
<input type="checkbox"/> C4N2	Ideal gas "C4N2" from NASA Glenn coefficients
<input type="checkbox"/> C5	Ideal gas "C5" from NASA Glenn coefficients
<input type="checkbox"/> C5H6_1_3cyclo	Ideal gas "C5H6_1_3cyclo" from NASA Glenn coefficients
<input type="checkbox"/> C5H8_cyclo	Ideal gas "C5H8_cyclo" from NASA Glenn coefficients
<input type="checkbox"/> C5H10_1_pentene	Ideal gas "C5H10_1_pentene" from NASA Glenn coefficients
<input type="checkbox"/> C5H10_cyclo	Ideal gas "C5H10_cyclo" from NASA Glenn coefficients
<input type="checkbox"/> C5H11_pentyl	Ideal gas "C5H11_pentyl" from NASA Glenn coefficients
<input type="checkbox"/> C5H11_t_pentyl	Ideal gas "C5H11_t_pentyl" from NASA Glenn coefficients
<input type="checkbox"/> C5H12_n_pentane	Ideal gas "C5H12_n_pentane" from NASA Glenn coefficients
<input type="checkbox"/> C5H12_i_pentane	Ideal gas "C5H12_i_pentane" from NASA Glenn coefficients
<input type="checkbox"/> CH3C_CH3_2CH3	Ideal gas "CH3C_CH3_2CH3" from NASA Glenn coefficients
<input type="checkbox"/> C6D5_phenyl	Ideal gas "C6D5_phenyl" from NASA Glenn coefficients
<input type="checkbox"/> C6D6	Ideal gas "C6D6" from NASA Glenn coefficients

---

**1250 Modelica.Media.IdealGases.SingleGases**


---

<input type="checkbox"/> C6H2	Ideal gas "C6H2" from NASA Glenn coefficients
<input type="checkbox"/> C6H5_phenyl	Ideal gas "C6H5_phenyl" from NASA Glenn coefficients
<input type="checkbox"/> C6H5O_phenoxy	Ideal gas "C6H5O_phenoxy" from NASA Glenn coefficients
<input type="checkbox"/> C6H6	Ideal gas "C6H6" from NASA Glenn coefficients
<input type="checkbox"/> C6H5OH_phenol	Ideal gas "C6H5OH_phenol" from NASA Glenn coefficients
<input type="checkbox"/> C6H10_cyclo	Ideal gas "C6H10_cyclo" from NASA Glenn coefficients
<input type="checkbox"/> C6H12_1_hexene	Ideal gas "C6H12_1_hexene" from NASA Glenn coefficients
<input type="checkbox"/> C6H12_cyclo	Ideal gas "C6H12_cyclo" from NASA Glenn coefficients
<input type="checkbox"/> C6H13_n_hexyl	Ideal gas "C6H13_n_hexyl" from NASA Glenn coefficients
<input type="checkbox"/> C6H14_n_hexane	Ideal gas "C6H14_n_hexane" from NASA Glenn coefficients
<input type="checkbox"/> C7H7_benzyl	Ideal gas "C7H7_benzyl" from NASA Glenn coefficients
<input type="checkbox"/> C7H8	Ideal gas "C7H8" from NASA Glenn coefficients
<input type="checkbox"/> C7H8O_cresol_mx	Ideal gas "C7H8O_cresol_mx" from NASA Glenn coefficients
<input type="checkbox"/> C7H14_1_heptene	Ideal gas "C7H14_1_heptene" from NASA Glenn coefficients
<input type="checkbox"/> C7H15_n_heptyl	Ideal gas "C7H15_n_heptyl" from NASA Glenn coefficients
<input type="checkbox"/> C7H16_n_heptane	Ideal gas "C7H16_n_heptane" from NASA Glenn coefficients
<input type="checkbox"/> C7H16_2_methylh	Ideal gas "C7H16_2_methylh" from NASA Glenn coefficients
<input type="checkbox"/> C8H8_styrene	Ideal gas "C8H8_styrene" from NASA Glenn coefficients
<input type="checkbox"/> C8H10_ethylbenz	Ideal gas "C8H10_ethylbenz" from NASA Glenn coefficients
<input type="checkbox"/> C8H16_1_octene	Ideal gas "C8H16_1_octene" from NASA Glenn coefficients
<input type="checkbox"/> C8H17_n_octyl	Ideal gas "C8H17_n_octyl" from NASA Glenn coefficients
<input type="checkbox"/> C8H18_n_octane	Ideal gas "C8H18_n_octane" from NASA Glenn coefficients
<input type="checkbox"/> C8H18_isooctane	Ideal gas "C8H18_isooctane" from NASA Glenn coefficients
<input type="checkbox"/> C9H19_n_nonyl	Ideal gas "C9H19_n_nonyl" from NASA Glenn coefficients
<input type="checkbox"/> C10H8_naphthale	Ideal gas "C10H8_naphthale" from NASA Glenn coefficients
<input type="checkbox"/> C10H21_n_decyl	Ideal gas "C10H21_n_decyl" from NASA Glenn coefficients
<input type="checkbox"/> C12H9_o_bipheny	Ideal gas "C12H9_o_bipheny" from NASA Glenn coefficients
<input type="checkbox"/> C12H10_biphenyl	Ideal gas "C12H10_biphenyl" from NASA Glenn coefficients
<input type="checkbox"/> Ca	Ideal gas "Ca" from NASA Glenn coefficients
<input type="checkbox"/> Caplus	Ideal gas "Ca+" from NASA Glenn coefficients
<input type="checkbox"/> CaBr	Ideal gas "CaBr" from NASA Glenn coefficients
<input type="checkbox"/> CaBr2	Ideal gas "CaBr2" from NASA Glenn coefficients
<input type="checkbox"/> CaCL	Ideal gas "CaCl" from NASA Glenn coefficients
<input type="checkbox"/> CaCLplus	Ideal gas "CaCl+" from NASA Glenn coefficients
<input type="checkbox"/> CaCL2	Ideal gas "CaCl2" from NASA Glenn coefficients
<input type="checkbox"/> CaF	Ideal gas "CaF" from NASA Glenn coefficients
<input type="checkbox"/> CaFplus	Ideal gas "CaF+" from NASA Glenn coefficients

<input type="checkbox"/> CaF2	Ideal gas "CaF2" from NASA Glenn coefficients
<input type="checkbox"/> CaH	Ideal gas "CaH" from NASA Glenn coefficients
<input type="checkbox"/> CaI	Ideal gas "CaI" from NASA Glenn coefficients
<input type="checkbox"/> CaI2	Ideal gas "CaI2" from NASA Glenn coefficients
<input type="checkbox"/> CaO	Ideal gas "CaO" from NASA Glenn coefficients
<input type="checkbox"/> CaOplus	Ideal gas "CaO+" from NASA Glenn coefficients
<input type="checkbox"/> CaOH	Ideal gas "CaOH" from NASA Glenn coefficients
<input type="checkbox"/> CaOHplus	Ideal gas "CaOH+" from NASA Glenn coefficients
<input type="checkbox"/> Ca_OH_2	Ideal gas "Ca_OH_2" from NASA Glenn coefficients
<input type="checkbox"/> CaS	Ideal gas "CaS" from NASA Glenn coefficients
<input type="checkbox"/> Ca2	Ideal gas "Ca2" from NASA Glenn coefficients
<input type="checkbox"/> Cd	Ideal gas "Cd" from NASA Glenn coefficients
<input type="checkbox"/> Cdplus	Ideal gas "Cd+" from NASA Glenn coefficients
<input type="checkbox"/> Cl	Ideal gas "Cl" from NASA Glenn coefficients
<input type="checkbox"/> Clplus	Ideal gas "Cl+" from NASA Glenn coefficients
<input type="checkbox"/> CLminus	Ideal gas "Cl-" from NASA Glenn coefficients
<input type="checkbox"/> CLCN	Ideal gas "CICN" from NASA Glenn coefficients
<input type="checkbox"/> CLF	Ideal gas "CIF" from NASA Glenn coefficients
<input type="checkbox"/> CLF3	Ideal gas "CIF3" from NASA Glenn coefficients
<input type="checkbox"/> CLF5	Ideal gas "CIF5" from NASA Glenn coefficients
<input type="checkbox"/> CLO	Ideal gas "CIO" from NASA Glenn coefficients
<input type="checkbox"/> CLO2	Ideal gas "CIO2" from NASA Glenn coefficients
<input type="checkbox"/> CL2	Ideal gas "Cl2" from NASA Glenn coefficients
<input type="checkbox"/> CL2O	Ideal gas "Cl2O" from NASA Glenn coefficients
<input type="checkbox"/> Co	Ideal gas "Co" from NASA Glenn coefficients
<input type="checkbox"/> Coplus	Ideal gas "Co+" from NASA Glenn coefficients
<input type="checkbox"/> Cominus	Ideal gas "Co-" from NASA Glenn coefficients
<input type="checkbox"/> Cr	Ideal gas "Cr" from NASA Glenn coefficients
<input type="checkbox"/> Crplus	Ideal gas "Cr+" from NASA Glenn coefficients
<input type="checkbox"/> Crminus	Ideal gas "Cr-" from NASA Glenn coefficients
<input type="checkbox"/> CrN	Ideal gas "CrN" from NASA Glenn coefficients
<input type="checkbox"/> CrO	Ideal gas "CrO" from NASA Glenn coefficients
<input type="checkbox"/> CrO2	Ideal gas "CrO2" from NASA Glenn coefficients
<input type="checkbox"/> CrO3	Ideal gas "CrO3" from NASA Glenn coefficients
<input type="checkbox"/> CrO3minus	Ideal gas "CrO3-" from NASA Glenn coefficients
<input type="checkbox"/> Cs	Ideal gas "Cs" from NASA Glenn coefficients
<input type="checkbox"/> Csplus	Ideal gas "Cs+" from NASA Glenn coefficients

---

**1252 Modelica.Media.IdealGases.SingleGases**


---

<input type="checkbox"/> Csminus	Ideal gas "Cs-" from NASA Glenn coefficients
<input type="checkbox"/> CsBO2	Ideal gas "CsBO2" from NASA Glenn coefficients
<input type="checkbox"/> CsBr	Ideal gas "CsBr" from NASA Glenn coefficients
<input type="checkbox"/> CsCL	Ideal gas "CsCl" from NASA Glenn coefficients
<input type="checkbox"/> CsF	Ideal gas "CsF" from NASA Glenn coefficients
<input type="checkbox"/> CsH	Ideal gas "CsH" from NASA Glenn coefficients
<input type="checkbox"/> CsI	Ideal gas "CsI" from NASA Glenn coefficients
<input type="checkbox"/> CsLi	Ideal gas "CsLi" from NASA Glenn coefficients
<input type="checkbox"/> CsNO2	Ideal gas "CsNO2" from NASA Glenn coefficients
<input type="checkbox"/> CsNO3	Ideal gas "CsNO3" from NASA Glenn coefficients
<input type="checkbox"/> CsNa	Ideal gas "CsNa" from NASA Glenn coefficients
<input type="checkbox"/> CsO	Ideal gas "CsO" from NASA Glenn coefficients
<input type="checkbox"/> CsOH	Ideal gas "CsOH" from NASA Glenn coefficients
<input type="checkbox"/> CsRb	Ideal gas "CsRb" from NASA Glenn coefficients
<input type="checkbox"/> Cs2	Ideal gas "Cs2" from NASA Glenn coefficients
<input type="checkbox"/> Cs2Br2	Ideal gas "Cs2Br2" from NASA Glenn coefficients
<input type="checkbox"/> Cs2CO3	Ideal gas "Cs2CO3" from NASA Glenn coefficients
<input type="checkbox"/> Cs2CL2	Ideal gas "Cs2Cl2" from NASA Glenn coefficients
<input type="checkbox"/> Cs2F2	Ideal gas "Cs2F2" from NASA Glenn coefficients
<input type="checkbox"/> Cs2I2	Ideal gas "Cs2I2" from NASA Glenn coefficients
<input type="checkbox"/> Cs2O	Ideal gas "Cs2O" from NASA Glenn coefficients
<input type="checkbox"/> Cs2Oplus	Ideal gas "Cs2O+" from NASA Glenn coefficients
<input type="checkbox"/> Cs2O2	Ideal gas "Cs2O2" from NASA Glenn coefficients
<input type="checkbox"/> Cs2O2H2	Ideal gas "Cs2O2H2" from NASA Glenn coefficients
<input type="checkbox"/> Cs2SO4	Ideal gas "Cs2SO4" from NASA Glenn coefficients
<input type="checkbox"/> Cu	Ideal gas "Cu" from NASA Glenn coefficients
<input type="checkbox"/> Cuplus	Ideal gas "Cu+" from NASA Glenn coefficients
<input type="checkbox"/> Cuminus	Ideal gas "Cu-" from NASA Glenn coefficients
<input type="checkbox"/> CuCL	Ideal gas "CuCl" from NASA Glenn coefficients
<input type="checkbox"/> CuF	Ideal gas "CuF" from NASA Glenn coefficients
<input type="checkbox"/> CuF2	Ideal gas "CuF2" from NASA Glenn coefficients
<input type="checkbox"/> CuO	Ideal gas "CuO" from NASA Glenn coefficients
<input type="checkbox"/> Cu2	Ideal gas "Cu2" from NASA Glenn coefficients
<input type="checkbox"/> Cu3CL3	Ideal gas "Cu3Cl3" from NASA Glenn coefficients
<input type="checkbox"/> D	Ideal gas "D" from NASA Glenn coefficients
<input type="checkbox"/> Dplus	Ideal gas "D+" from NASA Glenn coefficients
<input type="checkbox"/> Dminus	Ideal gas "D-" from NASA Glenn coefficients

<input type="checkbox"/> DBr	Ideal gas "DBr" from NASA Glenn coefficients
<input type="checkbox"/> DCL	Ideal gas "DCl" from NASA Glenn coefficients
<input type="checkbox"/> DF	Ideal gas "DF" from NASA Glenn coefficients
<input type="checkbox"/> DOCL	Ideal gas "DOCl" from NASA Glenn coefficients
<input type="checkbox"/> DO2	Ideal gas "DO2" from NASA Glenn coefficients
<input type="checkbox"/> DO2minus	Ideal gas "DO2-" from NASA Glenn coefficients
<input type="checkbox"/> D2	Ideal gas "D2" from NASA Glenn coefficients
<input type="checkbox"/> D2plus	Ideal gas "D2+" from NASA Glenn coefficients
<input type="checkbox"/> D2minus	Ideal gas "D2-" from NASA Glenn coefficients
<input type="checkbox"/> D2O	Ideal gas "D2O" from NASA Glenn coefficients
<input type="checkbox"/> D2O2	Ideal gas "D2O2" from NASA Glenn coefficients
<input type="checkbox"/> D2S	Ideal gas "D2S" from NASA Glenn coefficients
<input type="checkbox"/> eminus	Ideal gas "e-" from NASA Glenn coefficients
<input type="checkbox"/> F	Ideal gas "F" from NASA Glenn coefficients
<input type="checkbox"/> Fplus	Ideal gas "F+" from NASA Glenn coefficients
<input type="checkbox"/> Fminus	Ideal gas "F-" from NASA Glenn coefficients
<input type="checkbox"/> FCN	Ideal gas "FCN" from NASA Glenn coefficients
<input type="checkbox"/> FCO	Ideal gas "FCO" from NASA Glenn coefficients
<input type="checkbox"/> FO	Ideal gas "FO" from NASA Glenn coefficients
<input type="checkbox"/> FO2_FOO	Ideal gas "FO2_FOO" from NASA Glenn coefficients
<input type="checkbox"/> FO2_OFO	Ideal gas "FO2_OFO" from NASA Glenn coefficients
<input type="checkbox"/> F2	Ideal gas "F2" from NASA Glenn coefficients
<input type="checkbox"/> F2O	Ideal gas "F2O" from NASA Glenn coefficients
<input type="checkbox"/> F2O2	Ideal gas "F2O2" from NASA Glenn coefficients
<input type="checkbox"/> FS2F	Ideal gas "FS2F" from NASA Glenn coefficients
<input type="checkbox"/> Fe	Ideal gas "Fe" from NASA Glenn coefficients
<input type="checkbox"/> Feplus	Ideal gas "Fe+" from NASA Glenn coefficients
<input type="checkbox"/> Fe_CO_5	Ideal gas "Fe_CO_5" from NASA Glenn coefficients
<input type="checkbox"/> FeCL	Ideal gas "FeCl" from NASA Glenn coefficients
<input type="checkbox"/> FeCL2	Ideal gas "FeCl2" from NASA Glenn coefficients
<input type="checkbox"/> FeCL3	Ideal gas "FeCl3" from NASA Glenn coefficients
<input type="checkbox"/> FeO	Ideal gas "FeO" from NASA Glenn coefficients
<input type="checkbox"/> Fe_OH_2	Ideal gas "Fe_OH_2" from NASA Glenn coefficients
<input type="checkbox"/> Fe2CL4	Ideal gas "Fe2Cl4" from NASA Glenn coefficients
<input type="checkbox"/> Fe2CL6	Ideal gas "Fe2Cl6" from NASA Glenn coefficients
<input type="checkbox"/> Ga	Ideal gas "Ga" from NASA Glenn coefficients
<input type="checkbox"/> Gaplus	Ideal gas "Ga+" from NASA Glenn coefficients

<a href="#">GaBr</a>	Ideal gas "GaBr" from NASA Glenn coefficients
<a href="#">GaBr2</a>	Ideal gas "GaBr2" from NASA Glenn coefficients
<a href="#">GaBr3</a>	Ideal gas "GaBr3" from NASA Glenn coefficients
<a href="#">GaCL</a>	Ideal gas "GaCl" from NASA Glenn coefficients
<a href="#">GaCL2</a>	Ideal gas "GaCl2" from NASA Glenn coefficients
<a href="#">GaCL3</a>	Ideal gas "GaCl3" from NASA Glenn coefficients
<a href="#">GaF</a>	Ideal gas "GaF" from NASA Glenn coefficients
<a href="#">GaF2</a>	Ideal gas "GaF2" from NASA Glenn coefficients
<a href="#">GaF3</a>	Ideal gas "GaF3" from NASA Glenn coefficients
<a href="#">GaH</a>	Ideal gas "GaH" from NASA Glenn coefficients
<a href="#">Gal</a>	Ideal gas "Gal" from NASA Glenn coefficients
<a href="#">Gal2</a>	Ideal gas "Gal2" from NASA Glenn coefficients
<a href="#">Gal3</a>	Ideal gas "Gal3" from NASA Glenn coefficients
<a href="#">GaO</a>	Ideal gas "GaO" from NASA Glenn coefficients
<a href="#">GaOH</a>	Ideal gas "GaOH" from NASA Glenn coefficients
<a href="#">Ga2Br2</a>	Ideal gas "Ga2Br2" from NASA Glenn coefficients
<a href="#">Ga2Br4</a>	Ideal gas "Ga2Br4" from NASA Glenn coefficients
<a href="#">Ga2Br6</a>	Ideal gas "Ga2Br6" from NASA Glenn coefficients
<a href="#">Ga2CL2</a>	Ideal gas "Ga2Cl2" from NASA Glenn coefficients
<a href="#">Ga2CL4</a>	Ideal gas "Ga2Cl4" from NASA Glenn coefficients
<a href="#">Ga2CL6</a>	Ideal gas "Ga2Cl6" from NASA Glenn coefficients
<a href="#">Ga2F2</a>	Ideal gas "Ga2F2" from NASA Glenn coefficients
<a href="#">Ga2F4</a>	Ideal gas "Ga2F4" from NASA Glenn coefficients
<a href="#">Ga2F6</a>	Ideal gas "Ga2F6" from NASA Glenn coefficients
<a href="#">Ga2I2</a>	Ideal gas "Ga2I2" from NASA Glenn coefficients
<a href="#">Ga2I4</a>	Ideal gas "Ga2I4" from NASA Glenn coefficients
<a href="#">Ga2I6</a>	Ideal gas "Ga2I6" from NASA Glenn coefficients
<a href="#">Ga2O</a>	Ideal gas "Ga2O" from NASA Glenn coefficients
<a href="#">Ge</a>	Ideal gas "Ge" from NASA Glenn coefficients
<a href="#">Geplus</a>	Ideal gas "Ge+" from NASA Glenn coefficients
<a href="#">Geminus</a>	Ideal gas "Ge-" from NASA Glenn coefficients
<a href="#">GeBr</a>	Ideal gas "GeBr" from NASA Glenn coefficients
<a href="#">GeBr2</a>	Ideal gas "GeBr2" from NASA Glenn coefficients
<a href="#">GeBr3</a>	Ideal gas "GeBr3" from NASA Glenn coefficients
<a href="#">GeBr4</a>	Ideal gas "GeBr4" from NASA Glenn coefficients
<a href="#">GeCL</a>	Ideal gas "GeCl" from NASA Glenn coefficients
<a href="#">GeCL2</a>	Ideal gas "GeCl2" from NASA Glenn coefficients

<input type="checkbox"/> GeCl3	Ideal gas "GeCl3" from NASA Glenn coefficients
<input type="checkbox"/> GeCl4	Ideal gas "GeCl4" from NASA Glenn coefficients
<input type="checkbox"/> GeF	Ideal gas "GeF" from NASA Glenn coefficients
<input type="checkbox"/> GeF2	Ideal gas "GeF2" from NASA Glenn coefficients
<input type="checkbox"/> GeF3	Ideal gas "GeF3" from NASA Glenn coefficients
<input type="checkbox"/> GeF4	Ideal gas "GeF4" from NASA Glenn coefficients
<input type="checkbox"/> GeH4	Ideal gas "GeH4" from NASA Glenn coefficients
<input type="checkbox"/> Gel	Ideal gas "Gel" from NASA Glenn coefficients
<input type="checkbox"/> GeO	Ideal gas "GeO" from NASA Glenn coefficients
<input type="checkbox"/> GeO2	Ideal gas "GeO2" from NASA Glenn coefficients
<input type="checkbox"/> GeS	Ideal gas "GeS" from NASA Glenn coefficients
<input type="checkbox"/> GeS2	Ideal gas "GeS2" from NASA Glenn coefficients
<input type="checkbox"/> Ge2	Ideal gas "Ge2" from NASA Glenn coefficients
<input type="checkbox"/> H	Ideal gas "H" from NASA Glenn coefficients
<input type="checkbox"/> Hplus	Ideal gas "H+" from NASA Glenn coefficients
<input type="checkbox"/> Hminus	Ideal gas "H-" from NASA Glenn coefficients
<input type="checkbox"/> HALO	Ideal gas "HAIO" from NASA Glenn coefficients
<input type="checkbox"/> HALO2	Ideal gas "HAIO2" from NASA Glenn coefficients
<input type="checkbox"/> HBO	Ideal gas "HBO" from NASA Glenn coefficients
<input type="checkbox"/> HBOplus	Ideal gas "HBO+" from NASA Glenn coefficients
<input type="checkbox"/> HBO2	Ideal gas "HBO2" from NASA Glenn coefficients
<input type="checkbox"/> HBS	Ideal gas "HBS" from NASA Glenn coefficients
<input type="checkbox"/> HBSplus	Ideal gas "HBS+" from NASA Glenn coefficients
<input type="checkbox"/> HCN	Ideal gas "HCN" from NASA Glenn coefficients
<input type="checkbox"/> HCO	Ideal gas "HCO" from NASA Glenn coefficients
<input type="checkbox"/> HCOplus	Ideal gas "HCO+" from NASA Glenn coefficients
<input type="checkbox"/> HCCN	Ideal gas "HCCN" from NASA Glenn coefficients
<input type="checkbox"/> HCCO	Ideal gas "HCCO" from NASA Glenn coefficients
<input type="checkbox"/> HCL	Ideal gas "HCl" from NASA Glenn coefficients
<input type="checkbox"/> HD	Ideal gas "HD" from NASA Glenn coefficients
<input type="checkbox"/> HDplus	Ideal gas "HD+" from NASA Glenn coefficients
<input type="checkbox"/> HDO	Ideal gas "HDO" from NASA Glenn coefficients
<input type="checkbox"/> HDO2	Ideal gas "HDO2" from NASA Glenn coefficients
<input type="checkbox"/> HF	Ideal gas "HF" from NASA Glenn coefficients
<input type="checkbox"/> HI	Ideal gas "HI" from NASA Glenn coefficients
<input type="checkbox"/> HNC	Ideal gas "HNC" from NASA Glenn coefficients
<input type="checkbox"/> HNCO	Ideal gas "HNCO" from NASA Glenn coefficients

---

**1256 Modelica.Media.IdealGases.SingleGases**


---

<input type="checkbox"/> HNO	Ideal gas "HNO" from NASA Glenn coefficients
<input type="checkbox"/> HNO2	Ideal gas "HNO2" from NASA Glenn coefficients
<input type="checkbox"/> HNO3	Ideal gas "HNO3" from NASA Glenn coefficients
<input type="checkbox"/> HOCL	Ideal gas "HOCl" from NASA Glenn coefficients
<input type="checkbox"/> HOF	Ideal gas "HOF" from NASA Glenn coefficients
<input type="checkbox"/> HO2	Ideal gas "HO2" from NASA Glenn coefficients
<input type="checkbox"/> HO2minus	Ideal gas "HO2-" from NASA Glenn coefficients
<input type="checkbox"/> HPO	Ideal gas "HPO" from NASA Glenn coefficients
<input type="checkbox"/> HSO3F	Ideal gas "HSO3F" from NASA Glenn coefficients
<input type="checkbox"/> H2	Ideal gas "H2" from NASA Glenn coefficients
<input type="checkbox"/> H2plus	Ideal gas "H2+" from NASA Glenn coefficients
<input type="checkbox"/> H2minus	Ideal gas "H2-" from NASA Glenn coefficients
<input type="checkbox"/> HBOH	Ideal gas "HBOH" from NASA Glenn coefficients
<input type="checkbox"/> HCOOH	Ideal gas "HCOOH" from NASA Glenn coefficients
<input type="checkbox"/> H2F2	Ideal gas "H2F2" from NASA Glenn coefficients
<input type="checkbox"/> H2O	Ideal gas "H2O" from NASA Glenn coefficients
<input type="checkbox"/> H2Oplus	Ideal gas "H2O+" from NASA Glenn coefficients
<input type="checkbox"/> H2O2	Ideal gas "H2O2" from NASA Glenn coefficients
<input type="checkbox"/> H2S	Ideal gas "H2S" from NASA Glenn coefficients
<input type="checkbox"/> H2SO4	Ideal gas "H2SO4" from NASA Glenn coefficients
<input type="checkbox"/> H2BOH	Ideal gas "H2BOH" from NASA Glenn coefficients
<input type="checkbox"/> HB_OH_2	Ideal gas "HB_OH_2" from NASA Glenn coefficients
<input type="checkbox"/> H3BO3	Ideal gas "H3BO3" from NASA Glenn coefficients
<input type="checkbox"/> H3B3O3	Ideal gas "H3B3O3" from NASA Glenn coefficients
<input type="checkbox"/> H3B3O6	Ideal gas "H3B3O6" from NASA Glenn coefficients
<input type="checkbox"/> H3F3	Ideal gas "H3F3" from NASA Glenn coefficients
<input type="checkbox"/> H3Oplus	Ideal gas "H3O+" from NASA Glenn coefficients
<input type="checkbox"/> H4F4	Ideal gas "H4F4" from NASA Glenn coefficients
<input type="checkbox"/> H5F5	Ideal gas "H5F5" from NASA Glenn coefficients
<input type="checkbox"/> H6F6	Ideal gas "H6F6" from NASA Glenn coefficients
<input type="checkbox"/> H7F7	Ideal gas "H7F7" from NASA Glenn coefficients
<input type="checkbox"/> He	Ideal gas "He" from NASA Glenn coefficients
<input type="checkbox"/> Heplus	Ideal gas "He+" from NASA Glenn coefficients
<input type="checkbox"/> Hg	Ideal gas "Hg" from NASA Glenn coefficients
<input type="checkbox"/> Hgplus	Ideal gas "Hg+" from NASA Glenn coefficients
<input type="checkbox"/> HgBr2	Ideal gas "HgBr2" from NASA Glenn coefficients
<input type="checkbox"/> I	Ideal gas "I" from NASA Glenn coefficients

<input type="checkbox"/> Iplus	Ideal gas "I+" from NASA Glenn coefficients
<input type="checkbox"/> Iminus	Ideal gas "I-" from NASA Glenn coefficients
<input type="checkbox"/> IF5	Ideal gas "IF5" from NASA Glenn coefficients
<input type="checkbox"/> IF7	Ideal gas "IF7" from NASA Glenn coefficients
<input type="checkbox"/> I2	Ideal gas "I2" from NASA Glenn coefficients
<input type="checkbox"/> In	Ideal gas "In" from NASA Glenn coefficients
<input type="checkbox"/> Inplus	Ideal gas "In+" from NASA Glenn coefficients
<input type="checkbox"/> InBr	Ideal gas "InBr" from NASA Glenn coefficients
<input type="checkbox"/> InBr2	Ideal gas "InBr2" from NASA Glenn coefficients
<input type="checkbox"/> InBr3	Ideal gas "InBr3" from NASA Glenn coefficients
<input type="checkbox"/> InCL	Ideal gas "InCl" from NASA Glenn coefficients
<input type="checkbox"/> InCL2	Ideal gas "InCl2" from NASA Glenn coefficients
<input type="checkbox"/> InCL3	Ideal gas "InCl3" from NASA Glenn coefficients
<input type="checkbox"/> InF	Ideal gas "InF" from NASA Glenn coefficients
<input type="checkbox"/> InF2	Ideal gas "InF2" from NASA Glenn coefficients
<input type="checkbox"/> InF3	Ideal gas "InF3" from NASA Glenn coefficients
<input type="checkbox"/> InH	Ideal gas "InH" from NASA Glenn coefficients
<input type="checkbox"/> InI	Ideal gas "InI" from NASA Glenn coefficients
<input type="checkbox"/> InI2	Ideal gas "InI2" from NASA Glenn coefficients
<input type="checkbox"/> InI3	Ideal gas "InI3" from NASA Glenn coefficients
<input type="checkbox"/> InO	Ideal gas "InO" from NASA Glenn coefficients
<input type="checkbox"/> InOH	Ideal gas "InOH" from NASA Glenn coefficients
<input type="checkbox"/> In2Br2	Ideal gas "In2Br2" from NASA Glenn coefficients
<input type="checkbox"/> In2Br4	Ideal gas "In2Br4" from NASA Glenn coefficients
<input type="checkbox"/> In2Br6	Ideal gas "In2Br6" from NASA Glenn coefficients
<input type="checkbox"/> In2CL2	Ideal gas "In2Cl2" from NASA Glenn coefficients
<input type="checkbox"/> In2CL4	Ideal gas "In2Cl4" from NASA Glenn coefficients
<input type="checkbox"/> In2CL6	Ideal gas "In2Cl6" from NASA Glenn coefficients
<input type="checkbox"/> In2F2	Ideal gas "In2F2" from NASA Glenn coefficients
<input type="checkbox"/> In2F4	Ideal gas "In2F4" from NASA Glenn coefficients
<input type="checkbox"/> In2F6	Ideal gas "In2F6" from NASA Glenn coefficients
<input type="checkbox"/> In2I2	Ideal gas "In2I2" from NASA Glenn coefficients
<input type="checkbox"/> In2I4	Ideal gas "In2I4" from NASA Glenn coefficients
<input type="checkbox"/> In2I6	Ideal gas "In2I6" from NASA Glenn coefficients
<input type="checkbox"/> In2O	Ideal gas "In2O" from NASA Glenn coefficients
<input type="checkbox"/> K	Ideal gas "K" from NASA Glenn coefficients
<input type="checkbox"/> Kplus	Ideal gas "K+" from NASA Glenn coefficients

---

**1258 Modelica.Media.IdealGases.SingleGases**


---

<input type="checkbox"/> Kminus	Ideal gas "K-" from NASA Glenn coefficients
<input type="checkbox"/> KALF4	Ideal gas "KAIF4" from NASA Glenn coefficients
<input type="checkbox"/> KBO2	Ideal gas "KBO2" from NASA Glenn coefficients
<input type="checkbox"/> KBr	Ideal gas "KBr" from NASA Glenn coefficients
<input type="checkbox"/> KCN	Ideal gas "KCN" from NASA Glenn coefficients
<input type="checkbox"/> KCL	Ideal gas "KCl" from NASA Glenn coefficients
<input type="checkbox"/> KF	Ideal gas "KF" from NASA Glenn coefficients
<input type="checkbox"/> KH	Ideal gas "KH" from NASA Glenn coefficients
<input type="checkbox"/> KI	Ideal gas "KI" from NASA Glenn coefficients
<input type="checkbox"/> KLi	Ideal gas "KLi" from NASA Glenn coefficients
<input type="checkbox"/> KNO2	Ideal gas "KNO2" from NASA Glenn coefficients
<input type="checkbox"/> KNO3	Ideal gas "KNO3" from NASA Glenn coefficients
<input type="checkbox"/> KNa	Ideal gas "KNa" from NASA Glenn coefficients
<input type="checkbox"/> KO	Ideal gas "KO" from NASA Glenn coefficients
<input type="checkbox"/> KOH	Ideal gas "KOH" from NASA Glenn coefficients
<input type="checkbox"/> K2	Ideal gas "K2" from NASA Glenn coefficients
<input type="checkbox"/> K2plus	Ideal gas "K2+" from NASA Glenn coefficients
<input type="checkbox"/> K2Br2	Ideal gas "K2Br2" from NASA Glenn coefficients
<input type="checkbox"/> K2CO3	Ideal gas "K2CO3" from NASA Glenn coefficients
<input type="checkbox"/> K2C2N2	Ideal gas "K2C2N2" from NASA Glenn coefficients
<input type="checkbox"/> K2Cl2	Ideal gas "K2Cl2" from NASA Glenn coefficients
<input type="checkbox"/> K2F2	Ideal gas "K2F2" from NASA Glenn coefficients
<input type="checkbox"/> K2I2	Ideal gas "K2I2" from NASA Glenn coefficients
<input type="checkbox"/> K2O	Ideal gas "K2O" from NASA Glenn coefficients
<input type="checkbox"/> K2Oplus	Ideal gas "K2O+" from NASA Glenn coefficients
<input type="checkbox"/> K2O2	Ideal gas "K2O2" from NASA Glenn coefficients
<input type="checkbox"/> K2O2H2	Ideal gas "K2O2H2" from NASA Glenn coefficients
<input type="checkbox"/> K2SO4	Ideal gas "K2SO4" from NASA Glenn coefficients
<input type="checkbox"/> Kr	Ideal gas "Kr" from NASA Glenn coefficients
<input type="checkbox"/> Krplus	Ideal gas "Kr+" from NASA Glenn coefficients
<input type="checkbox"/> Li	Ideal gas "Li" from NASA Glenn coefficients
<input type="checkbox"/> Liplus	Ideal gas "Li+" from NASA Glenn coefficients
<input type="checkbox"/> Liminus	Ideal gas "Li-" from NASA Glenn coefficients
<input type="checkbox"/> LiAlF4	Ideal gas "LiAlF4" from NASA Glenn coefficients
<input type="checkbox"/> LiBO2	Ideal gas "LiBO2" from NASA Glenn coefficients
<input type="checkbox"/> LiBr	Ideal gas "LiBr" from NASA Glenn coefficients
<input type="checkbox"/> LiCl	Ideal gas "LiCl" from NASA Glenn coefficients

<input type="checkbox"/> LiF	Ideal gas "liF" from NASA Glenn coefficients
<input type="checkbox"/> LiH	Ideal gas "liH" from NASA Glenn coefficients
<input type="checkbox"/> LiI	Ideal gas "liI" from NASA Glenn coefficients
<input type="checkbox"/> LiN	Ideal gas "liN" from NASA Glenn coefficients
<input type="checkbox"/> LiNO2	Ideal gas "liNO2" from NASA Glenn coefficients
<input type="checkbox"/> LiNO3	Ideal gas "liNO3" from NASA Glenn coefficients
<input type="checkbox"/> LiO	Ideal gas "liO" from NASA Glenn coefficients
<input type="checkbox"/> LiOF	Ideal gas "liOF" from NASA Glenn coefficients
<input type="checkbox"/> LiOH	Ideal gas "liOH" from NASA Glenn coefficients
<input type="checkbox"/> LiON	Ideal gas "liON" from NASA Glenn coefficients
<input type="checkbox"/> Li2	Ideal gas "li2" from NASA Glenn coefficients
<input type="checkbox"/> Li2plus	Ideal gas "li2+" from NASA Glenn coefficients
<input type="checkbox"/> Li2Br2	Ideal gas "li2Br2" from NASA Glenn coefficients
<input type="checkbox"/> Li2F2	Ideal gas "li2F2" from NASA Glenn coefficients
<input type="checkbox"/> Li2I2	Ideal gas "li2I2" from NASA Glenn coefficients
<input type="checkbox"/> Li2O	Ideal gas "li2O" from NASA Glenn coefficients
<input type="checkbox"/> Li2Oplus	Ideal gas "li2O+" from NASA Glenn coefficients
<input type="checkbox"/> Li2O2	Ideal gas "li2O2" from NASA Glenn coefficients
<input type="checkbox"/> Li2O2H2	Ideal gas "li2O2H2" from NASA Glenn coefficients
<input type="checkbox"/> Li2SO4	Ideal gas "li2SO4" from NASA Glenn coefficients
<input type="checkbox"/> Li3plus	Ideal gas "li3+" from NASA Glenn coefficients
<input type="checkbox"/> Li3Br3	Ideal gas "li3Br3" from NASA Glenn coefficients
<input type="checkbox"/> Li3CL3	Ideal gas "li3Cl3" from NASA Glenn coefficients
<input type="checkbox"/> Li3F3	Ideal gas "li3F3" from NASA Glenn coefficients
<input type="checkbox"/> Li3I3	Ideal gas "li3I3" from NASA Glenn coefficients
<input type="checkbox"/> Mg	Ideal gas "Mg" from NASA Glenn coefficients
<input type="checkbox"/> Mgplus	Ideal gas "Mg+" from NASA Glenn coefficients
<input type="checkbox"/> MgBr	Ideal gas "MgBr" from NASA Glenn coefficients
<input type="checkbox"/> MgBr2	Ideal gas "MgBr2" from NASA Glenn coefficients
<input type="checkbox"/> MgCL	Ideal gas "MgCl" from NASA Glenn coefficients
<input type="checkbox"/> MgCLplus	Ideal gas "MgCl+" from NASA Glenn coefficients
<input type="checkbox"/> MgCL2	Ideal gas "MgCl2" from NASA Glenn coefficients
<input type="checkbox"/> MgF	Ideal gas "MgF" from NASA Glenn coefficients
<input type="checkbox"/> MgFplus	Ideal gas "MgF+" from NASA Glenn coefficients
<input type="checkbox"/> MgF2	Ideal gas "MgF2" from NASA Glenn coefficients
<input type="checkbox"/> MgF2plus	Ideal gas "MgF2+" from NASA Glenn coefficients
<input type="checkbox"/> MgH	Ideal gas "MgH" from NASA Glenn coefficients

<input type="checkbox"/> MgI	Ideal gas "MgI" from NASA Glenn coefficients
<input type="checkbox"/> MgI2	Ideal gas "MgI2" from NASA Glenn coefficients
<input type="checkbox"/> MgN	Ideal gas "MgN" from NASA Glenn coefficients
<input type="checkbox"/> MgO	Ideal gas "MgO" from NASA Glenn coefficients
<input type="checkbox"/> MgOH	Ideal gas "MgOH" from NASA Glenn coefficients
<input type="checkbox"/> MgOHplus	Ideal gas "MgOH+" from NASA Glenn coefficients
<input type="checkbox"/> Mg_OH_2	Ideal gas "Mg_OH_2" from NASA Glenn coefficients
<input type="checkbox"/> MgS	Ideal gas "MgS" from NASA Glenn coefficients
<input type="checkbox"/> Mg2	Ideal gas "Mg2" from NASA Glenn coefficients
<input type="checkbox"/> Mg2F4	Ideal gas "Mg2F4" from NASA Glenn coefficients
<input type="checkbox"/> Mn	Ideal gas "Mn" from NASA Glenn coefficients
<input type="checkbox"/> Mnplus	Ideal gas "Mn+" from NASA Glenn coefficients
<input type="checkbox"/> Mo	Ideal gas "Mo" from NASA Glenn coefficients
<input type="checkbox"/> Moplus	Ideal gas "Mo+" from NASA Glenn coefficients
<input type="checkbox"/> Mominus	Ideal gas "Mo-" from NASA Glenn coefficients
<input type="checkbox"/> MoO	Ideal gas "MoO" from NASA Glenn coefficients
<input type="checkbox"/> MoO2	Ideal gas "MoO2" from NASA Glenn coefficients
<input type="checkbox"/> MoO3	Ideal gas "MoO3" from NASA Glenn coefficients
<input type="checkbox"/> MoO3minus	Ideal gas "MoO3-" from NASA Glenn coefficients
<input type="checkbox"/> Mo2O6	Ideal gas "Mo2O6" from NASA Glenn coefficients
<input type="checkbox"/> Mo3O9	Ideal gas "Mo3O9" from NASA Glenn coefficients
<input type="checkbox"/> Mo4O12	Ideal gas "Mo4O12" from NASA Glenn coefficients
<input type="checkbox"/> Mo5O15	Ideal gas "Mo5O15" from NASA Glenn coefficients
<input type="checkbox"/> N	Ideal gas "N" from NASA Glenn coefficients
<input type="checkbox"/> Nplus	Ideal gas "N+" from NASA Glenn coefficients
<input type="checkbox"/> Nminus	Ideal gas "N-" from NASA Glenn coefficients
<input type="checkbox"/> NCO	Ideal gas "NCO" from NASA Glenn coefficients
<input type="checkbox"/> ND	Ideal gas "ND" from NASA Glenn coefficients
<input type="checkbox"/> ND2	Ideal gas "ND2" from NASA Glenn coefficients
<input type="checkbox"/> ND3	Ideal gas "ND3" from NASA Glenn coefficients
<input type="checkbox"/> NF	Ideal gas "NF" from NASA Glenn coefficients
<input type="checkbox"/> NF2	Ideal gas "NF2" from NASA Glenn coefficients
<input type="checkbox"/> NF3	Ideal gas "NF3" from NASA Glenn coefficients
<input type="checkbox"/> NH	Ideal gas "NH" from NASA Glenn coefficients
<input type="checkbox"/> NHplus	Ideal gas "NH+" from NASA Glenn coefficients
<input type="checkbox"/> NHF	Ideal gas "NHF" from NASA Glenn coefficients
<input type="checkbox"/> NHF2	Ideal gas "NHF2" from NASA Glenn coefficients

<input type="checkbox"/> NH2	Ideal gas "NH2" from NASA Glenn coefficients
<input type="checkbox"/> NH2F	Ideal gas "NH2F" from NASA Glenn coefficients
<input type="checkbox"/> NH3	Ideal gas "NH3" from NASA Glenn coefficients
<input type="checkbox"/> NH2OH	Ideal gas "NH2OH" from NASA Glenn coefficients
<input type="checkbox"/> NH4plus	Ideal gas "NH4+" from NASA Glenn coefficients
<input type="checkbox"/> NO	Ideal gas "NO" from NASA Glenn coefficients
<input type="checkbox"/> NOCL	Ideal gas "NOCl" from NASA Glenn coefficients
<input type="checkbox"/> NOF	Ideal gas "NOF" from NASA Glenn coefficients
<input type="checkbox"/> NOF3	Ideal gas "NOF3" from NASA Glenn coefficients
<input type="checkbox"/> NO2	Ideal gas "NO2" from NASA Glenn coefficients
<input type="checkbox"/> NO2minus	Ideal gas "NO2-" from NASA Glenn coefficients
<input type="checkbox"/> NO2CL	Ideal gas "NO2Cl" from NASA Glenn coefficients
<input type="checkbox"/> NO2F	Ideal gas "NO2F" from NASA Glenn coefficients
<input type="checkbox"/> NO3	Ideal gas "NO3" from NASA Glenn coefficients
<input type="checkbox"/> NO3minus	Ideal gas "NO3-" from NASA Glenn coefficients
<input type="checkbox"/> NO3F	Ideal gas "NO3F" from NASA Glenn coefficients
<input type="checkbox"/> N2	Ideal gas "N2" from NASA Glenn coefficients
<input type="checkbox"/> N2plus	Ideal gas "N2+" from NASA Glenn coefficients
<input type="checkbox"/> N2minus	Ideal gas "N2-" from NASA Glenn coefficients
<input type="checkbox"/> NCN	Ideal gas "NCN" from NASA Glenn coefficients
<input type="checkbox"/> N2D2_cis	Ideal gas "N2D2_cis" from NASA Glenn coefficients
<input type="checkbox"/> N2F2	Ideal gas "N2F2" from NASA Glenn coefficients
<input type="checkbox"/> N2F4	Ideal gas "N2F4" from NASA Glenn coefficients
<input type="checkbox"/> N2H2	Ideal gas "N2H2" from NASA Glenn coefficients
<input type="checkbox"/> NH2NO2	Ideal gas "NH2NO2" from NASA Glenn coefficients
<input type="checkbox"/> N2H4	Ideal gas "N2H4" from NASA Glenn coefficients
<input type="checkbox"/> N2O	Ideal gas "N2O" from NASA Glenn coefficients
<input type="checkbox"/> N2Oplus	Ideal gas "N2O+" from NASA Glenn coefficients
<input type="checkbox"/> N2O3	Ideal gas "N2O3" from NASA Glenn coefficients
<input type="checkbox"/> N2O4	Ideal gas "N2O4" from NASA Glenn coefficients
<input type="checkbox"/> N2O5	Ideal gas "N2O5" from NASA Glenn coefficients
<input type="checkbox"/> N3	Ideal gas "N3" from NASA Glenn coefficients
<input type="checkbox"/> N3H	Ideal gas "N3H" from NASA Glenn coefficients
<input type="checkbox"/> Na	Ideal gas "Na" from NASA Glenn coefficients
<input type="checkbox"/> Naplus	Ideal gas "Na+" from NASA Glenn coefficients
<input type="checkbox"/> Naminus	Ideal gas "Na-" from NASA Glenn coefficients
<input type="checkbox"/> NaAlF4	Ideal gas "NaAlF4" from NASA Glenn coefficients

<input type="checkbox"/> NaBO2	Ideal gas "NaBO2" from NASA Glenn coefficients
<input type="checkbox"/> NaBr	Ideal gas "NaBr" from NASA Glenn coefficients
<input type="checkbox"/> NaCN	Ideal gas "NaCN" from NASA Glenn coefficients
<input type="checkbox"/> NaCL	Ideal gas "NaCl" from NASA Glenn coefficients
<input type="checkbox"/> NaF	Ideal gas "NaF" from NASA Glenn coefficients
<input type="checkbox"/> NaH	Ideal gas "NaH" from NASA Glenn coefficients
<input type="checkbox"/> NaI	Ideal gas "NaI" from NASA Glenn coefficients
<input type="checkbox"/> NaLi	Ideal gas "NaLi" from NASA Glenn coefficients
<input type="checkbox"/> NaNO2	Ideal gas "NaNO2" from NASA Glenn coefficients
<input type="checkbox"/> NaNO3	Ideal gas "NaNO3" from NASA Glenn coefficients
<input type="checkbox"/> NaO	Ideal gas "NaO" from NASA Glenn coefficients
<input type="checkbox"/> NaOH	Ideal gas "NaOH" from NASA Glenn coefficients
<input type="checkbox"/> NaOHplus	Ideal gas "NaOH+" from NASA Glenn coefficients
<input type="checkbox"/> Na2	Ideal gas "Na2" from NASA Glenn coefficients
<input type="checkbox"/> Na2Br2	Ideal gas "Na2Br2" from NASA Glenn coefficients
<input type="checkbox"/> Na2CL2	Ideal gas "Na2Cl2" from NASA Glenn coefficients
<input type="checkbox"/> Na2F2	Ideal gas "Na2F2" from NASA Glenn coefficients
<input type="checkbox"/> Na2I2	Ideal gas "Na2I2" from NASA Glenn coefficients
<input type="checkbox"/> Na2O	Ideal gas "Na2O" from NASA Glenn coefficients
<input type="checkbox"/> Na2Oplus	Ideal gas "Na2O+" from NASA Glenn coefficients
<input type="checkbox"/> Na2O2	Ideal gas "Na2O2" from NASA Glenn coefficients
<input type="checkbox"/> Na2O2H2	Ideal gas "Na2O2H2" from NASA Glenn coefficients
<input type="checkbox"/> Na2SO4	Ideal gas "Na2SO4" from NASA Glenn coefficients
<input type="checkbox"/> Na3CL3	Ideal gas "Na3Cl3" from NASA Glenn coefficients
<input type="checkbox"/> Na3F3	Ideal gas "Na3F3" from NASA Glenn coefficients
<input type="checkbox"/> Nb	Ideal gas "Nb" from NASA Glenn coefficients
<input type="checkbox"/> Nbplus	Ideal gas "Nb+" from NASA Glenn coefficients
<input type="checkbox"/> Nbmminus	Ideal gas "Nb-" from NASA Glenn coefficients
<input type="checkbox"/> NbCL5	Ideal gas "NbCl5" from NASA Glenn coefficients
<input type="checkbox"/> NbO	Ideal gas "NbO" from NASA Glenn coefficients
<input type="checkbox"/> NbOCL3	Ideal gas "NbOCl3" from NASA Glenn coefficients
<input type="checkbox"/> NbO2	Ideal gas "NbO2" from NASA Glenn coefficients
<input type="checkbox"/> Ne	Ideal gas "Ne" from NASA Glenn coefficients
<input type="checkbox"/> Neplus	Ideal gas "Ne+" from NASA Glenn coefficients
<input type="checkbox"/> Ni	Ideal gas "Ni" from NASA Glenn coefficients
<input type="checkbox"/> Niplus	Ideal gas "Ni+" from NASA Glenn coefficients
<input type="checkbox"/> Niminus	Ideal gas "Ni-" from NASA Glenn coefficients

<input type="checkbox"/> NiCL	Ideal gas "NiCl" from NASA Glenn coefficients
<input type="checkbox"/> NiCL2	Ideal gas "NiCl2" from NASA Glenn coefficients
<input type="checkbox"/> NiO	Ideal gas "NiO" from NASA Glenn coefficients
<input type="checkbox"/> NiS	Ideal gas "NiS" from NASA Glenn coefficients
<input type="checkbox"/> O	Ideal gas "O" from NASA Glenn coefficients
<input type="checkbox"/> Oplus	Ideal gas "O+" from NASA Glenn coefficients
<input type="checkbox"/> Ominus	Ideal gas "O-" from NASA Glenn coefficients
<input type="checkbox"/> OD	Ideal gas "OD" from NASA Glenn coefficients
<input type="checkbox"/> ODminus	Ideal gas "OD-" from NASA Glenn coefficients
<input type="checkbox"/> OH	Ideal gas "OH" from NASA Glenn coefficients
<input type="checkbox"/> OHplus	Ideal gas "OH+" from NASA Glenn coefficients
<input type="checkbox"/> OHminus	Ideal gas "OH-" from NASA Glenn coefficients
<input type="checkbox"/> O2	Ideal gas "O2" from NASA Glenn coefficients
<input type="checkbox"/> O2plus	Ideal gas "O2+" from NASA Glenn coefficients
<input type="checkbox"/> O2minus	Ideal gas "O2-" from NASA Glenn coefficients
<input type="checkbox"/> O3	Ideal gas "O3" from NASA Glenn coefficients
<input type="checkbox"/> P	Ideal gas "P" from NASA Glenn coefficients
<input type="checkbox"/> Pplus	Ideal gas "P+" from NASA Glenn coefficients
<input type="checkbox"/> Pminus	Ideal gas "P-" from NASA Glenn coefficients
<input type="checkbox"/> PCL	Ideal gas "PCI" from NASA Glenn coefficients
<input type="checkbox"/> PCL2	Ideal gas "PCI2" from NASA Glenn coefficients
<input type="checkbox"/> PCL2minus	Ideal gas "PCI2-" from NASA Glenn coefficients
<input type="checkbox"/> PCL3	Ideal gas "PCI3" from NASA Glenn coefficients
<input type="checkbox"/> PCL5	Ideal gas "PCI5" from NASA Glenn coefficients
<input type="checkbox"/> PF	Ideal gas "PF" from NASA Glenn coefficients
<input type="checkbox"/> PFplus	Ideal gas "PF+" from NASA Glenn coefficients
<input type="checkbox"/> PFminus	Ideal gas "PF-" from NASA Glenn coefficients
<input type="checkbox"/> PFCL	Ideal gas "PFCI" from NASA Glenn coefficients
<input type="checkbox"/> PFCLminus	Ideal gas "PFCI-" from NASA Glenn coefficients
<input type="checkbox"/> PFCL2	Ideal gas "PFCI2" from NASA Glenn coefficients
<input type="checkbox"/> PFCL4	Ideal gas "PFCI4" from NASA Glenn coefficients
<input type="checkbox"/> PF2	Ideal gas "PF2" from NASA Glenn coefficients
<input type="checkbox"/> PF2minus	Ideal gas "PF2-" from NASA Glenn coefficients
<input type="checkbox"/> PF2CL	Ideal gas "PF2CI" from NASA Glenn coefficients
<input type="checkbox"/> PF2CL3	Ideal gas "PF2CI3" from NASA Glenn coefficients
<input type="checkbox"/> PF3	Ideal gas "PF3" from NASA Glenn coefficients
<input type="checkbox"/> PF3CL2	Ideal gas "PF3CI2" from NASA Glenn coefficients

<input type="checkbox"/> PF4CL	Ideal gas "PF4Cl" from NASA Glenn coefficients
<input type="checkbox"/> PF5	Ideal gas "PF5" from NASA Glenn coefficients
<input type="checkbox"/> PH	Ideal gas "PH" from NASA Glenn coefficients
<input type="checkbox"/> PH2	Ideal gas "PH2" from NASA Glenn coefficients
<input type="checkbox"/> PH2minus	Ideal gas "PH2-" from NASA Glenn coefficients
<input type="checkbox"/> PH3	Ideal gas "PH3" from NASA Glenn coefficients
<input type="checkbox"/> PN	Ideal gas "PN" from NASA Glenn coefficients
<input type="checkbox"/> PO	Ideal gas "PO" from NASA Glenn coefficients
<input type="checkbox"/> POminus	Ideal gas "PO-" from NASA Glenn coefficients
<input type="checkbox"/> POCL3	Ideal gas "POCl3" from NASA Glenn coefficients
<input type="checkbox"/> POFCL2	Ideal gas "POFCI2" from NASA Glenn coefficients
<input type="checkbox"/> POF2CL	Ideal gas "POF2Cl" from NASA Glenn coefficients
<input type="checkbox"/> POF3	Ideal gas "POF3" from NASA Glenn coefficients
<input type="checkbox"/> PO2	Ideal gas "PO2" from NASA Glenn coefficients
<input type="checkbox"/> PO2minus	Ideal gas "PO2-" from NASA Glenn coefficients
<input type="checkbox"/> PS	Ideal gas "PS" from NASA Glenn coefficients
<input type="checkbox"/> P2	Ideal gas "P2" from NASA Glenn coefficients
<input type="checkbox"/> P2O3	Ideal gas "P2O3" from NASA Glenn coefficients
<input type="checkbox"/> P2O4	Ideal gas "P2O4" from NASA Glenn coefficients
<input type="checkbox"/> P2O5	Ideal gas "P2O5" from NASA Glenn coefficients
<input type="checkbox"/> P3	Ideal gas "P3" from NASA Glenn coefficients
<input type="checkbox"/> P3O6	Ideal gas "P3O6" from NASA Glenn coefficients
<input type="checkbox"/> P4	Ideal gas "P4" from NASA Glenn coefficients
<input type="checkbox"/> P4O6	Ideal gas "P4O6" from NASA Glenn coefficients
<input type="checkbox"/> P4O7	Ideal gas "P4O7" from NASA Glenn coefficients
<input type="checkbox"/> P4O8	Ideal gas "P4O8" from NASA Glenn coefficients
<input type="checkbox"/> P4O9	Ideal gas "P4O9" from NASA Glenn coefficients
<input type="checkbox"/> P4O10	Ideal gas "P4O10" from NASA Glenn coefficients
<input type="checkbox"/> Pb	Ideal gas "Pb" from NASA Glenn coefficients
<input type="checkbox"/> Pbplus	Ideal gas "Pb+" from NASA Glenn coefficients
<input type="checkbox"/> Pbminus	Ideal gas "Pb-" from NASA Glenn coefficients
<input type="checkbox"/> PbBr	Ideal gas "PbBr" from NASA Glenn coefficients
<input type="checkbox"/> PbBr2	Ideal gas "PbBr2" from NASA Glenn coefficients
<input type="checkbox"/> PbBr3	Ideal gas "PbBr3" from NASA Glenn coefficients
<input type="checkbox"/> PbBr4	Ideal gas "PbBr4" from NASA Glenn coefficients
<input type="checkbox"/> PbCL	Ideal gas "PbCl" from NASA Glenn coefficients
<input type="checkbox"/> PbCL2	Ideal gas "PbCl2" from NASA Glenn coefficients

<input type="checkbox"/> PbCl3	Ideal gas "PbCl3" from NASA Glenn coefficients
<input type="checkbox"/> PbCl4	Ideal gas "PbCl4" from NASA Glenn coefficients
<input type="checkbox"/> PbF	Ideal gas "PbF" from NASA Glenn coefficients
<input type="checkbox"/> PbF2	Ideal gas "PbF2" from NASA Glenn coefficients
<input type="checkbox"/> PbF3	Ideal gas "PbF3" from NASA Glenn coefficients
<input type="checkbox"/> PbF4	Ideal gas "PbF4" from NASA Glenn coefficients
<input type="checkbox"/> Pbl	Ideal gas "Pbl" from NASA Glenn coefficients
<input type="checkbox"/> Pbl2	Ideal gas "Pbl2" from NASA Glenn coefficients
<input type="checkbox"/> Pbl3	Ideal gas "Pbl3" from NASA Glenn coefficients
<input type="checkbox"/> Pbl4	Ideal gas "Pbl4" from NASA Glenn coefficients
<input type="checkbox"/> PbO	Ideal gas "PbO" from NASA Glenn coefficients
<input type="checkbox"/> PbO2	Ideal gas "PbO2" from NASA Glenn coefficients
<input type="checkbox"/> PbS	Ideal gas "PbS" from NASA Glenn coefficients
<input type="checkbox"/> PbS2	Ideal gas "PbS2" from NASA Glenn coefficients
<input type="checkbox"/> Rb	Ideal gas "Rb" from NASA Glenn coefficients
<input type="checkbox"/> Rbplus	Ideal gas "Rb+" from NASA Glenn coefficients
<input type="checkbox"/> Rbminus	Ideal gas "Rb-" from NASA Glenn coefficients
<input type="checkbox"/> RbBO2	Ideal gas "RbBO2" from NASA Glenn coefficients
<input type="checkbox"/> RbBr	Ideal gas "RbBr" from NASA Glenn coefficients
<input type="checkbox"/> RbCL	Ideal gas "RbCl" from NASA Glenn coefficients
<input type="checkbox"/> RbF	Ideal gas "RbF" from NASA Glenn coefficients
<input type="checkbox"/> RbH	Ideal gas "RbH" from NASA Glenn coefficients
<input type="checkbox"/> Rbl	Ideal gas "Rbl" from NASA Glenn coefficients
<input type="checkbox"/> RbK	Ideal gas "RbK" from NASA Glenn coefficients
<input type="checkbox"/> RbLi	Ideal gas "Rbli" from NASA Glenn coefficients
<input type="checkbox"/> RbNO2	Ideal gas "RbNO2" from NASA Glenn coefficients
<input type="checkbox"/> RbNO3	Ideal gas "RbNO3" from NASA Glenn coefficients
<input type="checkbox"/> RbNa	Ideal gas "RbNa" from NASA Glenn coefficients
<input type="checkbox"/> RbO	Ideal gas "RbO" from NASA Glenn coefficients
<input type="checkbox"/> RbOH	Ideal gas "RbOH" from NASA Glenn coefficients
<input type="checkbox"/> Rb2Br2	Ideal gas "Rb2Br2" from NASA Glenn coefficients
<input type="checkbox"/> Rb2CL2	Ideal gas "Rb2Cl2" from NASA Glenn coefficients
<input type="checkbox"/> Rb2F2	Ideal gas "Rb2F2" from NASA Glenn coefficients
<input type="checkbox"/> Rb2I2	Ideal gas "Rb2I2" from NASA Glenn coefficients
<input type="checkbox"/> Rb2O	Ideal gas "Rb2O" from NASA Glenn coefficients
<input type="checkbox"/> Rb2O2	Ideal gas "Rb2O2" from NASA Glenn coefficients
<input type="checkbox"/> Rb2O2H2	Ideal gas "Rb2O2H2" from NASA Glenn coefficients

---

**1266 Modelica.Media.IdealGases.SingleGases**


---

<input type="checkbox"/> Rb2SO4	Ideal gas "Rb2SO4" from NASA Glenn coefficients
<input type="checkbox"/> Rn	Ideal gas "Rn" from NASA Glenn coefficients
<input type="checkbox"/> Rnplus	Ideal gas "Rn+" from NASA Glenn coefficients
<input type="checkbox"/> S	Ideal gas "S" from NASA Glenn coefficients
<input type="checkbox"/> Splus	Ideal gas "S+" from NASA Glenn coefficients
<input type="checkbox"/> Sminus	Ideal gas "S-" from NASA Glenn coefficients
<input type="checkbox"/> SCL	Ideal gas "SCL" from NASA Glenn coefficients
<input type="checkbox"/> SCL2	Ideal gas "SCL2" from NASA Glenn coefficients
<input type="checkbox"/> SCL2plus	Ideal gas "SCL2+" from NASA Glenn coefficients
<input type="checkbox"/> SD	Ideal gas "SD" from NASA Glenn coefficients
<input type="checkbox"/> SF	Ideal gas "SF" from NASA Glenn coefficients
<input type="checkbox"/> SFplus	Ideal gas "SF+" from NASA Glenn coefficients
<input type="checkbox"/> SFminus	Ideal gas "SF-" from NASA Glenn coefficients
<input type="checkbox"/> SF2	Ideal gas "SF2" from NASA Glenn coefficients
<input type="checkbox"/> SF2plus	Ideal gas "SF2+" from NASA Glenn coefficients
<input type="checkbox"/> SF2minus	Ideal gas "SF2-" from NASA Glenn coefficients
<input type="checkbox"/> SF3	Ideal gas "SF3" from NASA Glenn coefficients
<input type="checkbox"/> SF3plus	Ideal gas "SF3+" from NASA Glenn coefficients
<input type="checkbox"/> SF3minus	Ideal gas "SF3-" from NASA Glenn coefficients
<input type="checkbox"/> SF4	Ideal gas "SF4" from NASA Glenn coefficients
<input type="checkbox"/> SF4plus	Ideal gas "SF4+" from NASA Glenn coefficients
<input type="checkbox"/> SF4minus	Ideal gas "SF4-" from NASA Glenn coefficients
<input type="checkbox"/> SF5	Ideal gas "SF5" from NASA Glenn coefficients
<input type="checkbox"/> SF5plus	Ideal gas "SF5+" from NASA Glenn coefficients
<input type="checkbox"/> SF5minus	Ideal gas "SF5-" from NASA Glenn coefficients
<input type="checkbox"/> SF6	Ideal gas "SF6" from NASA Glenn coefficients
<input type="checkbox"/> SF6minus	Ideal gas "SF6-" from NASA Glenn coefficients
<input type="checkbox"/> SH	Ideal gas "SH" from NASA Glenn coefficients
<input type="checkbox"/> SHminus	Ideal gas "SH-" from NASA Glenn coefficients
<input type="checkbox"/> SN	Ideal gas "SN" from NASA Glenn coefficients
<input type="checkbox"/> SO	Ideal gas "SO" from NASA Glenn coefficients
<input type="checkbox"/> SOminus	Ideal gas "SO-" from NASA Glenn coefficients
<input type="checkbox"/> SOF2	Ideal gas "SOF2" from NASA Glenn coefficients
<input type="checkbox"/> SO2	Ideal gas "SO2" from NASA Glenn coefficients
<input type="checkbox"/> SO2minus	Ideal gas "SO2-" from NASA Glenn coefficients
<input type="checkbox"/> SO2CL2	Ideal gas "SO2Cl2" from NASA Glenn coefficients
<input type="checkbox"/> SO2FCL	Ideal gas "SO2FCI" from NASA Glenn coefficients

<input type="checkbox"/> SO2F2	Ideal gas "SO2F2" from NASA Glenn coefficients
<input type="checkbox"/> SO3	Ideal gas "SO3" from NASA Glenn coefficients
<input type="checkbox"/> S2	Ideal gas "S2" from NASA Glenn coefficients
<input type="checkbox"/> S2minus	Ideal gas "S2-" from NASA Glenn coefficients
<input type="checkbox"/> S2Cl2	Ideal gas "S2Cl2" from NASA Glenn coefficients
<input type="checkbox"/> S2F2	Ideal gas "S2F2" from NASA Glenn coefficients
<input type="checkbox"/> S2O	Ideal gas "S2O" from NASA Glenn coefficients
<input type="checkbox"/> S3	Ideal gas "S3" from NASA Glenn coefficients
<input type="checkbox"/> S4	Ideal gas "S4" from NASA Glenn coefficients
<input type="checkbox"/> S5	Ideal gas "S5" from NASA Glenn coefficients
<input type="checkbox"/> S6	Ideal gas "S6" from NASA Glenn coefficients
<input type="checkbox"/> S7	Ideal gas "S7" from NASA Glenn coefficients
<input type="checkbox"/> S8	Ideal gas "S8" from NASA Glenn coefficients
<input type="checkbox"/> Sc	Ideal gas "Sc" from NASA Glenn coefficients
<input type="checkbox"/> Scplus	Ideal gas "Sc+" from NASA Glenn coefficients
<input type="checkbox"/> Scminus	Ideal gas "Sc-" from NASA Glenn coefficients
<input type="checkbox"/> ScO	Ideal gas "ScO" from NASA Glenn coefficients
<input type="checkbox"/> ScOplus	Ideal gas "ScO+" from NASA Glenn coefficients
<input type="checkbox"/> ScO2	Ideal gas "ScO2" from NASA Glenn coefficients
<input type="checkbox"/> Sc2O	Ideal gas "Sc2O" from NASA Glenn coefficients
<input type="checkbox"/> Sc2O2	Ideal gas "Sc2O2" from NASA Glenn coefficients
<input type="checkbox"/> Si	Ideal gas "Si" from NASA Glenn coefficients
<input type="checkbox"/> Siplus	Ideal gas "Si+" from NASA Glenn coefficients
<input type="checkbox"/> Siminus	Ideal gas "Si-" from NASA Glenn coefficients
<input type="checkbox"/> SiBr	Ideal gas "SiBr" from NASA Glenn coefficients
<input type="checkbox"/> SiBr2	Ideal gas "SiBr2" from NASA Glenn coefficients
<input type="checkbox"/> SiBr3	Ideal gas "SiBr3" from NASA Glenn coefficients
<input type="checkbox"/> SiBr4	Ideal gas "SiBr4" from NASA Glenn coefficients
<input type="checkbox"/> SiC	Ideal gas "SiC" from NASA Glenn coefficients
<input type="checkbox"/> SiC2	Ideal gas "SiC2" from NASA Glenn coefficients
<input type="checkbox"/> SiCl	Ideal gas "SiCl" from NASA Glenn coefficients
<input type="checkbox"/> SiCl2	Ideal gas "SiCl2" from NASA Glenn coefficients
<input type="checkbox"/> SiCl3	Ideal gas "SiCl3" from NASA Glenn coefficients
<input type="checkbox"/> SiCl4	Ideal gas "SiCl4" from NASA Glenn coefficients
<input type="checkbox"/> SiF	Ideal gas "SiF" from NASA Glenn coefficients
<input type="checkbox"/> SiFCL	Ideal gas "SiFCI" from NASA Glenn coefficients
<input type="checkbox"/> SiF2	Ideal gas "SiF2" from NASA Glenn coefficients

<input type="checkbox"/> SiF3	Ideal gas "SiF3" from NASA Glenn coefficients
<input type="checkbox"/> SiF4	Ideal gas "SiF4" from NASA Glenn coefficients
<input type="checkbox"/> SiH	Ideal gas "SiH" from NASA Glenn coefficients
<input type="checkbox"/> SiHplus	Ideal gas "SiH+" from NASA Glenn coefficients
<input type="checkbox"/> SiHBr3	Ideal gas "SiHBr3" from NASA Glenn coefficients
<input type="checkbox"/> SiHCL	Ideal gas "SiHCl" from NASA Glenn coefficients
<input type="checkbox"/> SiHCL3	Ideal gas "SiHCl3" from NASA Glenn coefficients
<input type="checkbox"/> SiHF	Ideal gas "SiHF" from NASA Glenn coefficients
<input type="checkbox"/> SiHF3	Ideal gas "SiHF3" from NASA Glenn coefficients
<input type="checkbox"/> SiHI3	Ideal gas "SiHI3" from NASA Glenn coefficients
<input type="checkbox"/> SiH2	Ideal gas "SiH2" from NASA Glenn coefficients
<input type="checkbox"/> SiH2Br2	Ideal gas "SiH2Br2" from NASA Glenn coefficients
<input type="checkbox"/> SiH2CL2	Ideal gas "SiH2Cl2" from NASA Glenn coefficients
<input type="checkbox"/> SiH2F2	Ideal gas "SiH2F2" from NASA Glenn coefficients
<input type="checkbox"/> SiH2I2	Ideal gas "SiH2I2" from NASA Glenn coefficients
<input type="checkbox"/> SiH3	Ideal gas "SiH3" from NASA Glenn coefficients
<input type="checkbox"/> SiH3Br	Ideal gas "SiH3Br" from NASA Glenn coefficients
<input type="checkbox"/> SiH3CL	Ideal gas "SiH3Cl" from NASA Glenn coefficients
<input type="checkbox"/> SiH3F	Ideal gas "SiH3F" from NASA Glenn coefficients
<input type="checkbox"/> SiH3I	Ideal gas "SiH3I" from NASA Glenn coefficients
<input type="checkbox"/> SiH4	Ideal gas "SiH4" from NASA Glenn coefficients
<input type="checkbox"/> SiI	Ideal gas "SiI" from NASA Glenn coefficients
<input type="checkbox"/> SiI2	Ideal gas "SiI2" from NASA Glenn coefficients
<input type="checkbox"/> SiN	Ideal gas "SiN" from NASA Glenn coefficients
<input type="checkbox"/> SiO	Ideal gas "SiO" from NASA Glenn coefficients
<input type="checkbox"/> SiO2	Ideal gas "SiO2" from NASA Glenn coefficients
<input type="checkbox"/> SiS	Ideal gas "SiS" from NASA Glenn coefficients
<input type="checkbox"/> SiS2	Ideal gas "SiS2" from NASA Glenn coefficients
<input type="checkbox"/> Si2	Ideal gas "Si2" from NASA Glenn coefficients
<input type="checkbox"/> Si2C	Ideal gas "Si2C" from NASA Glenn coefficients
<input type="checkbox"/> Si2F6	Ideal gas "Si2F6" from NASA Glenn coefficients
<input type="checkbox"/> Si2N	Ideal gas "Si2N" from NASA Glenn coefficients
<input type="checkbox"/> Si3	Ideal gas "Si3" from NASA Glenn coefficients
<input type="checkbox"/> Sn	Ideal gas "Sn" from NASA Glenn coefficients
<input type="checkbox"/> Snplus	Ideal gas "Sn+" from NASA Glenn coefficients
<input type="checkbox"/> Snminus	Ideal gas "Sn-" from NASA Glenn coefficients
<input type="checkbox"/> SnBr	Ideal gas "SnBr" from NASA Glenn coefficients

<input type="checkbox"/> SnBr2	Ideal gas "SnBr2" from NASA Glenn coefficients
<input type="checkbox"/> SnBr3	Ideal gas "SnBr3" from NASA Glenn coefficients
<input type="checkbox"/> SnBr4	Ideal gas "SnBr4" from NASA Glenn coefficients
<input type="checkbox"/> SnCL	Ideal gas "SnCl" from NASA Glenn coefficients
<input type="checkbox"/> SnCL2	Ideal gas "SnCl2" from NASA Glenn coefficients
<input type="checkbox"/> SnCL3	Ideal gas "SnCl3" from NASA Glenn coefficients
<input type="checkbox"/> SnCL4	Ideal gas "SnCl4" from NASA Glenn coefficients
<input type="checkbox"/> SnF	Ideal gas "SnF" from NASA Glenn coefficients
<input type="checkbox"/> SnF2	Ideal gas "SnF2" from NASA Glenn coefficients
<input type="checkbox"/> SnF3	Ideal gas "SnF3" from NASA Glenn coefficients
<input type="checkbox"/> SnF4	Ideal gas "SnF4" from NASA Glenn coefficients
<input type="checkbox"/> SnI	Ideal gas "SnI" from NASA Glenn coefficients
<input type="checkbox"/> SnI2	Ideal gas "SnI2" from NASA Glenn coefficients
<input type="checkbox"/> SnI3	Ideal gas "SnI3" from NASA Glenn coefficients
<input type="checkbox"/> SnI4	Ideal gas "SnI4" from NASA Glenn coefficients
<input type="checkbox"/> SnO	Ideal gas "SnO" from NASA Glenn coefficients
<input type="checkbox"/> SnO2	Ideal gas "SnO2" from NASA Glenn coefficients
<input type="checkbox"/> SnS	Ideal gas "SnS" from NASA Glenn coefficients
<input type="checkbox"/> SnS2	Ideal gas "SnS2" from NASA Glenn coefficients
<input type="checkbox"/> Sn2	Ideal gas "Sn2" from NASA Glenn coefficients
<input type="checkbox"/> Sr	Ideal gas "Sr" from NASA Glenn coefficients
<input type="checkbox"/> Srplus	Ideal gas "Sr+" from NASA Glenn coefficients
<input type="checkbox"/> SrBr	Ideal gas "SrBr" from NASA Glenn coefficients
<input type="checkbox"/> SrBr2	Ideal gas "SrBr2" from NASA Glenn coefficients
<input type="checkbox"/> SrCL	Ideal gas "SrCl" from NASA Glenn coefficients
<input type="checkbox"/> SrCLplus	Ideal gas "SrCl+" from NASA Glenn coefficients
<input type="checkbox"/> SrCL2	Ideal gas "SrCl2" from NASA Glenn coefficients
<input type="checkbox"/> SrF	Ideal gas "SrF" from NASA Glenn coefficients
<input type="checkbox"/> SrFplus	Ideal gas "SrF+" from NASA Glenn coefficients
<input type="checkbox"/> SrF2	Ideal gas "SrF2" from NASA Glenn coefficients
<input type="checkbox"/> SrH	Ideal gas "SrH" from NASA Glenn coefficients
<input type="checkbox"/> Srl	Ideal gas "Srl" from NASA Glenn coefficients
<input type="checkbox"/> Srl2	Ideal gas "Srl2" from NASA Glenn coefficients
<input type="checkbox"/> SrO	Ideal gas "SrO" from NASA Glenn coefficients
<input type="checkbox"/> SrOH	Ideal gas "SrOH" from NASA Glenn coefficients
<input type="checkbox"/> SrOHplus	Ideal gas "SrOH+" from NASA Glenn coefficients
<input type="checkbox"/> Sr_OH_2	Ideal gas "Sr_OH_2" from NASA Glenn coefficients

---

**1270 Modelica.Media.IdealGases.SingleGases**


---

<input type="checkbox"/> SrS	Ideal gas "SrS" from NASA Glenn coefficients
<input type="checkbox"/> Sr2	Ideal gas "Sr2" from NASA Glenn coefficients
<input type="checkbox"/> Ta	Ideal gas "Ta" from NASA Glenn coefficients
<input type="checkbox"/> Taplus	Ideal gas "Ta+" from NASA Glenn coefficients
<input type="checkbox"/> Taminus	Ideal gas "Ta-" from NASA Glenn coefficients
<input type="checkbox"/> TaCL5	Ideal gas "TaCl5" from NASA Glenn coefficients
<input type="checkbox"/> TaO	Ideal gas "TaO" from NASA Glenn coefficients
<input type="checkbox"/> TaO2	Ideal gas "TaO2" from NASA Glenn coefficients
<input type="checkbox"/> Ti	Ideal gas "Ti" from NASA Glenn coefficients
<input type="checkbox"/> Tiplus	Ideal gas "Ti+" from NASA Glenn coefficients
<input type="checkbox"/> Timinus	Ideal gas "Ti-" from NASA Glenn coefficients
<input type="checkbox"/> TiCL	Ideal gas "TiCl" from NASA Glenn coefficients
<input type="checkbox"/> TiCL2	Ideal gas "TiCl2" from NASA Glenn coefficients
<input type="checkbox"/> TiCL3	Ideal gas "TiCl3" from NASA Glenn coefficients
<input type="checkbox"/> TiCL4	Ideal gas "TiCl4" from NASA Glenn coefficients
<input type="checkbox"/> TiO	Ideal gas "TiO" from NASA Glenn coefficients
<input type="checkbox"/> TiOplus	Ideal gas "TiO+" from NASA Glenn coefficients
<input type="checkbox"/> TiOCL	Ideal gas "TiOCl" from NASA Glenn coefficients
<input type="checkbox"/> TiOCL2	Ideal gas "TiOCl2" from NASA Glenn coefficients
<input type="checkbox"/> TiO2	Ideal gas "TiO2" from NASA Glenn coefficients
<input type="checkbox"/> U	Ideal gas "U" from NASA Glenn coefficients
<input type="checkbox"/> UF	Ideal gas "UF" from NASA Glenn coefficients
<input type="checkbox"/> UFplus	Ideal gas "UF+" from NASA Glenn coefficients
<input type="checkbox"/> UFminus	Ideal gas "UF-" from NASA Glenn coefficients
<input type="checkbox"/> UF2	Ideal gas "UF2" from NASA Glenn coefficients
<input type="checkbox"/> UF2plus	Ideal gas "UF2+" from NASA Glenn coefficients
<input type="checkbox"/> UF2minus	Ideal gas "UF2-" from NASA Glenn coefficients
<input type="checkbox"/> UF3	Ideal gas "UF3" from NASA Glenn coefficients
<input type="checkbox"/> UF3plus	Ideal gas "UF3+" from NASA Glenn coefficients
<input type="checkbox"/> UF3minus	Ideal gas "UF3-" from NASA Glenn coefficients
<input type="checkbox"/> UF4	Ideal gas "UF4" from NASA Glenn coefficients
<input type="checkbox"/> UF4plus	Ideal gas "UF4+" from NASA Glenn coefficients
<input type="checkbox"/> UF4minus	Ideal gas "UF4-" from NASA Glenn coefficients
<input type="checkbox"/> UF5	Ideal gas "UF5" from NASA Glenn coefficients
<input type="checkbox"/> UF5plus	Ideal gas "UF5+" from NASA Glenn coefficients
<input type="checkbox"/> UF5minus	Ideal gas "UF5-" from NASA Glenn coefficients
<input type="checkbox"/> UF6	Ideal gas "UF6" from NASA Glenn coefficients

<input type="checkbox"/> UF6minus	Ideal gas "UF6-" from NASA Glenn coefficients
<input type="checkbox"/> UO	Ideal gas "UO" from NASA Glenn coefficients
<input type="checkbox"/> UOplus	Ideal gas "UO+" from NASA Glenn coefficients
<input type="checkbox"/> UOF	Ideal gas "UOF" from NASA Glenn coefficients
<input type="checkbox"/> UOF2	Ideal gas "UOF2" from NASA Glenn coefficients
<input type="checkbox"/> UOF3	Ideal gas "UOF3" from NASA Glenn coefficients
<input type="checkbox"/> UOF4	Ideal gas "UOF4" from NASA Glenn coefficients
<input type="checkbox"/> UO2	Ideal gas "UO2" from NASA Glenn coefficients
<input type="checkbox"/> UO2plus	Ideal gas "UO2+" from NASA Glenn coefficients
<input type="checkbox"/> UO2minus	Ideal gas "UO2-" from NASA Glenn coefficients
<input type="checkbox"/> UO2F	Ideal gas "UO2F" from NASA Glenn coefficients
<input type="checkbox"/> UO2F2	Ideal gas "UO2F2" from NASA Glenn coefficients
<input type="checkbox"/> UO3	Ideal gas "UO3" from NASA Glenn coefficients
<input type="checkbox"/> UO3minus	Ideal gas "UO3-" from NASA Glenn coefficients
<input type="checkbox"/> V	Ideal gas "V" from NASA Glenn coefficients
<input type="checkbox"/> Vplus	Ideal gas "V+" from NASA Glenn coefficients
<input type="checkbox"/> Vminus	Ideal gas "V-" from NASA Glenn coefficients
<input type="checkbox"/> VCl4	Ideal gas "VCl4" from NASA Glenn coefficients
<input type="checkbox"/> VN	Ideal gas "VN" from NASA Glenn coefficients
<input type="checkbox"/> VO	Ideal gas "VO" from NASA Glenn coefficients
<input type="checkbox"/> VO2	Ideal gas "VO2" from NASA Glenn coefficients
<input type="checkbox"/> V4O10	Ideal gas "V4O10" from NASA Glenn coefficients
<input type="checkbox"/> W	Ideal gas "W" from NASA Glenn coefficients
<input type="checkbox"/> Wplus	Ideal gas "W+" from NASA Glenn coefficients
<input type="checkbox"/> Wminus	Ideal gas "W-" from NASA Glenn coefficients
<input type="checkbox"/> WCl6	Ideal gas "WCl6" from NASA Glenn coefficients
<input type="checkbox"/> WO	Ideal gas "WO" from NASA Glenn coefficients
<input type="checkbox"/> WOCl4	Ideal gas "WOCl4" from NASA Glenn coefficients
<input type="checkbox"/> WO2	Ideal gas "WO2" from NASA Glenn coefficients
<input type="checkbox"/> WO2Cl2	Ideal gas "WO2Cl2" from NASA Glenn coefficients
<input type="checkbox"/> WO3	Ideal gas "WO3" from NASA Glenn coefficients
<input type="checkbox"/> WO3minus	Ideal gas "WO3-" from NASA Glenn coefficients
<input type="checkbox"/> Xe	Ideal gas "Xe" from NASA Glenn coefficients
<input type="checkbox"/> Xeplus	Ideal gas "Xe+" from NASA Glenn coefficients
<input type="checkbox"/> Zn	Ideal gas "Zn" from NASA Glenn coefficients
<input type="checkbox"/> Znplus	Ideal gas "Zn+" from NASA Glenn coefficients
<input type="checkbox"/> Zr	Ideal gas "Zr" from NASA Glenn coefficients

## 1272 Modelica.Media.IdealGases.SingleGases

---

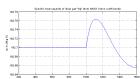
<input type="checkbox"/> Zrplus	Ideal gas "Zr+" from NASA Glenn coefficients
<input type="checkbox"/> Zrminus	Ideal gas "Zr-" from NASA Glenn coefficients
<input type="checkbox"/> ZrN	Ideal gas "ZrN" from NASA Glenn coefficients
<input type="checkbox"/> ZrO	Ideal gas "ZrO" from NASA Glenn coefficients
<input type="checkbox"/> ZrOplus	Ideal gas "ZrO+" from NASA Glenn coefficients
<input type="checkbox"/> ZrO2	Ideal gas "ZrO2" from NASA Glenn coefficients

---

### Modelica.Media.IdealGases.SingleGases.Ag

Ideal gas "Ag" from NASA Glenn coefficients

#### Information

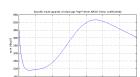


---

### Modelica.Media.IdealGases.SingleGases.Agplus

Ideal gas "Ag+" from NASA Glenn coefficients

#### Information



---

### Modelica.Media.IdealGases.SingleGases.Agminus

Ideal gas "Ag-" from NASA Glenn coefficients

#### Information

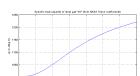


---

### Modelica.Media.IdealGases.SingleGases.Air

Ideal gas "Air" from NASA Glenn coefficients

#### Information

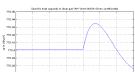


**Modelica.Media.IdealGases.SingleGases.AL**

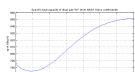
Ideal gas "Al" from NASA Glenn coefficients

**Information****Modelica.Media.IdealGases.SingleGases.ALplus**

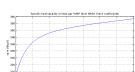
Ideal gas "Al+" from NASA Glenn coefficients

**Information****Modelica.Media.IdealGases.SingleGases.ALminus**

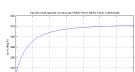
Ideal gas "Al-" from NASA Glenn coefficients

**Information****Modelica.Media.IdealGases.SingleGases.ALBr**

Ideal gas "AlBr" from NASA Glenn coefficients

**Information****Modelica.Media.IdealGases.SingleGases.ALBr2**

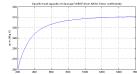
Ideal gas "AlBr2" from NASA Glenn coefficients

**Information**

**Modelica.Media.IdealGases.SingleGases.ALBr3**

Ideal gas "AlBr3" from NASA Glenn coefficients

**Information**

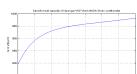


---

**Modelica.Media.IdealGases.SingleGases.ALC**

Ideal gas "AIC" from NASA Glenn coefficients

**Information**

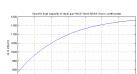


---

**Modelica.Media.IdealGases.SingleGases.ALC2**

Ideal gas "AIC2" from NASA Glenn coefficients

**Information**

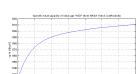


---

**Modelica.Media.IdealGases.SingleGases.ALCL**

Ideal gas "AlCl" from NASA Glenn coefficients

**Information**

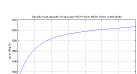


---

**Modelica.Media.IdealGases.SingleGases.ALCLplus**

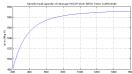
Ideal gas "AlCl+" from NASA Glenn coefficients

**Information**

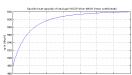


**Modelica.Media.IdealGases.SingleGases.ALCL2**

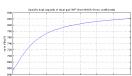
Ideal gas "AlCl2" from NASA Glenn coefficients

**Information****Modelica.Media.IdealGases.SingleGases.ALCL3**

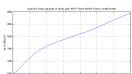
Ideal gas "AlCl3" from NASA Glenn coefficients

**Information****Modelica.Media.IdealGases.SingleGases.ALF**

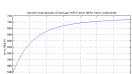
Ideal gas "AlF" from NASA Glenn coefficients

**Information****Modelica.Media.IdealGases.SingleGases.ALFplus**

Ideal gas "AlF+" from NASA Glenn coefficients

**Information****Modelica.Media.IdealGases.SingleGases.ALFCL**

Ideal gas "AlFCI" from NASA Glenn coefficients

**Information**

---

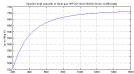
## 1276 Modelica.Media.IdealGases.SingleGases.ALFCL2

---

### Modelica.Media.IdealGases.SingleGases.ALFCL2

Ideal gas "AIFCI2" from NASA Glenn coefficients

#### Information

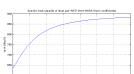


---

### Modelica.Media.IdealGases.SingleGases.ALF2

Ideal gas "AIF2" from NASA Glenn coefficients

#### Information

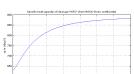


---

### Modelica.Media.IdealGases.SingleGases.ALF2minus

Ideal gas "AIF2-" from NASA Glenn coefficients

#### Information

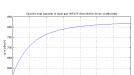


---

### Modelica.Media.IdealGases.SingleGases.ALF2CL

Ideal gas "AIF2CI" from NASA Glenn coefficients

#### Information

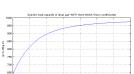


---

### Modelica.Media.IdealGases.SingleGases.ALF3

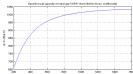
Ideal gas "AIF3" from NASA Glenn coefficients

#### Information

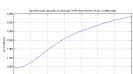


**Modelica.Media.IdealGases.SingleGases.ALF4minus**

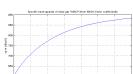
Ideal gas "AlF4-" from NASA Glenn coefficients

**Information****Modelica.Media.IdealGases.SingleGases.ALH**

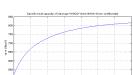
Ideal gas "AlH" from NASA Glenn coefficients

**Information****Modelica.Media.IdealGases.SingleGases.ALHCL**

Ideal gas "AlHCl" from NASA Glenn coefficients

**Information****Modelica.Media.IdealGases.SingleGases.ALHCL2**

Ideal gas "AlHCl2" from NASA Glenn coefficients

**Information****Modelica.Media.IdealGases.SingleGases.ALHF**

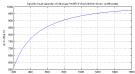
Ideal gas "AlHF" from NASA Glenn coefficients

**Information**

## Modelica.Media.IdealGases.SingleGases.ALHFCL

Ideal gas "AIHFCI" from NASA Glenn coefficients

### Information

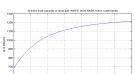


---

## Modelica.Media.IdealGases.SingleGases.ALHF2

Ideal gas "AIHF2" from NASA Glenn coefficients

### Information

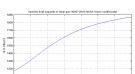


---

## Modelica.Media.IdealGases.SingleGases.ALH2

Ideal gas "AIH2" from NASA Glenn coefficients

### Information

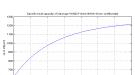


---

## Modelica.Media.IdealGases.SingleGases.ALH2CL

Ideal gas "AIH2CI" from NASA Glenn coefficients

### Information

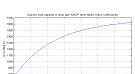


---

## Modelica.Media.IdealGases.SingleGases.ALH2F

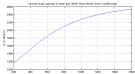
Ideal gas "AIH2F" from NASA Glenn coefficients

### Information

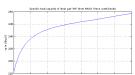


**Modelica.Media.IdealGases.SingleGases.ALH3**

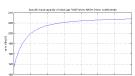
Ideal gas "AIH3" from NASA Glenn coefficients

**Information****Modelica.Media.IdealGases.SingleGases.ALI**

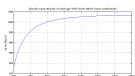
Ideal gas "AI" from NASA Glenn coefficients

**Information****Modelica.Media.IdealGases.SingleGases.ALI2**

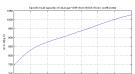
Ideal gas "AI2" from NASA Glenn coefficients

**Information****Modelica.Media.IdealGases.SingleGases.ALI3**

Ideal gas "AI3" from NASA Glenn coefficients

**Information****Modelica.Media.IdealGases.SingleGases.ALN**

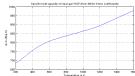
Ideal gas "AIN" from NASA Glenn coefficients

**Information**

**Modelica.Media.IdealGases.SingleGases.ALO**

Ideal gas "ALO" from NASA Glenn coefficients

**Information**

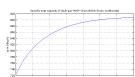


---

**Modelica.Media.IdealGases.SingleGases.ALOplus**

Ideal gas "ALO+" from NASA Glenn coefficients

**Information**

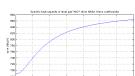


---

**Modelica.Media.IdealGases.SingleGases.ALominus**

Ideal gas "ALO-" from NASA Glenn coefficients

**Information**

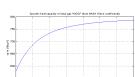


---

**Modelica.Media.IdealGases.SingleGases.ALOCL**

Ideal gas "ALOCl" from NASA Glenn coefficients

**Information**

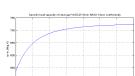


---

**Modelica.Media.IdealGases.SingleGases.ALOCL2**

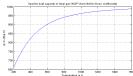
Ideal gas "ALOCl2" from NASA Glenn coefficients

**Information**

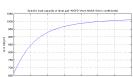


**Modelica.Media.IdealGases.SingleGases.ALOF**

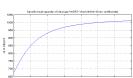
Ideal gas "ALOF" from NASA Glenn coefficients

**Information****Modelica.Media.IdealGases.SingleGases.ALOF2**

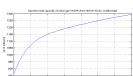
Ideal gas "ALOF2" from NASA Glenn coefficients

**Information****Modelica.Media.IdealGases.SingleGases.ALOF2minus**

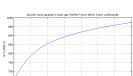
Ideal gas "ALOF2-" from NASA Glenn coefficients

**Information****Modelica.Media.IdealGases.SingleGases.ALOH**

Ideal gas "ALOH" from NASA Glenn coefficients

**Information****Modelica.Media.IdealGases.SingleGases.ALOHCL**

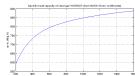
Ideal gas "ALOHCl" from NASA Glenn coefficients

**Information**

### Modelica.Media.IdealGases.SingleGases.ALOHCL2

Ideal gas "ALOHCl2" from NASA Glenn coefficients

#### Information

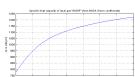


---

### Modelica.Media.IdealGases.SingleGases.ALOHF

Ideal gas "ALOHF" from NASA Glenn coefficients

#### Information

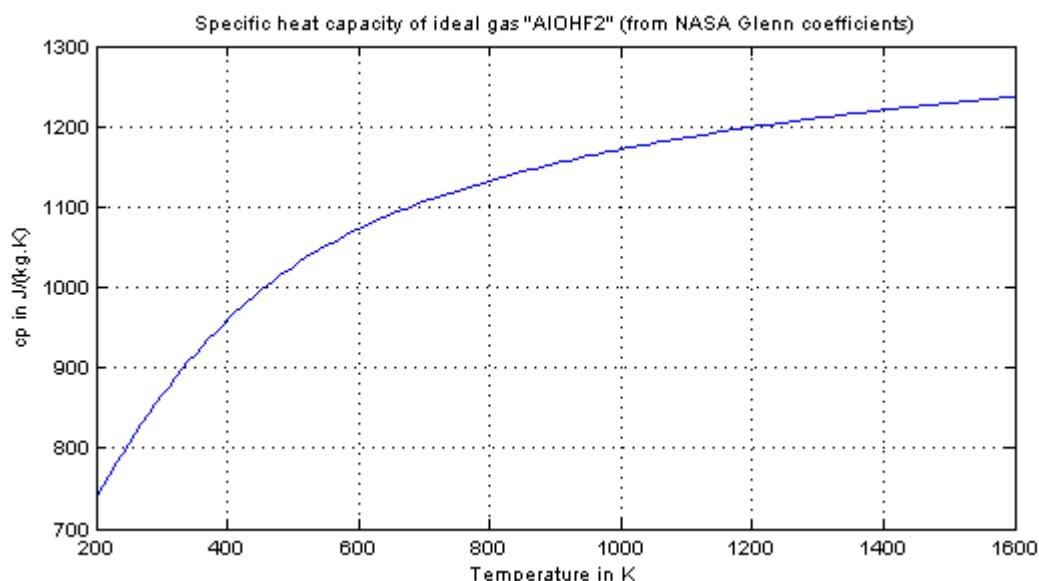


---

### Modelica.Media.IdealGases.SingleGases.ALOHF2

Ideal gas "ALOHF2" from NASA Glenn coefficients

#### Information

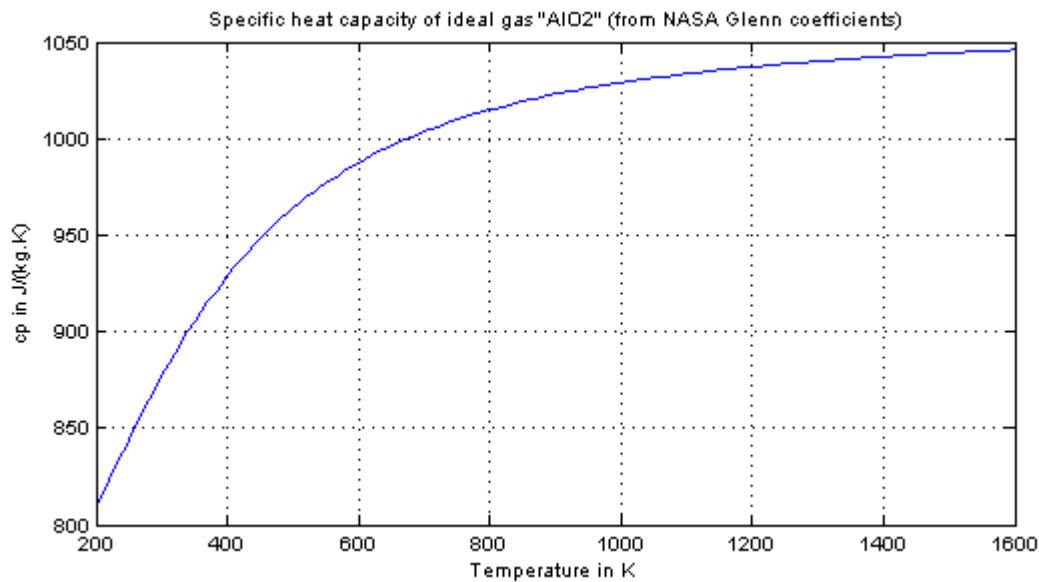


---

### Modelica.Media.IdealGases.SingleGases.ALO2

Ideal gas "ALO2" from NASA Glenn coefficients

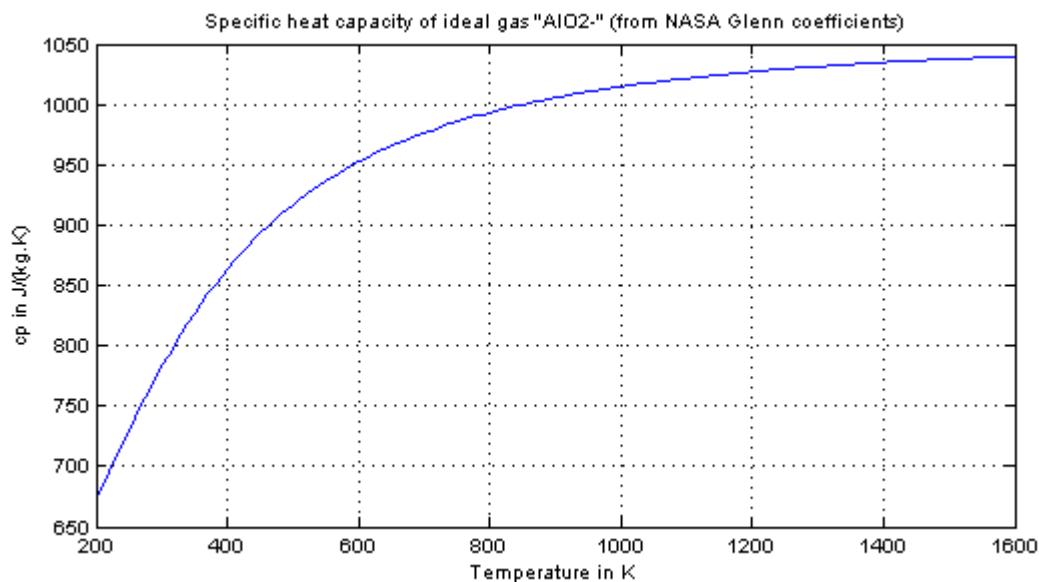
## Information



## Modelica.Media.IdealGases.SingleGases.ALO2minus

Ideal gas "ALO2-" from NASA Glenn coefficients

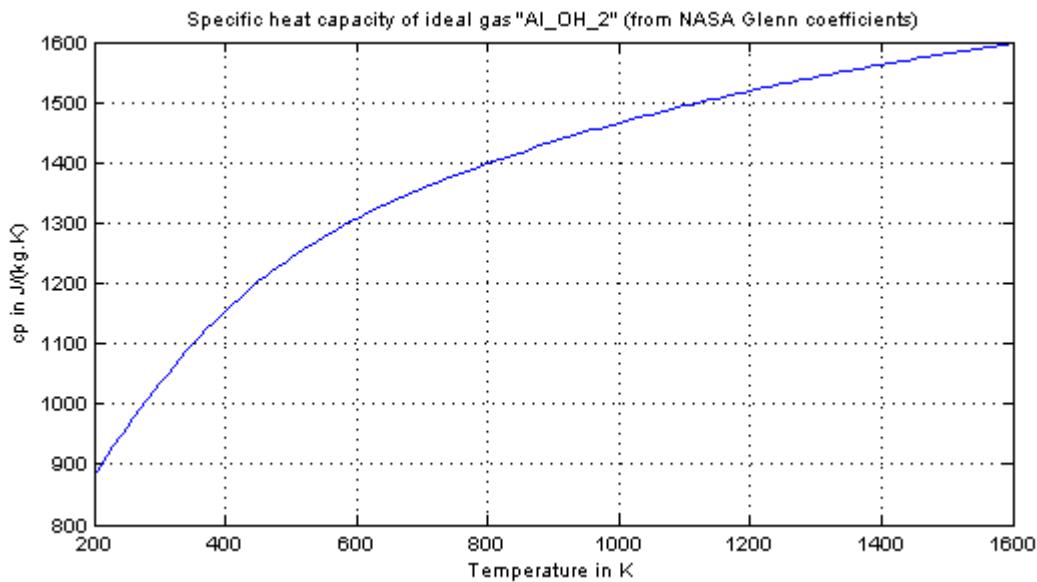
## Information



## Modelica.Media.IdealGases.SingleGases.AL\_OH\_2

Ideal gas "Al\_OH\_2" from NASA Glenn coefficients

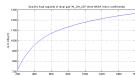
## Information



## Modelica.Media.IdealGases.SingleGases.AL\_OH\_2CL

Ideal gas "AL\_OH\_2CL" from NASA Glenn coefficients

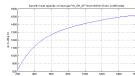
## Information



## Modelica.Media.IdealGases.SingleGases.AL\_OH\_2F

Ideal gas "AL\_OH\_2F" from NASA Glenn coefficients

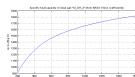
## Information



## Modelica.Media.IdealGases.SingleGases.AL\_OH\_3

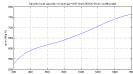
Ideal gas "AL\_OH\_3" from NASA Glenn coefficients

## Information

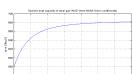


**Modelica.Media.IdealGases.SingleGases.ALS**

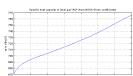
Ideal gas "AIS" from NASA Glenn coefficients

**Information****Modelica.Media.IdealGases.SingleGases.ALS2**

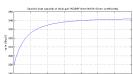
Ideal gas "AIS2" from NASA Glenn coefficients

**Information****Modelica.Media.IdealGases.SingleGases.AL2**

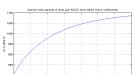
Ideal gas "Al2" from NASA Glenn coefficients

**Information****Modelica.Media.IdealGases.SingleGases.AL2Br6**

Ideal gas "Al2Br6" from NASA Glenn coefficients

**Information****Modelica.Media.IdealGases.SingleGases.AL2C2**

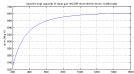
Ideal gas "Al2C2" from NASA Glenn coefficients

**Information**

**Modelica.Media.IdealGases.SingleGases.AL2CL6**

Ideal gas "Al2Cl6" from NASA Glenn coefficients

**Information**

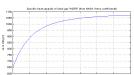


---

**Modelica.Media.IdealGases.SingleGases.AL2F6**

Ideal gas "Al2F6" from NASA Glenn coefficients

**Information**

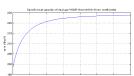


---

**Modelica.Media.IdealGases.SingleGases.AL2I6**

Ideal gas "Al2I6" from NASA Glenn coefficients

**Information**

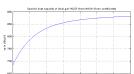


---

**Modelica.Media.IdealGases.SingleGases.AL2O**

Ideal gas "Al2O" from NASA Glenn coefficients

**Information**

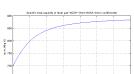


---

**Modelica.Media.IdealGases.SingleGases.AL2Oplus**

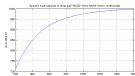
Ideal gas "Al2O+" from NASA Glenn coefficients

**Information**

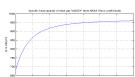


**Modelica.Media.IdealGases.SingleGases.AL2O2**

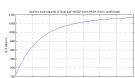
Ideal gas "Al2O2" from NASA Glenn coefficients

**Information****Modelica.Media.IdealGases.SingleGases.AL2O2plus**

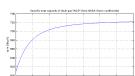
Ideal gas "Al2O2+" from NASA Glenn coefficients

**Information****Modelica.Media.IdealGases.SingleGases.AL2O3**

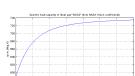
Ideal gas "Al2O3" from NASA Glenn coefficients

**Information****Modelica.Media.IdealGases.SingleGases.AL2S**

Ideal gas "Al2S" from NASA Glenn coefficients

**Information****Modelica.Media.IdealGases.SingleGases.AL2S2**

Ideal gas "Al2S2" from NASA Glenn coefficients

**Information**

### Modelica.Media.IdealGases.SingleGases.Ar

Ideal gas "Ar" from NASA Glenn coefficients

#### Information

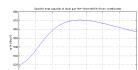


---

### Modelica.Media.IdealGases.SingleGases.Arplus

Ideal gas "Ar+" from NASA Glenn coefficients

#### Information

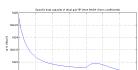


---

### Modelica.Media.IdealGases.SingleGases.B

Ideal gas "B" from NASA Glenn coefficients

#### Information



---

### Modelica.Media.IdealGases.SingleGases.Bplus

Ideal gas "B+" from NASA Glenn coefficients

#### Information

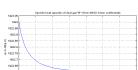


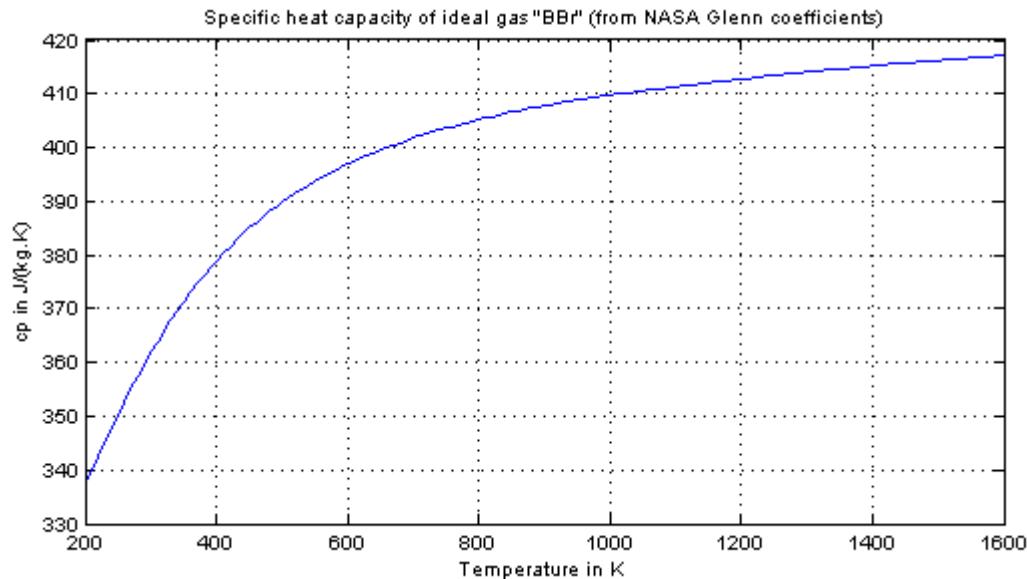
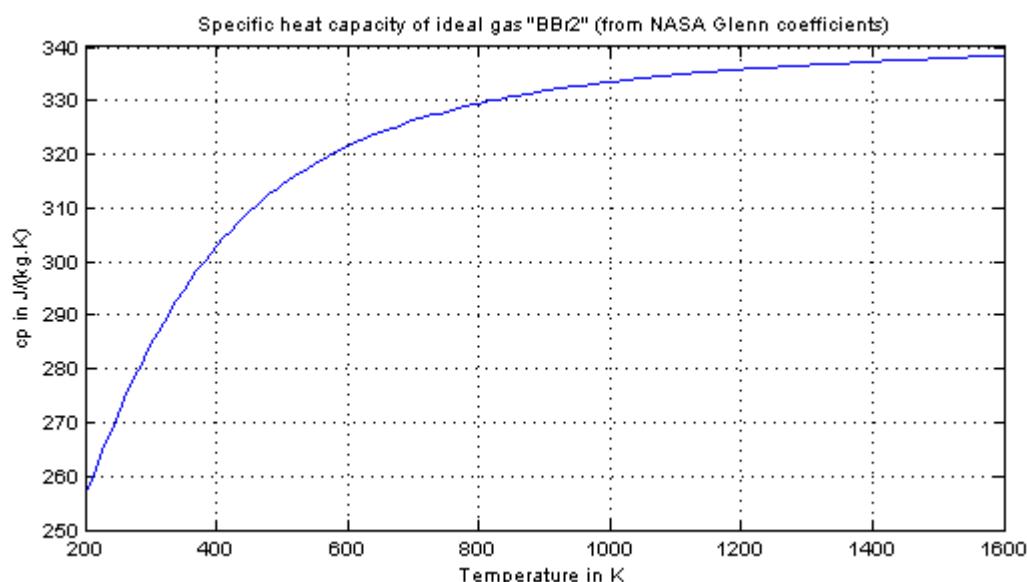
---

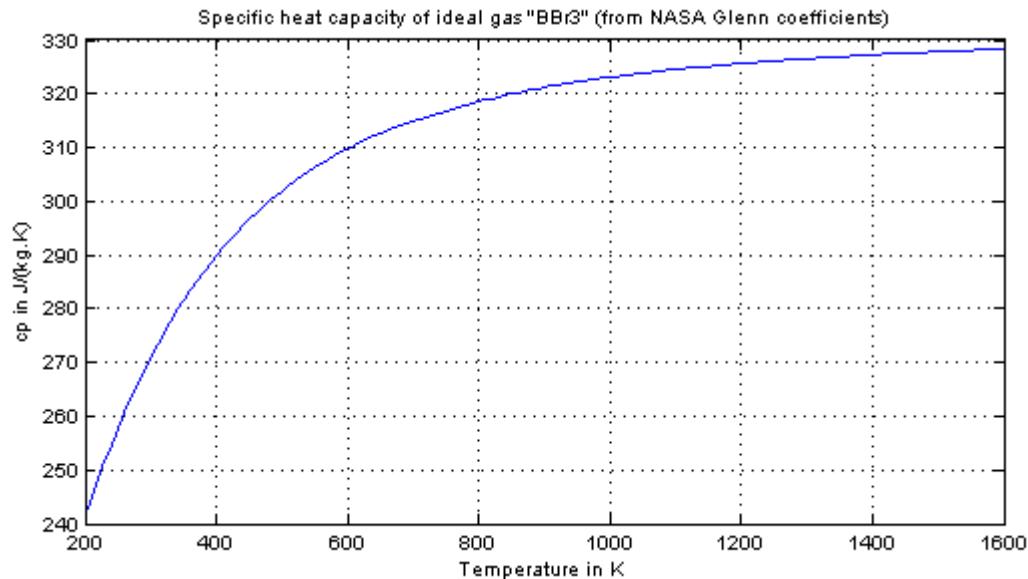
### Modelica.Media.IdealGases.SingleGases.Bminus

Ideal gas "B-" from NASA Glenn coefficients

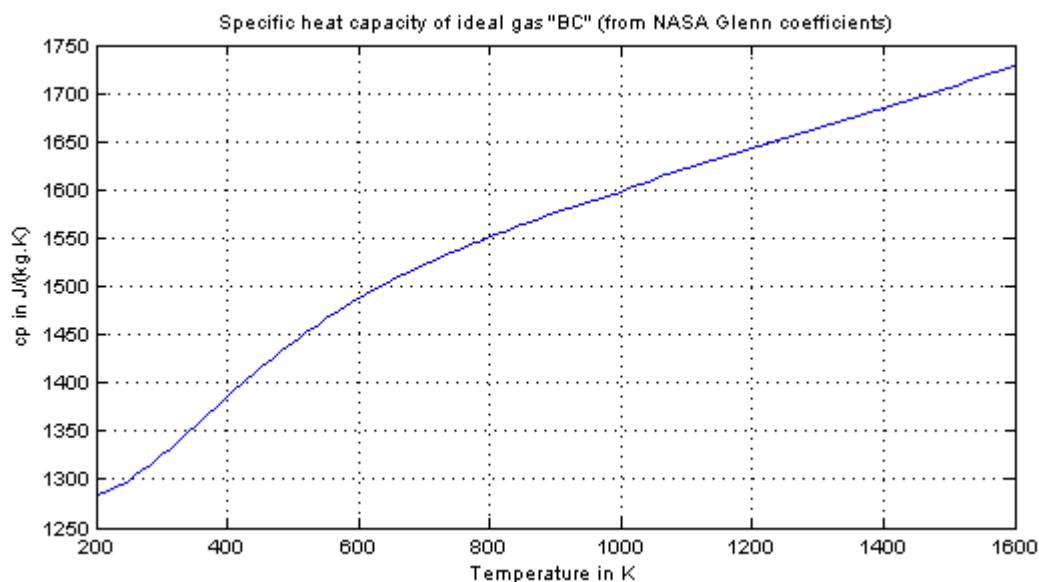
#### Information

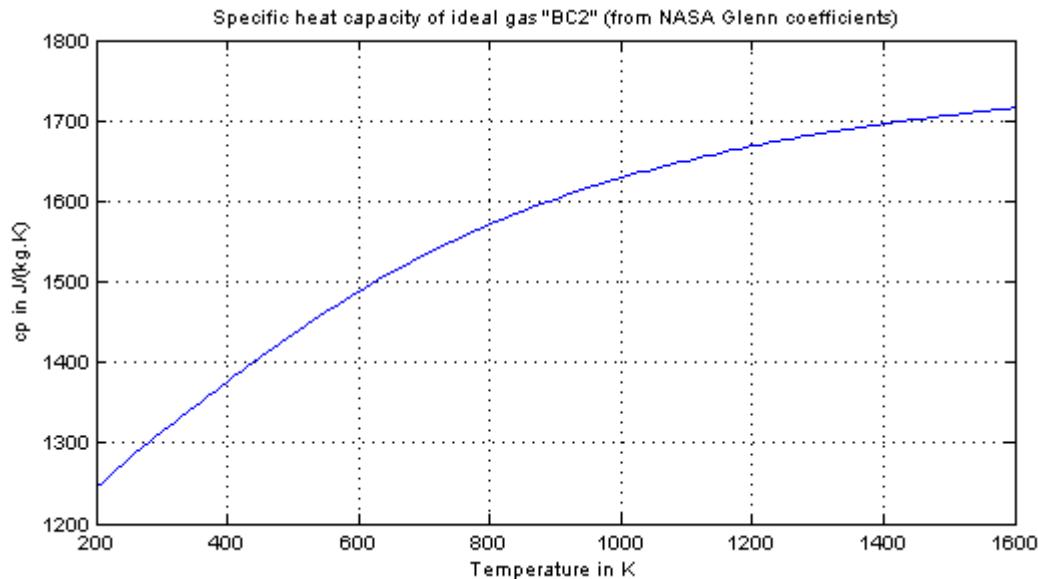
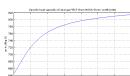
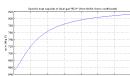


**Modelica.Media.IdealGases.SingleGases.BBr****Ideal gas "BBr" from NASA Glenn coefficients****Information****Modelica.Media.IdealGases.SingleGases.BBr2****Ideal gas "BBr2" from NASA Glenn coefficients****Information**

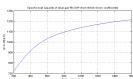
**Modelica.Media.IdealGases.SingleGases.BBr3****Ideal gas "BBr3" from NASA Glenn coefficients****Information**

---

**Modelica.Media.IdealGases.SingleGases.BC****Ideal gas "BC" from NASA Glenn coefficients****Information**

**Modelica.Media.IdealGases.SingleGases.BC2****Ideal gas "BC2" from NASA Glenn coefficients****Information****Modelica.Media.IdealGases.SingleGases.BCL****Ideal gas "BCI" from NASA Glenn coefficients****Information****Modelica.Media.IdealGases.SingleGases.BCLplus****Ideal gas "BCI+" from NASA Glenn coefficients****Information****Modelica.Media.IdealGases.SingleGases.BCLOH****Ideal gas "BCIOH" from NASA Glenn coefficients**

**Information**

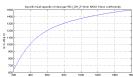


---

**Modelica.Media.IdealGases.SingleGases.BCL\_OH\_2**

Ideal gas "BCI\_OH\_2" from NASA Glenn coefficients

**Information**

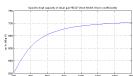


---

**Modelica.Media.IdealGases.SingleGases.BCL2**

Ideal gas "BCI2" from NASA Glenn coefficients

**Information**

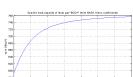


---

**Modelica.Media.IdealGases.SingleGases.BCL2plus**

Ideal gas "BCI2+" from NASA Glenn coefficients

**Information**

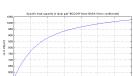


---

**Modelica.Media.IdealGases.SingleGases.BCL2OH**

Ideal gas "BCI2OH" from NASA Glenn coefficients

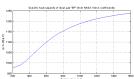
**Information**



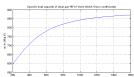
---

**Modelica.Media.IdealGases.SingleGases.BF**

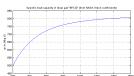
Ideal gas "BF" from NASA Glenn coefficients

**Information****Modelica.Media.IdealGases.SingleGases.BFCL**

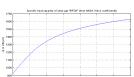
Ideal gas "BFCL" from NASA Glenn coefficients

**Information****Modelica.Media.IdealGases.SingleGases.BFCL2**

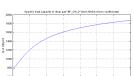
Ideal gas "BFCL2" from NASA Glenn coefficients

**Information****Modelica.Media.IdealGases.SingleGases.BFOH**

Ideal gas "BFOH" from NASA Glenn coefficients

**Information****Modelica.Media.IdealGases.SingleGases.BF\_OH\_2**

Ideal gas "BF\_OH\_2" from NASA Glenn coefficients

**Information****Modelica.Media.IdealGases.SingleGases.BF2**

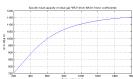
Ideal gas "BF2" from NASA Glenn coefficients

---

## 1294 Modelica.Media.IdealGases.SingleGases.BF2

---

### Information

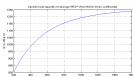


---

## Modelica.Media.IdealGases.SingleGases.BF2plus

Ideal gas "BF2+" from NASA Glenn coefficients

### Information

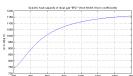


---

## Modelica.Media.IdealGases.SingleGases.BF2minus

Ideal gas "BF2-" from NASA Glenn coefficients

### Information

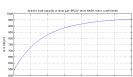


---

## Modelica.Media.IdealGases.SingleGases.BF2CL

Ideal gas "BF2CI" from NASA Glenn coefficients

### Information

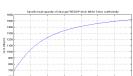


---

## Modelica.Media.IdealGases.SingleGases.BF2OH

Ideal gas "BF2OH" from NASA Glenn coefficients

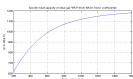
### Information



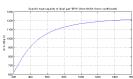
---

## Modelica.Media.IdealGases.SingleGases.BF3

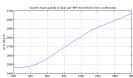
Ideal gas "BF3" from NASA Glenn coefficients

**Information****Modelica.Media.IdealGases.SingleGases.BF4minus**

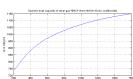
Ideal gas "BF4-" from NASA Glenn coefficients

**Information****Modelica.Media.IdealGases.SingleGases.BH**

Ideal gas "BH" from NASA Glenn coefficients

**Information****Modelica.Media.IdealGases.SingleGases.BHCl**

Ideal gas "BHCl" from NASA Glenn coefficients

**Information****Modelica.Media.IdealGases.SingleGases.BHCl2**

Ideal gas "BHCl2" from NASA Glenn coefficients

**Information****Modelica.Media.IdealGases.SingleGases.BHF**

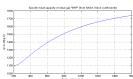
Ideal gas "BHF" from NASA Glenn coefficients

---

## 1296 Modelica.Media.IdealGases.SingleGases.BHF

---

### Information

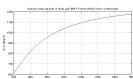


---

## Modelica.Media.IdealGases.SingleGases.BHFCL

Ideal gas "BHFCL" from NASA Glenn coefficients

### Information

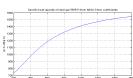


---

## Modelica.Media.IdealGases.SingleGases.BHF2

Ideal gas "BHF2" from NASA Glenn coefficients

### Information

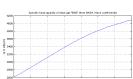


---

## Modelica.Media.IdealGases.SingleGases.BH2

Ideal gas "BH2" from NASA Glenn coefficients

### Information

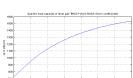


---

## Modelica.Media.IdealGases.SingleGases.BH2CL

Ideal gas "BH2CL" from NASA Glenn coefficients

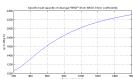
### Information



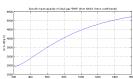
---

## Modelica.Media.IdealGases.SingleGases.BH2F

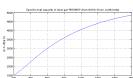
Ideal gas "BH2F" from NASA Glenn coefficients

**Information****Modelica.Media.IdealGases.SingleGases.BH3**

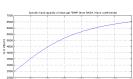
Ideal gas "BH3" from NASA Glenn coefficients

**Information****Modelica.Media.IdealGases.SingleGases.BH3NH3**

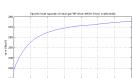
Ideal gas "BH3NH3" from NASA Glenn coefficients

**Information****Modelica.Media.IdealGases.SingleGases.BH4**

Ideal gas "BH4" from NASA Glenn coefficients

**Information****Modelica.Media.IdealGases.SingleGases.BI**

Ideal gas "BI" from NASA Glenn coefficients

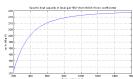
**Information****Modelica.Media.IdealGases.SingleGases.BI2**

Ideal gas "BI2" from NASA Glenn coefficients

## 1298 Modelica.Media.IdealGases.SingleGases.BI2

---

### Information

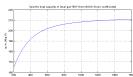


---

## Modelica.Media.IdealGases.SingleGases.BI3

Ideal gas "BI3" from NASA Glenn coefficients

### Information

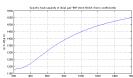


---

## Modelica.Media.IdealGases.SingleGases.BN

Ideal gas "BN" from NASA Glenn coefficients

### Information

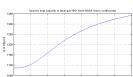


---

## Modelica.Media.IdealGases.SingleGases.BO

Ideal gas "BO" from NASA Glenn coefficients

### Information

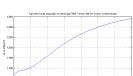


---

## Modelica.Media.IdealGases.SingleGases.BOminus

Ideal gas "BO-" from NASA Glenn coefficients

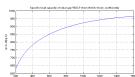
### Information



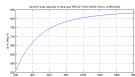
---

## Modelica.Media.IdealGases.SingleGases.BOCL

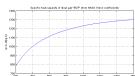
Ideal gas "BOCI" from NASA Glenn coefficients

**Information****Modelica.Media.IdealGases.SingleGases.BOCL2**

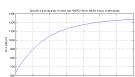
Ideal gas "BOCL2" from NASA Glenn coefficients

**Information****Modelica.Media.IdealGases.SingleGases.BOF**

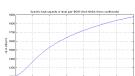
Ideal gas "BOF" from NASA Glenn coefficients

**Information****Modelica.Media.IdealGases.SingleGases.BOF2**

Ideal gas "BOF2" from NASA Glenn coefficients

**Information****Modelica.Media.IdealGases.SingleGases.BOH**

Ideal gas "BOH" from NASA Glenn coefficients

**Information****Modelica.Media.IdealGases.SingleGases.BO2**

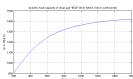
Ideal gas "BO2" from NASA Glenn coefficients

---

## 1300 Modelica.Media.IdealGases.SingleGases.BO2

---

### Information

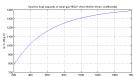


---

## Modelica.Media.IdealGases.SingleGases.BO2minus

Ideal gas "BO2-" from NASA Glenn coefficients

### Information

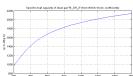


---

## Modelica.Media.IdealGases.SingleGases.B\_OH\_2

Ideal gas "B\_OH\_2" from NASA Glenn coefficients

### Information

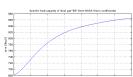


---

## Modelica.Media.IdealGases.SingleGases.BS

Ideal gas "BS" from NASA Glenn coefficients

### Information

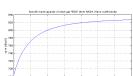


---

## Modelica.Media.IdealGases.SingleGases.BS2

Ideal gas "BS2" from NASA Glenn coefficients

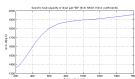
### Information



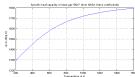
---

## Modelica.Media.IdealGases.SingleGases.B2

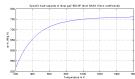
Ideal gas "B2" from NASA Glenn coefficients

**Information****Modelica.Media.IdealGases.SingleGases.B2C**

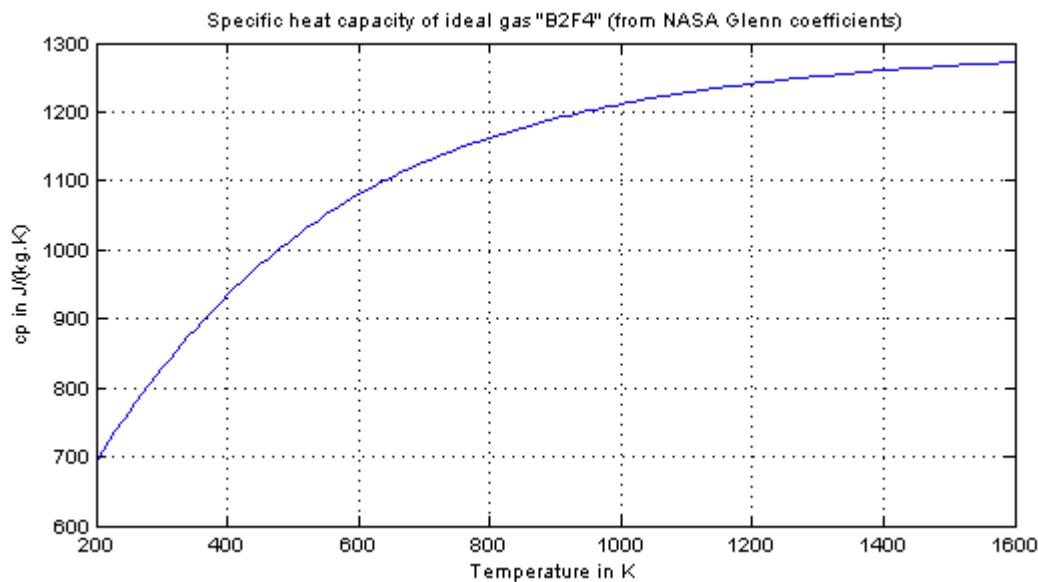
Ideal gas "B2C" from NASA Glenn coefficients

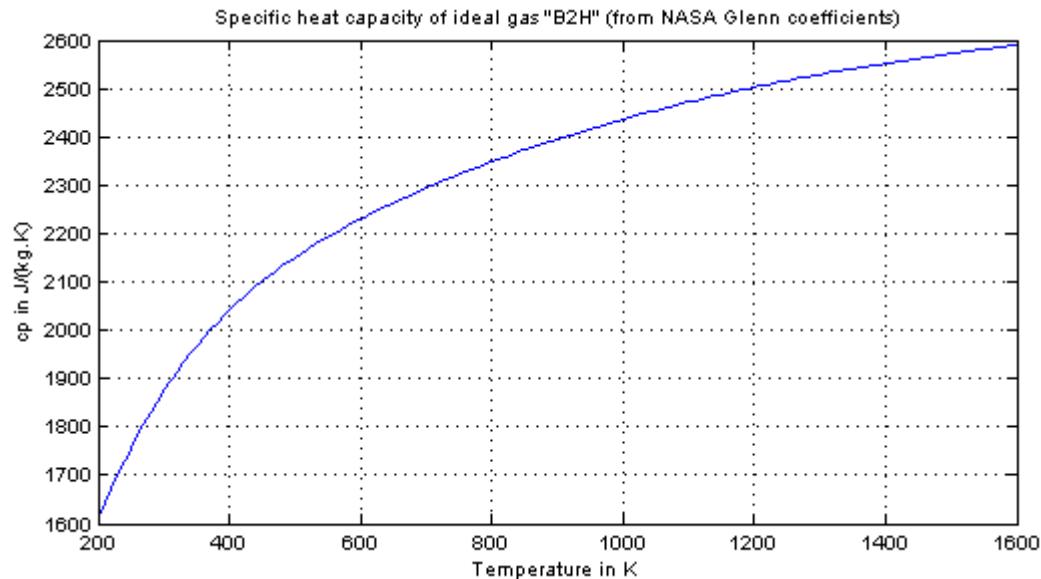
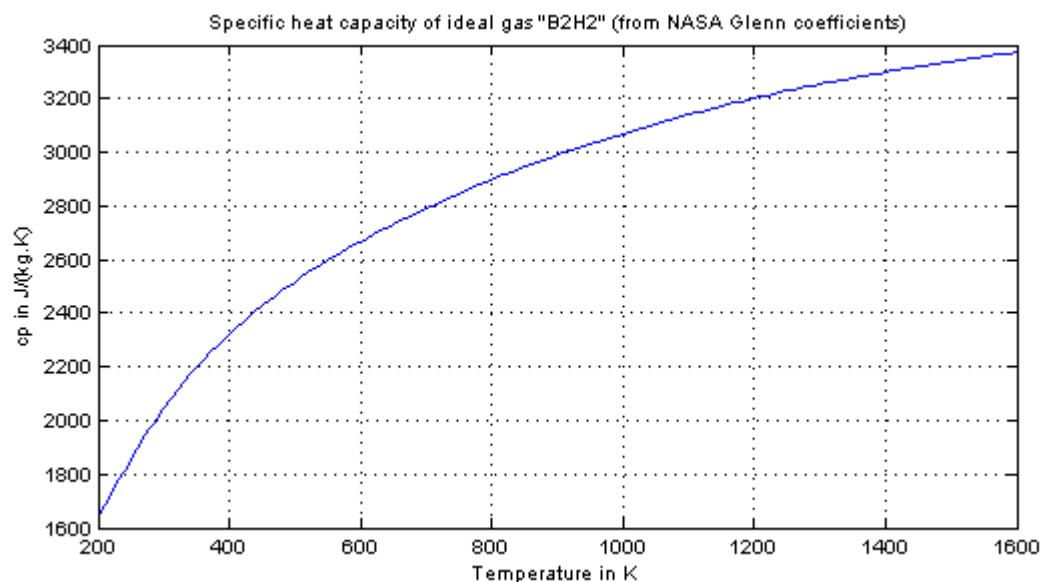
**Information****Modelica.Media.IdealGases.SingleGases.B2Cl4**

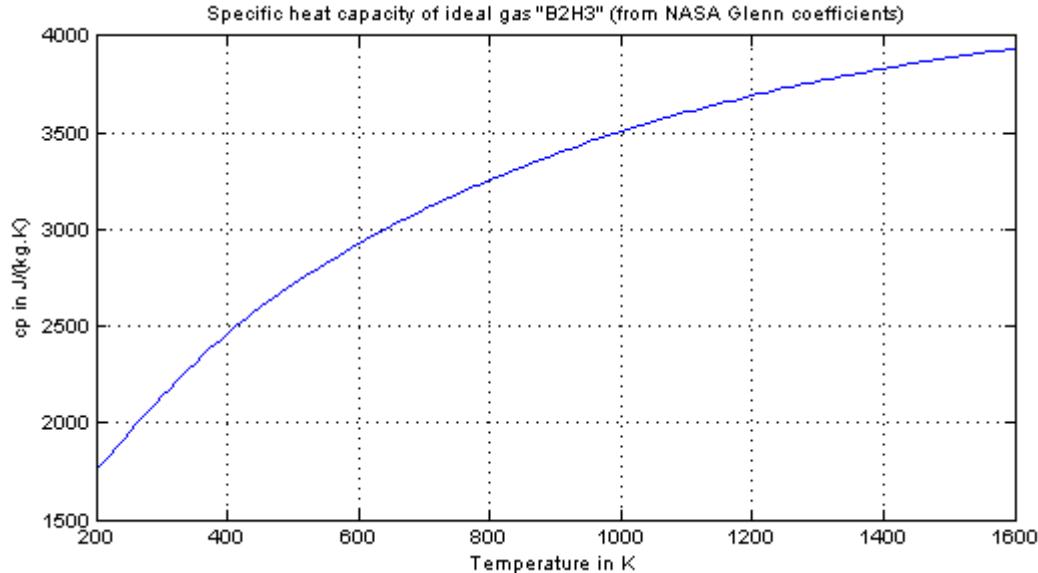
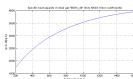
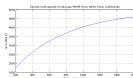
Ideal gas "B2Cl4" from NASA Glenn coefficients

**Information****Modelica.Media.IdealGases.SingleGases.B2F4**

Ideal gas "B2F4" from NASA Glenn coefficients

**Information**

**Modelica.Media.IdealGases.SingleGases.B2H****Ideal gas "B2H" from NASA Glenn coefficients****Information****Modelica.Media.IdealGases.SingleGases.B2H2****Ideal gas "B2H2" from NASA Glenn coefficients****Information**

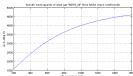
**Modelica.Media.IdealGases.SingleGases.B2H3****Ideal gas "B2H3" from NASA Glenn coefficients****Information****Modelica.Media.IdealGases.SingleGases.B2H3\_db****Ideal gas "B2H3\_db" from NASA Glenn coefficients****Information****Modelica.Media.IdealGases.SingleGases.B2H4****Ideal gas "B2H4" from NASA Glenn coefficients****Information****Modelica.Media.IdealGases.SingleGases.B2H4\_db****Ideal gas "B2H4\_db" from NASA Glenn coefficients**

---

## **1304 Modelica.Media.IdealGases.SingleGases.B2H4\_db**

---

### **Information**

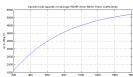


---

## **Modelica.Media.IdealGases.SingleGases.B2H5**

Ideal gas "B2H5" from NASA Glenn coefficients

### **Information**

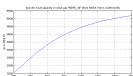


---

## **Modelica.Media.IdealGases.SingleGases.B2H5\_db**

Ideal gas "B2H5\_db" from NASA Glenn coefficients

### **Information**

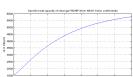


---

## **Modelica.Media.IdealGases.SingleGases.B2H6**

Ideal gas "B2H6" from NASA Glenn coefficients

### **Information**

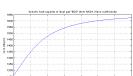


---

## **Modelica.Media.IdealGases.SingleGases.B2O**

Ideal gas "B2O" from NASA Glenn coefficients

### **Information**

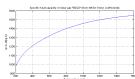


---

## **Modelica.Media.IdealGases.SingleGases.B2O2**

Ideal gas "B2O2" from NASA Glenn coefficients

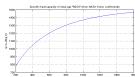
## Information



## Modelica.Media.IdealGases.SingleGases.B2O3

Ideal gas "B2O3" from NASA Glenn coefficients

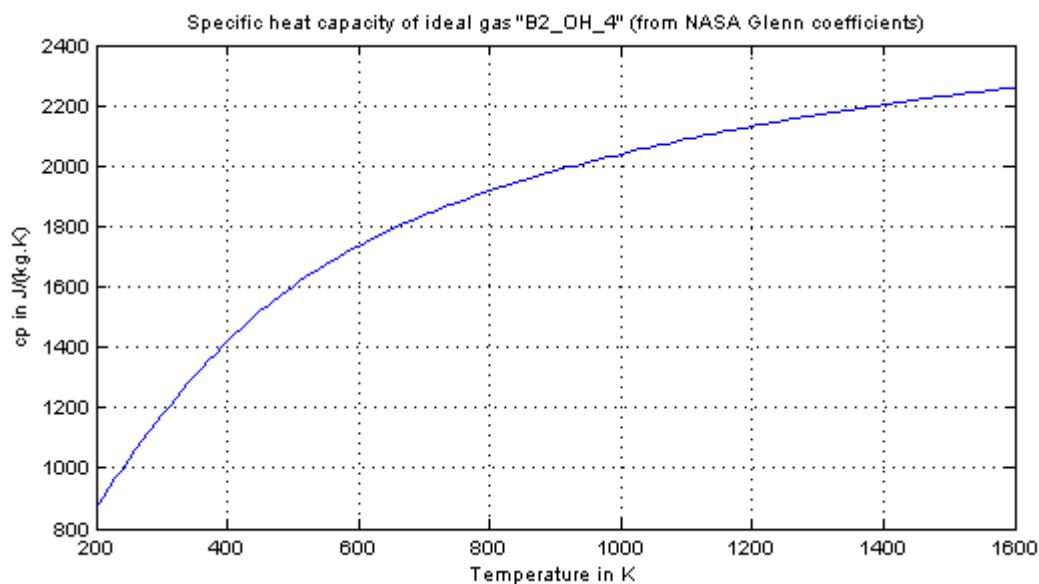
## Information



## Modelica.Media.IdealGases.SingleGases.B2\_OH\_4

Ideal gas "B2\_OH\_4" from NASA Glenn coefficients

## Information



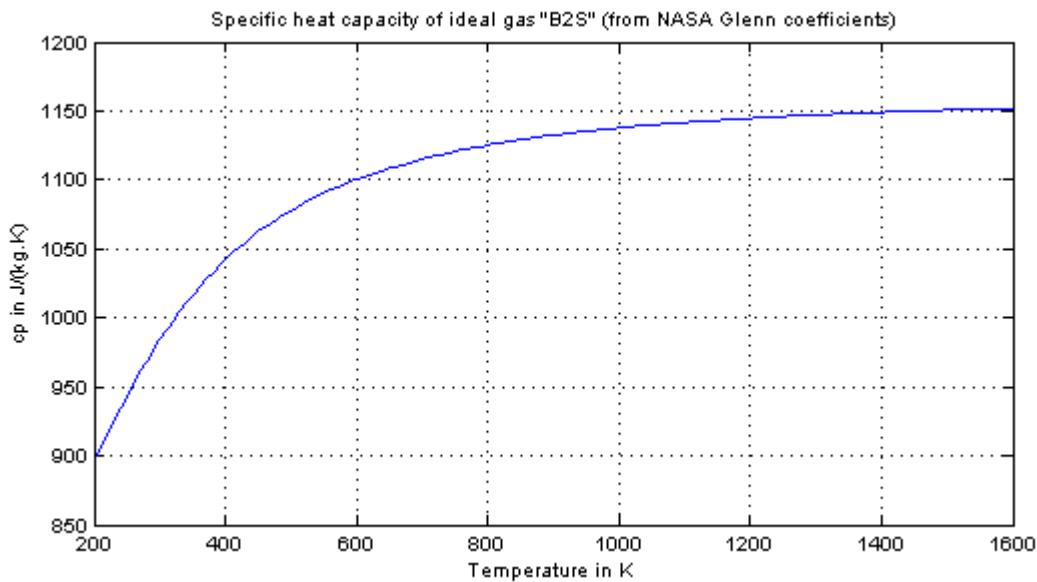
## Modelica.Media.IdealGases.SingleGases.B2S

Ideal gas "B2S" from NASA Glenn coefficients

## 1306 Modelica.Media.IdealGases.SingleGases.B2S

---

### Information

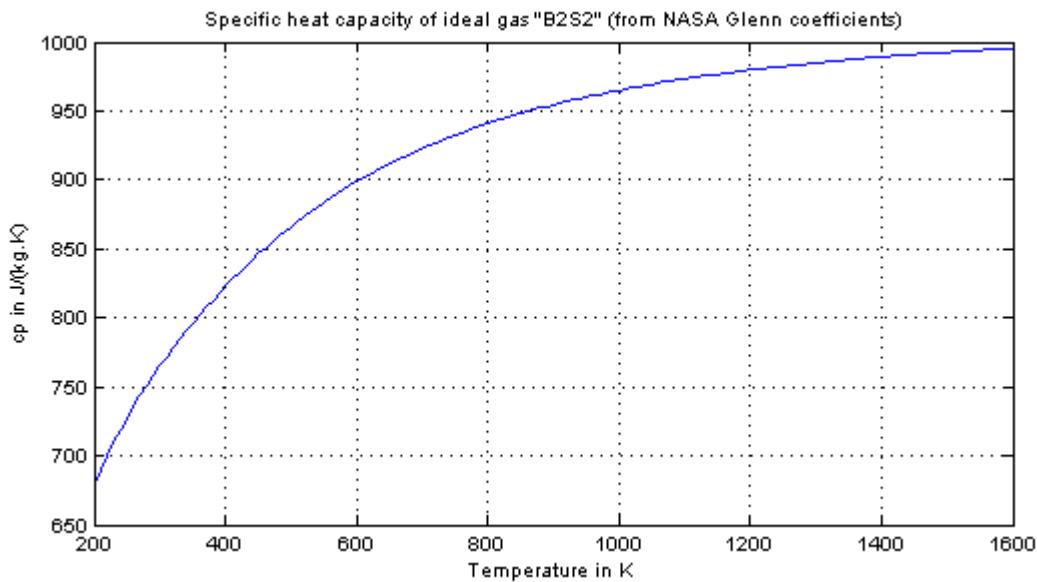


---

## Modelica.Media.IdealGases.SingleGases.B2S2

Ideal gas "B2S2" from NASA Glenn coefficients

### Information

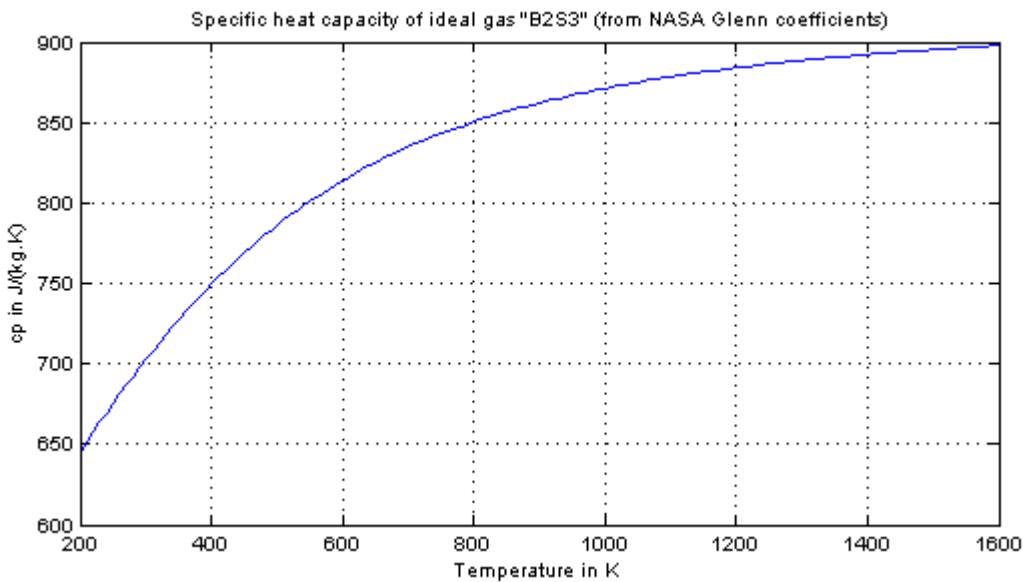


---

## Modelica.Media.IdealGases.SingleGases.B2S3

Ideal gas "B2S3" from NASA Glenn coefficients

## Information



## Modelica.Media.IdealGases.SingleGases.B3H7\_C2v

Ideal gas "B3H7\_C2v" from NASA Glenn coefficients

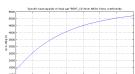
## Information



## Modelica.Media.IdealGases.SingleGases.B3H7\_Cs

Ideal gas "B3H7\_Cs" from NASA Glenn coefficients

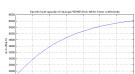
## Information



## Modelica.Media.IdealGases.SingleGases.B3H9

Ideal gas "B3H9" from NASA Glenn coefficients

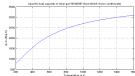
## Information



## Modelica.Media.IdealGases.SingleGases.B3N3H6

Ideal gas "B3N3H6" from NASA Glenn coefficients

### Information

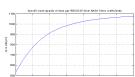


---

## Modelica.Media.IdealGases.SingleGases.B3O3CL3

Ideal gas "B3O3Cl3" from NASA Glenn coefficients

### Information

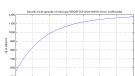


---

## Modelica.Media.IdealGases.SingleGases.B3O3FCL2

Ideal gas "B3O3FCI2" from NASA Glenn coefficients

### Information

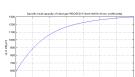


---

## Modelica.Media.IdealGases.SingleGases.B3O3F2CL

Ideal gas "B3O3F2CI" from NASA Glenn coefficients

### Information

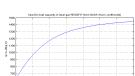


---

## Modelica.Media.IdealGases.SingleGases.B3O3F3

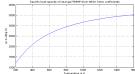
Ideal gas "B3O3F3" from NASA Glenn coefficients

### Information

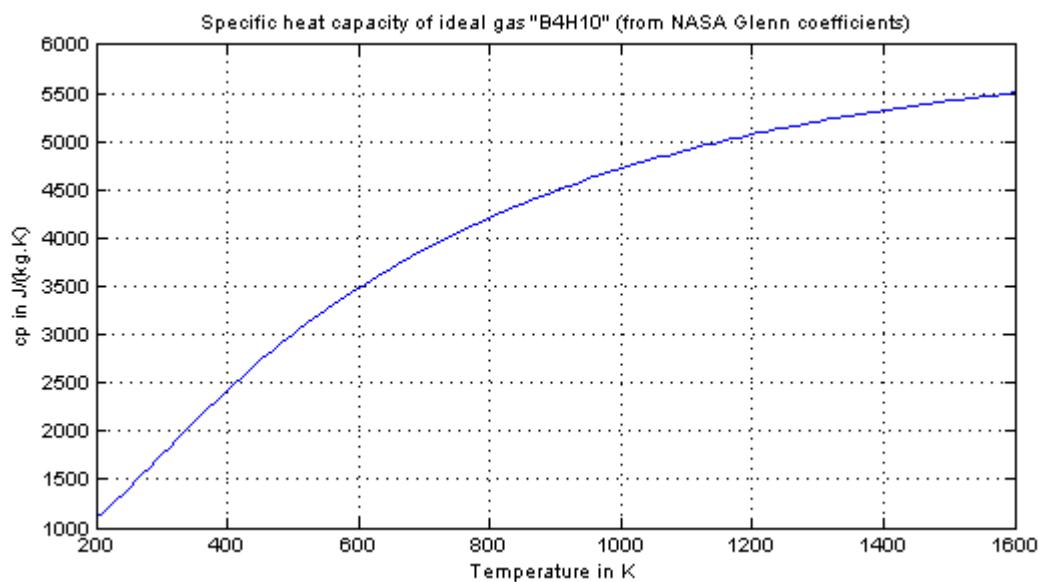


**Modelica.Media.IdealGases.SingleGases.B4H4**

Ideal gas "B4H4" from NASA Glenn coefficients

**Information****Modelica.Media.IdealGases.SingleGases.B4H10**

Ideal gas "B4H10" from NASA Glenn coefficients

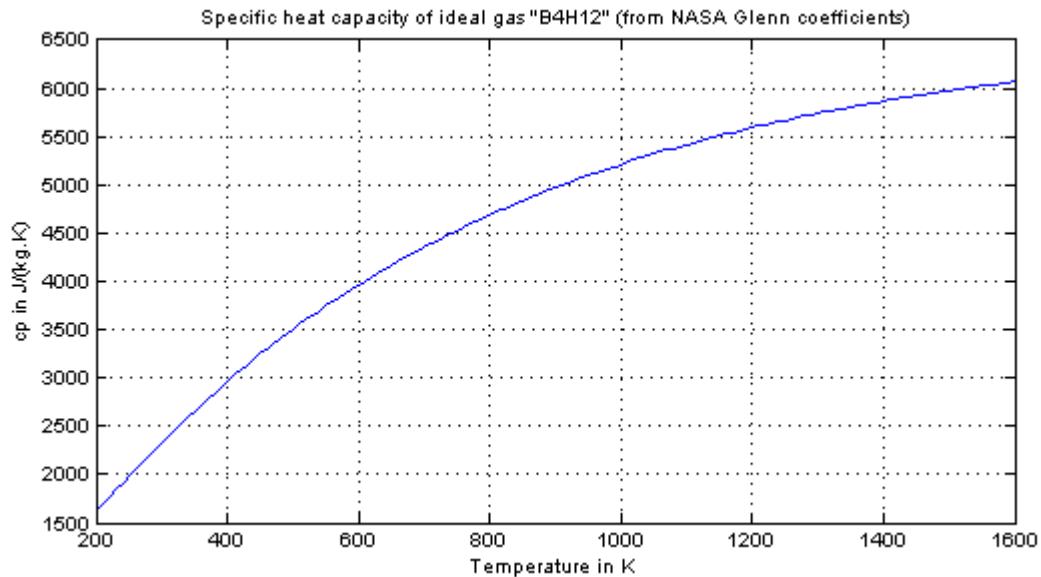
**Information****Modelica.Media.IdealGases.SingleGases.B4H12**

Ideal gas "B4H12" from NASA Glenn coefficients

## 1310 Modelica.Media.IdealGases.SingleGases.B4H12

---

### Information

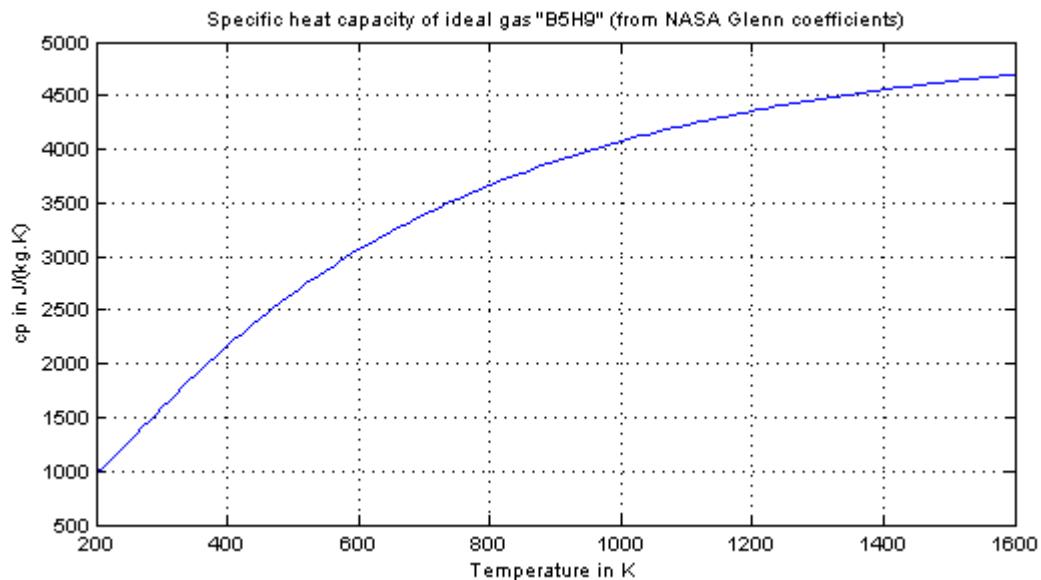


---

## Modelica.Media.IdealGases.SingleGases.B5H9

Ideal gas "B5H9" from NASA Glenn coefficients

### Information

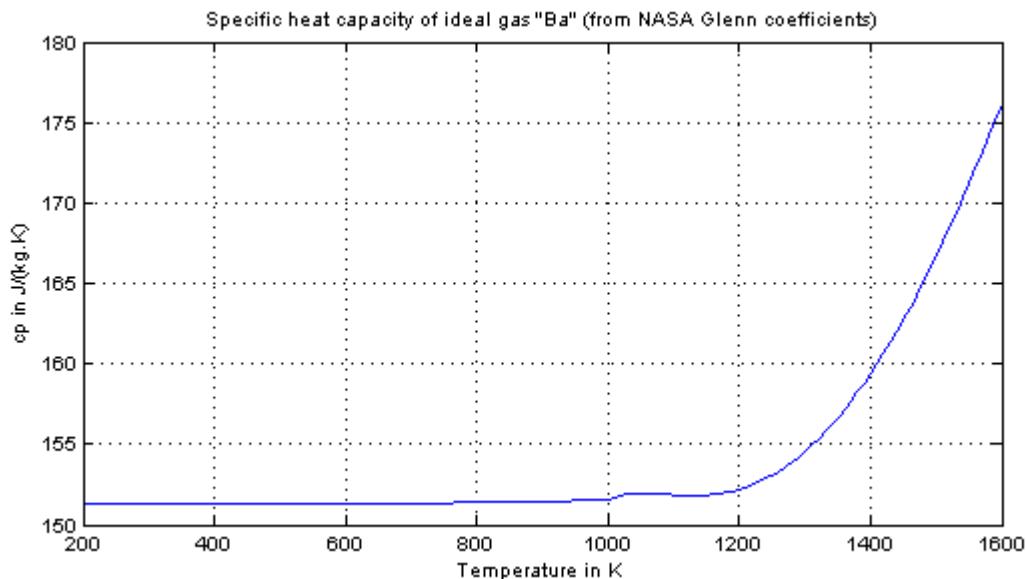


---

## Modelica.Media.IdealGases.SingleGases.Ba

Ideal gas "Ba" from NASA Glenn coefficients

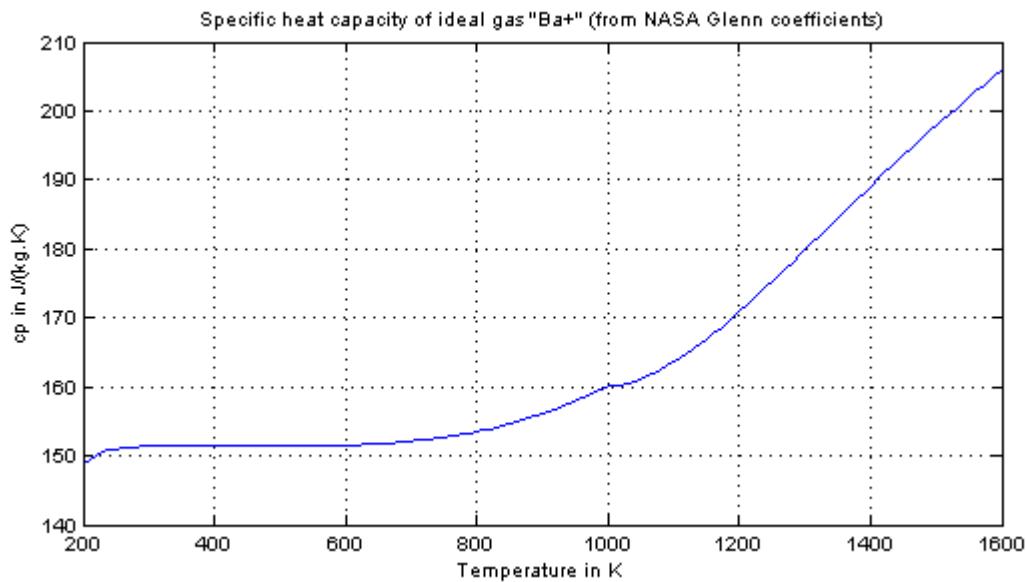
## Information



## Modelica.Media.IdealGases.SingleGases.Baplus

Ideal gas "Ba+" from NASA Glenn coefficients

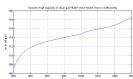
## Information



## Modelica.Media.IdealGases.SingleGases.BaBr

Ideal gas "BaBr" from NASA Glenn coefficients

**Information**

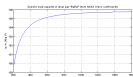


---

**Modelica.Media.IdealGases.SingleGases.BaBr2**

Ideal gas "BaBr2" from NASA Glenn coefficients

**Information**

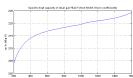


---

**Modelica.Media.IdealGases.SingleGases.BaCL**

Ideal gas "BaCl" from NASA Glenn coefficients

**Information**



---

**Modelica.Media.IdealGases.SingleGases.BaCLplus**

Ideal gas "BaCl+" from NASA Glenn coefficients

**Information**

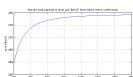


---

**Modelica.Media.IdealGases.SingleGases.BaCL2**

Ideal gas "BaCl2" from NASA Glenn coefficients

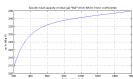
**Information**



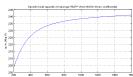
---

**Modelica.Media.IdealGases.SingleGases.BaF**

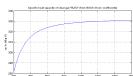
Ideal gas "BaF" from NASA Glenn coefficients

**Information****Modelica.Media.IdealGases.SingleGases.BaFplus**

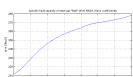
Ideal gas "BaF+" from NASA Glenn coefficients

**Information****Modelica.Media.IdealGases.SingleGases.BaF2**

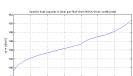
Ideal gas "BaF2" from NASA Glenn coefficients

**Information****Modelica.Media.IdealGases.SingleGases.BaH**

Ideal gas "BaH" from NASA Glenn coefficients

**Information****Modelica.Media.IdealGases.SingleGases.Bal**

Ideal gas "Bal" from NASA Glenn coefficients

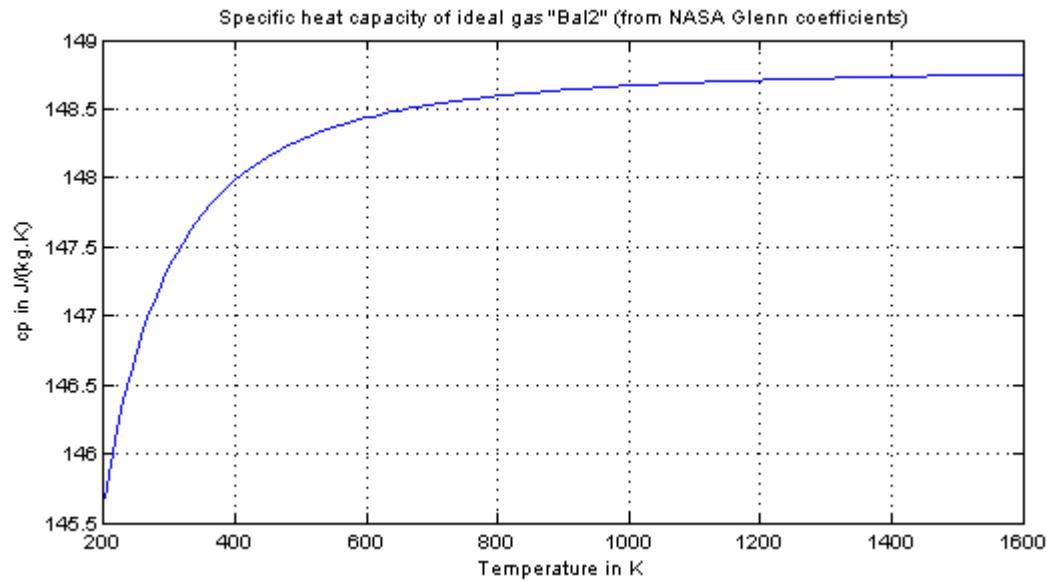
**Information****Modelica.Media.IdealGases.SingleGases.Bal2**

Ideal gas "Bal2" from NASA Glenn coefficients

## 1314 Modelica.Media.IdealGases.SingleGases.BaI2

---

### Information

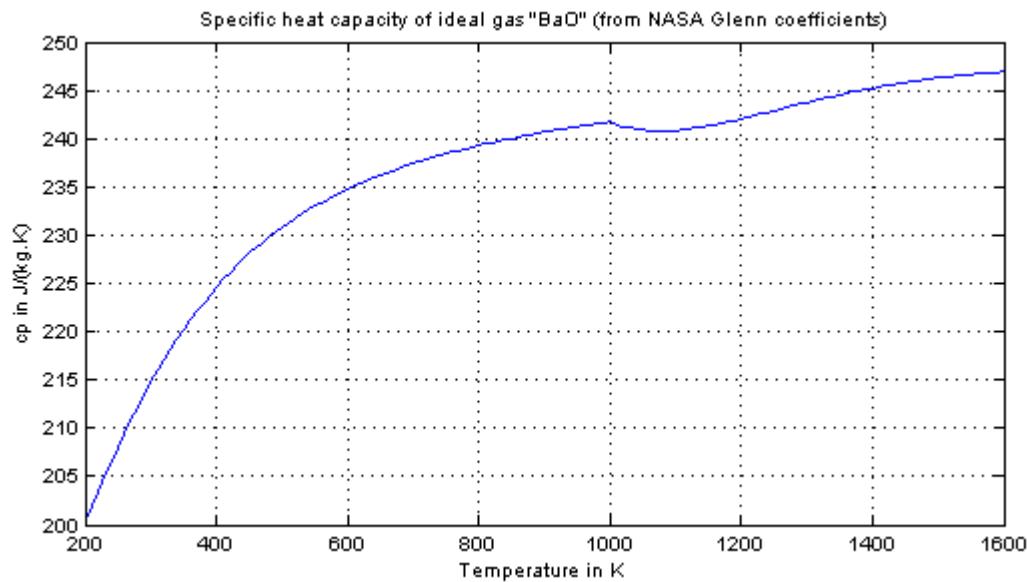


---

## Modelica.Media.IdealGases.SingleGases.BaO

Ideal gas "BaO" from NASA Glenn coefficients

### Information

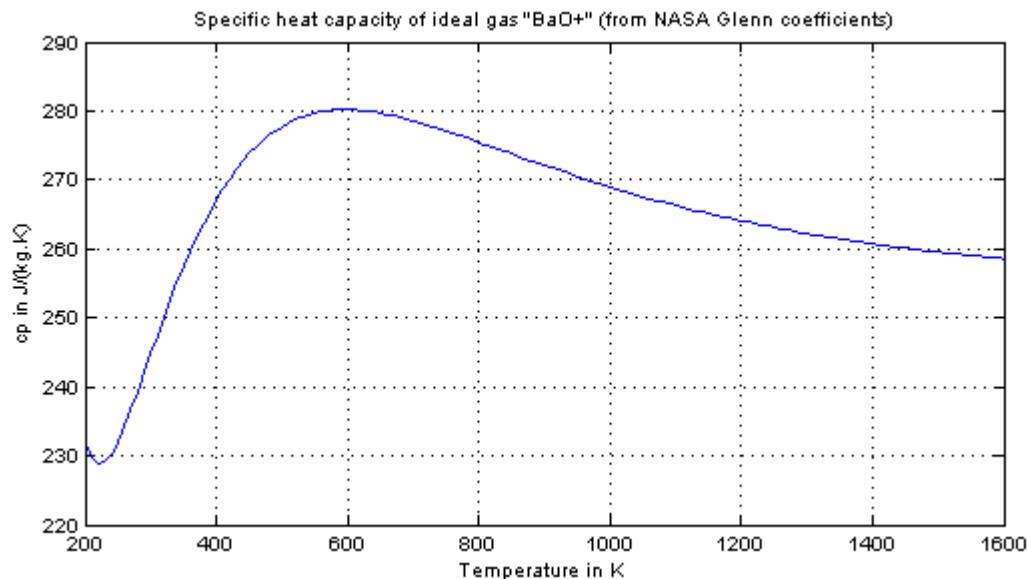


---

## Modelica.Media.IdealGases.SingleGases.BaOplus

Ideal gas "BaO+" from NASA Glenn coefficients

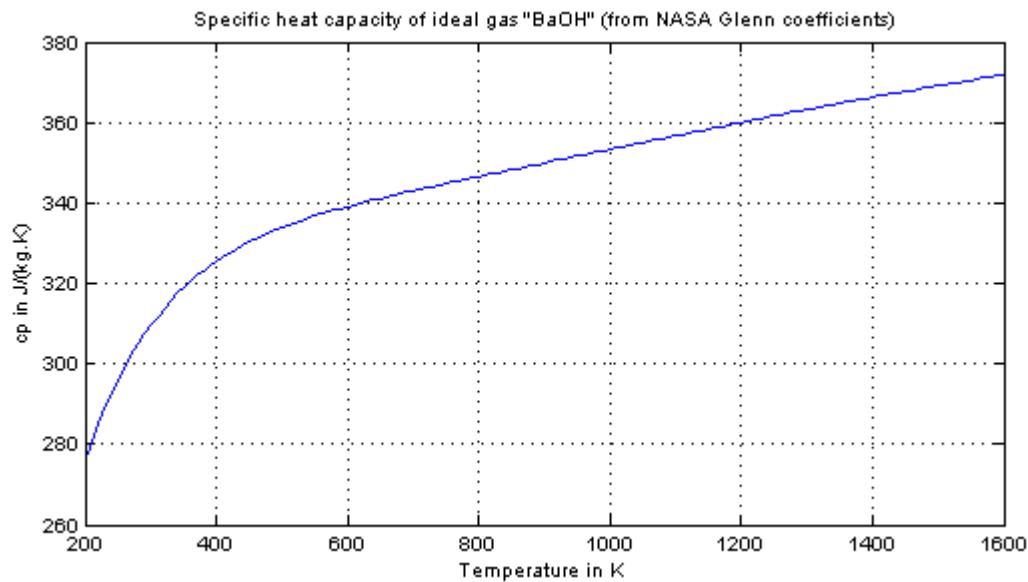
## Information



## Modelica.Media.IdealGases.SingleGases.BaOH

Ideal gas "BaOH" from NASA Glenn coefficients

## Information



## Modelica.Media.IdealGases.SingleGases.BaOHplus

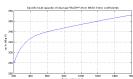
Ideal gas "BaOH+" from NASA Glenn coefficients

---

## 1316 Modelica.Media.IdealGases.SingleGases.BaOHplus

---

### Information

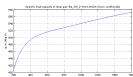


---

## Modelica.Media.IdealGases.SingleGases.Ba\_OH\_2

Ideal gas "Ba\_OH\_2" from NASA Glenn coefficients

### Information

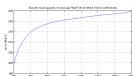


---

## Modelica.Media.IdealGases.SingleGases.BaS

Ideal gas "BaS" from NASA Glenn coefficients

### Information

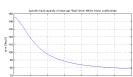


---

## Modelica.Media.IdealGases.SingleGases.Ba2

Ideal gas "Ba2" from NASA Glenn coefficients

### Information



---

## Modelica.Media.IdealGases.SingleGases.Be

Ideal gas "Be" from NASA Glenn coefficients

### Information

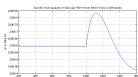


---

## Modelica.Media.IdealGases.SingleGases.Beplus

Ideal gas "Be+" from NASA Glenn coefficients

**Information**

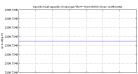


---

**Modelica.Media.IdealGases.SingleGases.Beplusplus**

Ideal gas "Be++" from NASA Glenn coefficients

**Information**

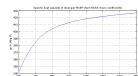


---

**Modelica.Media.IdealGases.SingleGases.BeBr**

Ideal gas "BeBr" from NASA Glenn coefficients

**Information**

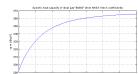


---

**Modelica.Media.IdealGases.SingleGases.BeBr2**

Ideal gas "BeBr2" from NASA Glenn coefficients

**Information**



---

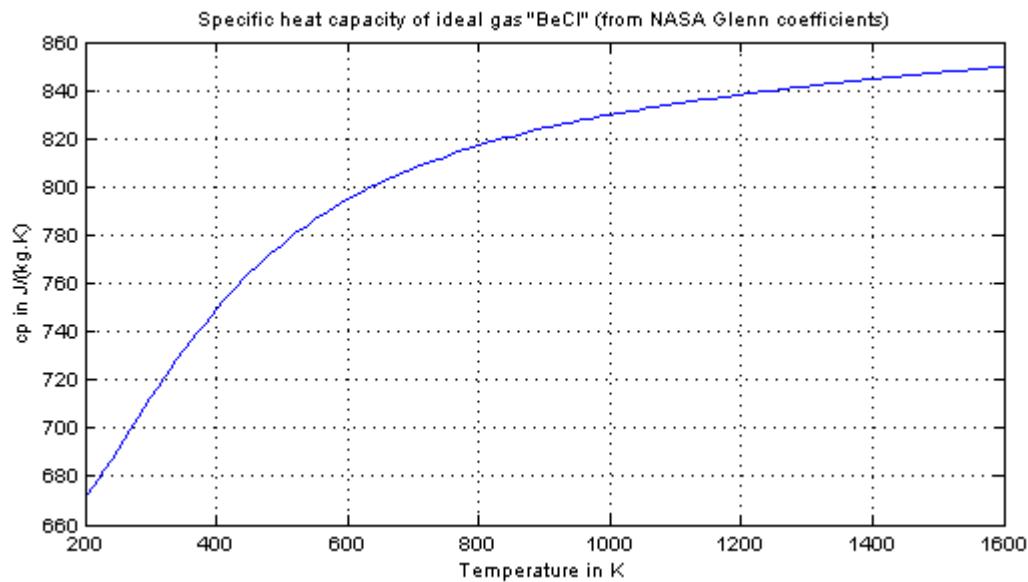
**Modelica.Media.IdealGases.SingleGases.BeCL**

Ideal gas "BeCl" from NASA Glenn coefficients

## 1318 Modelica.Media.IdealGases.SingleGases.BeCL

---

### Information

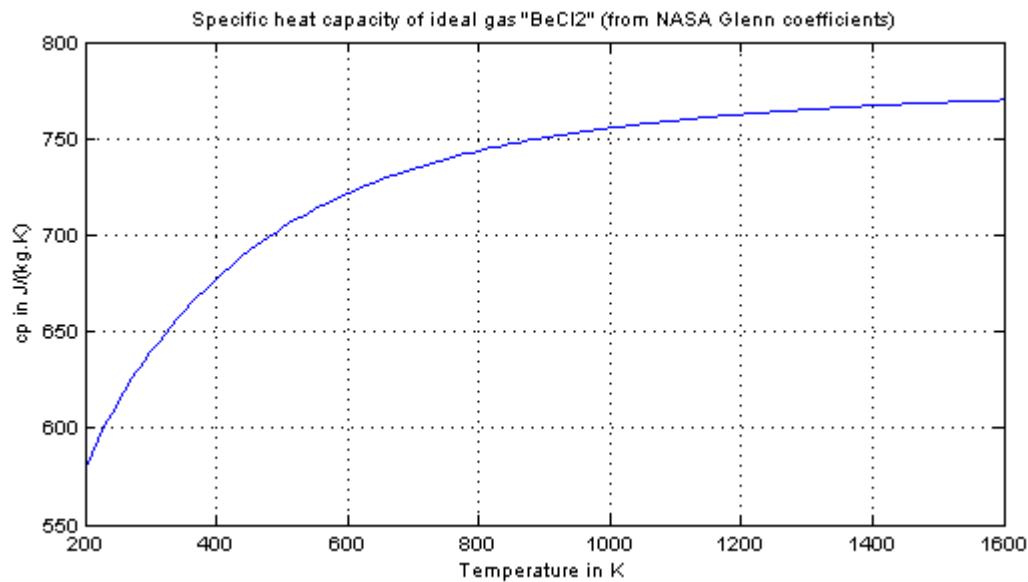


---

## Modelica.Media.IdealGases.SingleGases.BeCL2

Ideal gas "BeCl<sub>2</sub>" from NASA Glenn coefficients

### Information

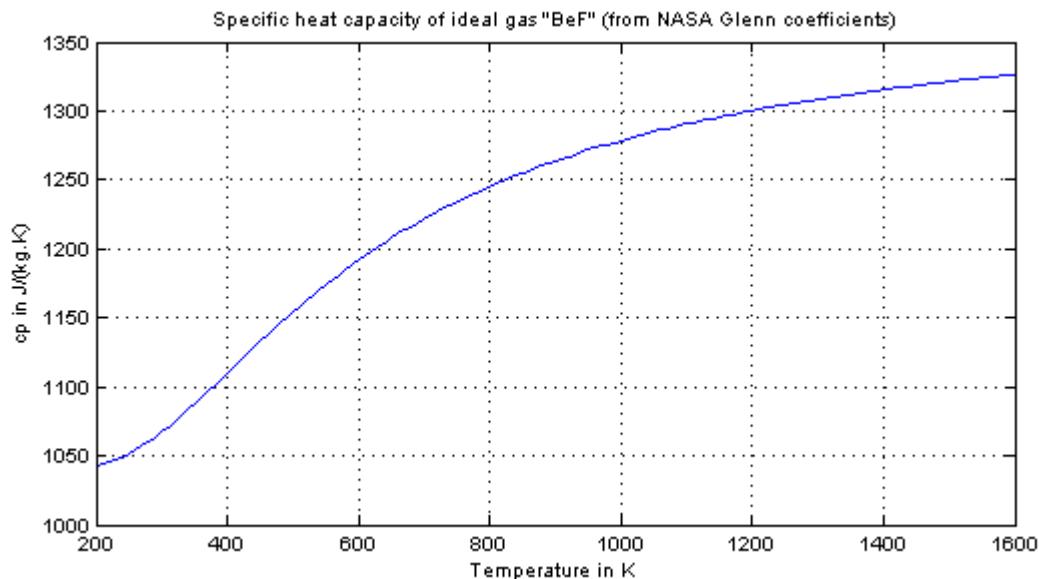


---

## Modelica.Media.IdealGases.SingleGases.BeF

Ideal gas "BeF" from NASA Glenn coefficients

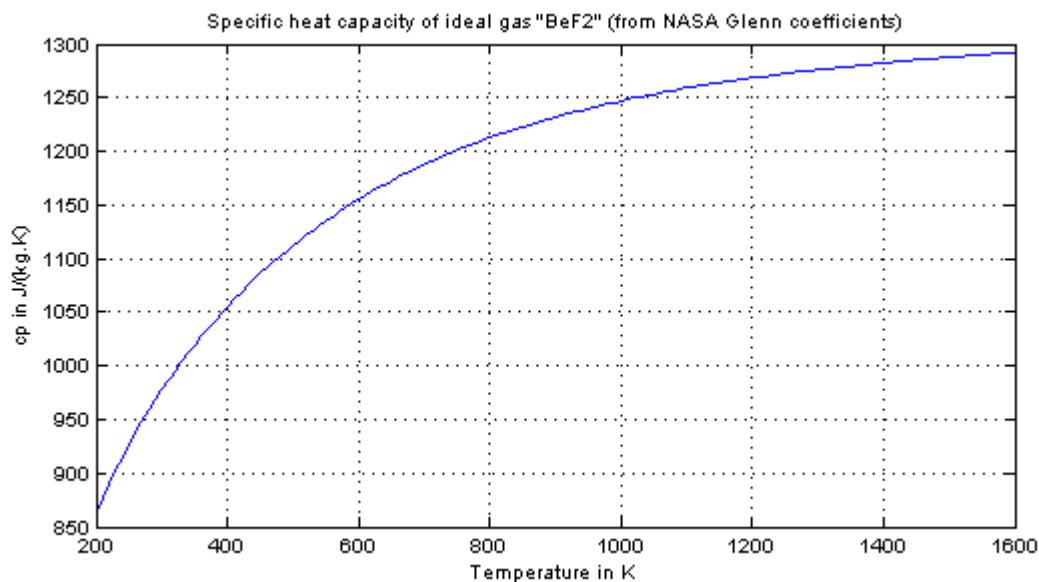
## Information



## Modelica.Media.IdealGases.SingleGases.BeF2

Ideal gas "BeF2" from NASA Glenn coefficients

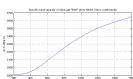
## Information



## Modelica.Media.IdealGases.SingleGases.BeH

Ideal gas "BeH" from NASA Glenn coefficients

**Information**

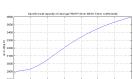


---

**Modelica.Media.IdealGases.SingleGases.BeHplus**

Ideal gas "BeH+" from NASA Glenn coefficients

**Information**

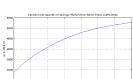


---

**Modelica.Media.IdealGases.SingleGases.BeH2**

Ideal gas "BeH2" from NASA Glenn coefficients

**Information**

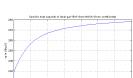


---

**Modelica.Media.IdealGases.SingleGases.Bel**

Ideal gas "Bel" from NASA Glenn coefficients

**Information**

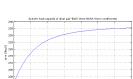


---

**Modelica.Media.IdealGases.SingleGases.Bel2**

Ideal gas "Bel2" from NASA Glenn coefficients

**Information**

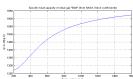


---

**Modelica.Media.IdealGases.SingleGases.BeN**

Ideal gas "BeN" from NASA Glenn coefficients

## Information



---

## Modelica.Media.IdealGases.SingleGases.BeO

Ideal gas "BeO" from NASA Glenn coefficients

## Information

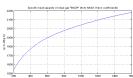


---

## Modelica.Media.IdealGases.SingleGases.BeOH

Ideal gas "BeOH" from NASA Glenn coefficients

## Information

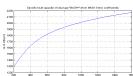


---

## Modelica.Media.IdealGases.SingleGases.BeOHplus

Ideal gas "BeOH+" from NASA Glenn coefficients

## Information

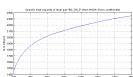


---

## Modelica.Media.IdealGases.SingleGases.Be\_OH\_2

Ideal gas "Be\_OH\_2" from NASA Glenn coefficients

## Information

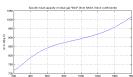


---

## Modelica.Media.IdealGases.SingleGases.BeS

Ideal gas "BeS" from NASA Glenn coefficients

**Information**

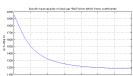


---

**Modelica.Media.IdealGases.SingleGases.Be2**

Ideal gas "Be2" from NASA Glenn coefficients

**Information**

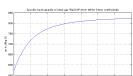


---

**Modelica.Media.IdealGases.SingleGases.Be2Cl4**

Ideal gas "Be2Cl4" from NASA Glenn coefficients

**Information**

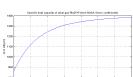


---

**Modelica.Media.IdealGases.SingleGases.Be2F4**

Ideal gas "Be2F4" from NASA Glenn coefficients

**Information**

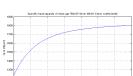


---

**Modelica.Media.IdealGases.SingleGases.Be2O**

Ideal gas "Be2O" from NASA Glenn coefficients

**Information**

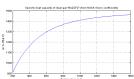


---

**Modelica.Media.IdealGases.SingleGases.Be2OF2**

Ideal gas "Be2OF2" from NASA Glenn coefficients

**Information**

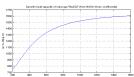


---

**Modelica.Media.IdealGases.SingleGases.Be2O2**

Ideal gas "Be2O2" from NASA Glenn coefficients

**Information**

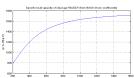


---

**Modelica.Media.IdealGases.SingleGases.Be3O3**

Ideal gas "Be3O3" from NASA Glenn coefficients

**Information**

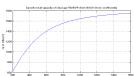


---

**Modelica.Media.IdealGases.SingleGases.Be4O4**

Ideal gas "Be4O4" from NASA Glenn coefficients

**Information**

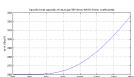


---

**Modelica.Media.IdealGases.SingleGases.Br**

Ideal gas "Br" from NASA Glenn coefficients

**Information**



---

**Modelica.Media.IdealGases.SingleGases.Brplus**

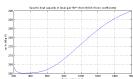
Ideal gas "Br+" from NASA Glenn coefficients

---

## 1324 Modelica.Media.IdealGases.SingleGases.Brplus

---

### Information

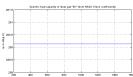


---

## Modelica.Media.IdealGases.SingleGases.Brminus

Ideal gas "Br-" from NASA Glenn coefficients

### Information

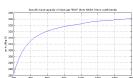


---

## Modelica.Media.IdealGases.SingleGases.BrCL

Ideal gas "BrCl" from NASA Glenn coefficients

### Information

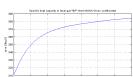


---

## Modelica.Media.IdealGases.SingleGases.BrF

Ideal gas "BrF" from NASA Glenn coefficients

### Information

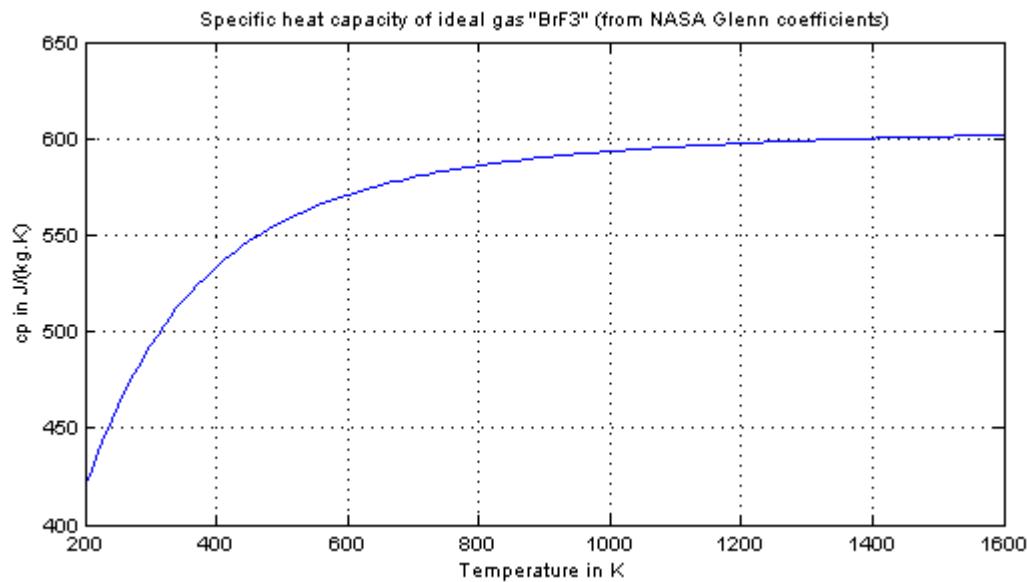


---

## Modelica.Media.IdealGases.SingleGases.BrF3

Ideal gas "BrF3" from NASA Glenn coefficients

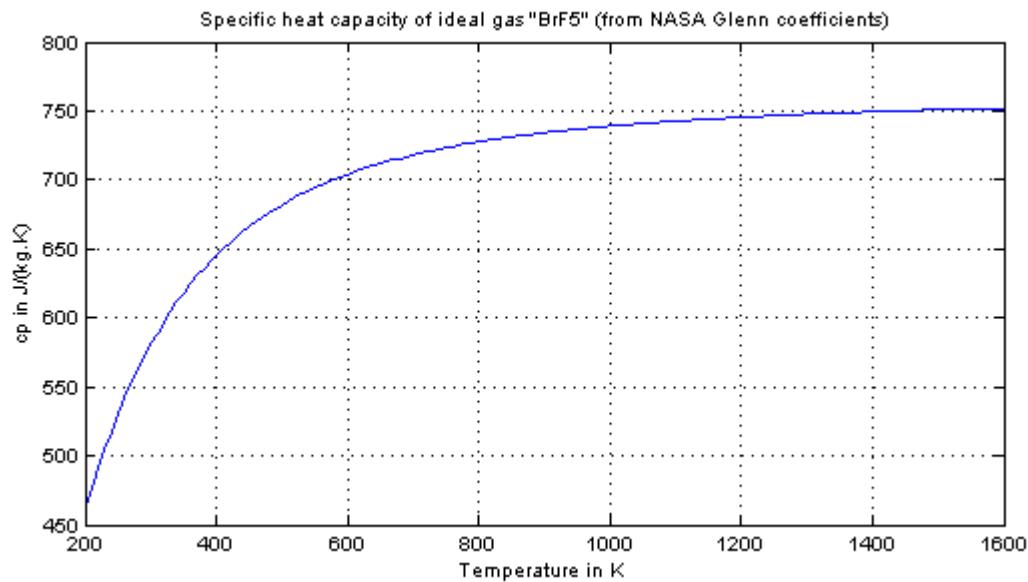
## Information



## Modelica.Media.IdealGases.SingleGases.BrF5

Ideal gas "BrF5" from NASA Glenn coefficients

## Information



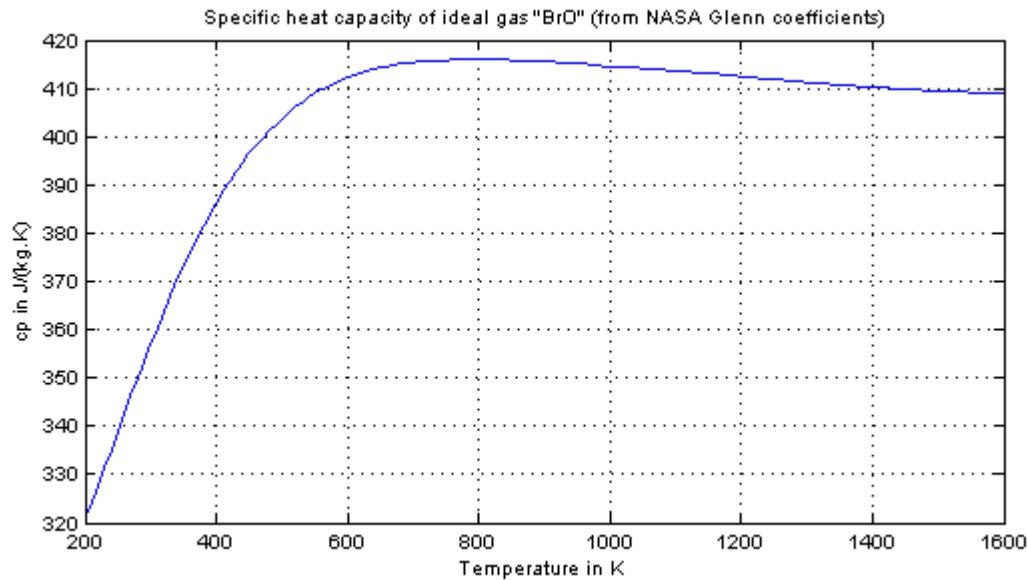
## Modelica.Media.IdealGases.SingleGases.BrO

Ideal gas "BrO" from NASA Glenn coefficients

## 1326 Modelica.Media.IdealGases.SingleGases.BrO

---

### Information

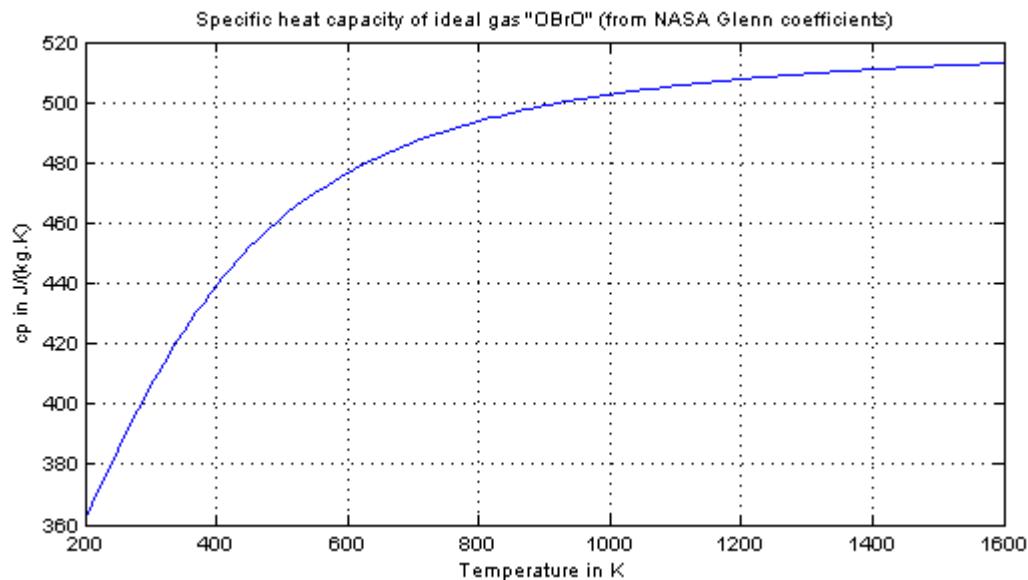


---

## Modelica.Media.IdealGases.SingleGases.Obro

Ideal gas "Obro" from NASA Glenn coefficients

### Information

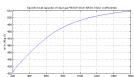


---

## Modelica.Media.IdealGases.SingleGases.BrOO

Ideal gas "BrOO" from NASA Glenn coefficients

## Information

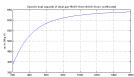


---

## Modelica.Media.IdealGases.SingleGases.BrO3

Ideal gas "BrO3" from NASA Glenn coefficients

## Information

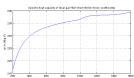


---

## Modelica.Media.IdealGases.SingleGases.Br2

Ideal gas "Br2" from NASA Glenn coefficients

## Information

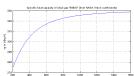


---

## Modelica.Media.IdealGases.SingleGases.BrBrO

Ideal gas "BrBrO" from NASA Glenn coefficients

## Information

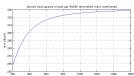


---

## Modelica.Media.IdealGases.SingleGases.BrOBr

Ideal gas "BrOBr" from NASA Glenn coefficients

## Information

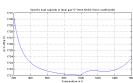


---

## Modelica.Media.IdealGases.SingleGases.C

Ideal gas "C" from NASA Glenn coefficients

**Information**

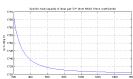


---

**Modelica.Media.IdealGases.SingleGases.Cplus**

Ideal gas "C+" from NASA Glenn coefficients

**Information**

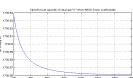


---

**Modelica.Media.IdealGases.SingleGases.Cminus**

Ideal gas "C-" from NASA Glenn coefficients

**Information**

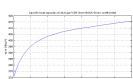


---

**Modelica.Media.IdealGases.SingleGases.CBr**

Ideal gas "CBr" from NASA Glenn coefficients

**Information**

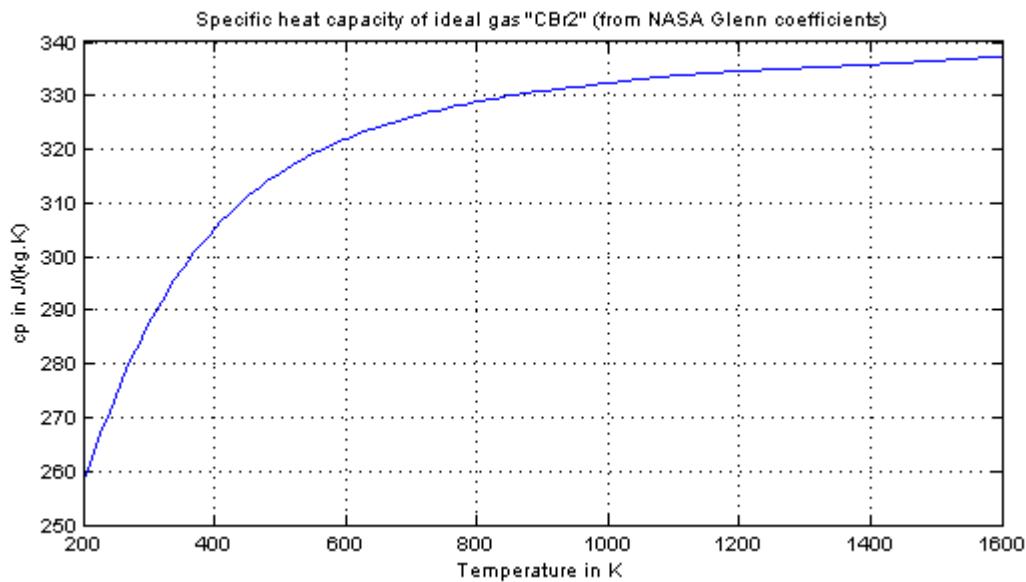


---

**Modelica.Media.IdealGases.SingleGases.CBr2**

Ideal gas "CBr2" from NASA Glenn coefficients

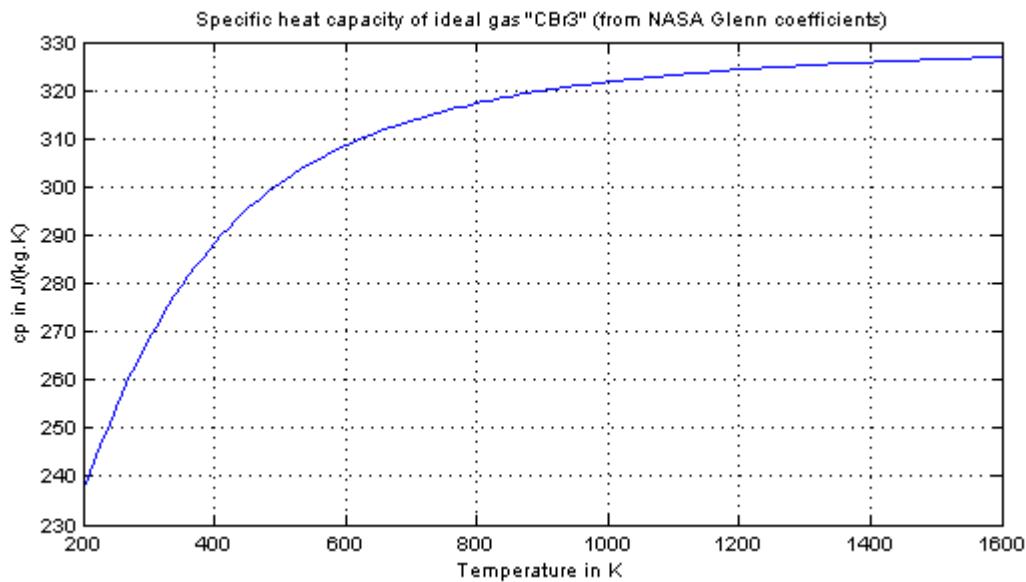
## Information



## Modelica.Media.IdealGases.SingleGases.CBr3

Ideal gas "CBr3" from NASA Glenn coefficients

## Information



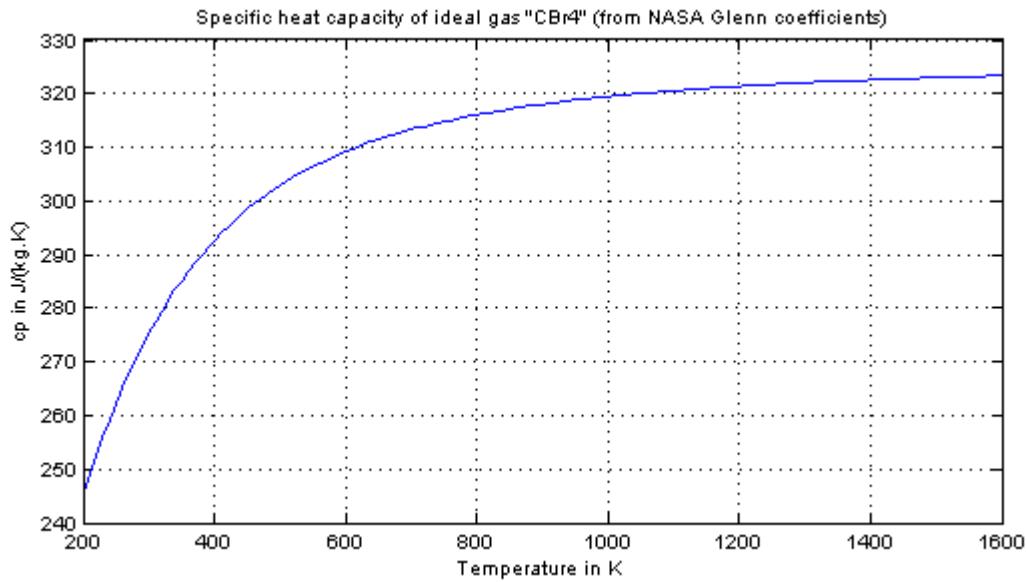
## Modelica.Media.IdealGases.SingleGases.CBr4

Ideal gas "CBr4" from NASA Glenn coefficients

## 1330 Modelica.Media.IdealGases.SingleGases.CBr4

---

### Information

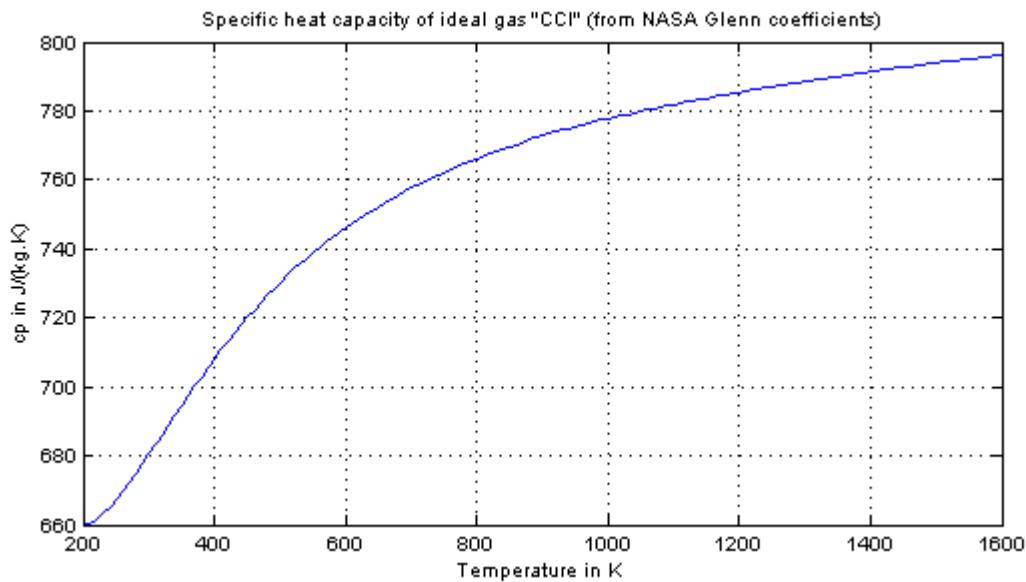


---

## Modelica.Media.IdealGases.SingleGases.CCL

Ideal gas "CCL" from NASA Glenn coefficients

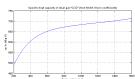
### Information



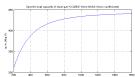
---

## Modelica.Media.IdealGases.SingleGases.CCL2

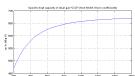
Ideal gas "CCL2" from NASA Glenn coefficients

**Information****Modelica.Media.IdealGases.SingleGases.CCL2Br2**

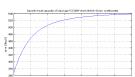
Ideal gas "CCl2Br2" from NASA Glenn coefficients

**Information****Modelica.Media.IdealGases.SingleGases.CCL3**

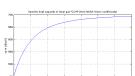
Ideal gas "CCl3" from NASA Glenn coefficients

**Information****Modelica.Media.IdealGases.SingleGases.CCL3Br**

Ideal gas "CCl3Br" from NASA Glenn coefficients

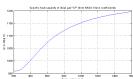
**Information****Modelica.Media.IdealGases.SingleGases.CCL4**

Ideal gas "CCl4" from NASA Glenn coefficients

**Information****Modelica.Media.IdealGases.SingleGases.CF**

Ideal gas "CF" from NASA Glenn coefficients

**Information**

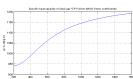


---

**Modelica.Media.IdealGases.SingleGases.CFplus**

Ideal gas "CF+" from NASA Glenn coefficients

**Information**

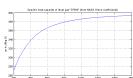


---

**Modelica.Media.IdealGases.SingleGases.CFBr3**

Ideal gas "CFBr3" from NASA Glenn coefficients

**Information**



---

**Modelica.Media.IdealGases.SingleGases.CFCI**

Ideal gas "CFCI" from NASA Glenn coefficients

**Information**

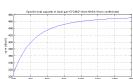


---

**Modelica.Media.IdealGases.SingleGases.CFCLBr2**

Ideal gas "CFCIBr2" from NASA Glenn coefficients

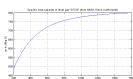
**Information**



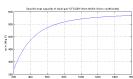
---

**Modelica.Media.IdealGases.SingleGases.CFCL2**

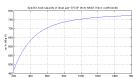
Ideal gas "CFCI2" from NASA Glenn coefficients

**Information****Modelica.Media.IdealGases.SingleGases.CFCL2Br**

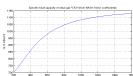
Ideal gas "CFCL2Br" from NASA Glenn coefficients

**Information****Modelica.Media.IdealGases.SingleGases.CFCL3**

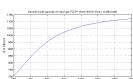
Ideal gas "CFCL3" from NASA Glenn coefficients

**Information****Modelica.Media.IdealGases.SingleGases.CF2**

Ideal gas "CF2" from NASA Glenn coefficients

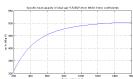
**Information****Modelica.Media.IdealGases.SingleGases.CF2plus**

Ideal gas "CF2+" from NASA Glenn coefficients

**Information****Modelica.Media.IdealGases.SingleGases.CF2Br2**

Ideal gas "CF2Br2" from NASA Glenn coefficients

**Information**

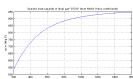


---

**Modelica.Media.IdealGases.SingleGases.CF2CL**

Ideal gas "CF2Cl" from NASA Glenn coefficients

**Information**

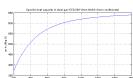


---

**Modelica.Media.IdealGases.SingleGases.CF2CLBr**

Ideal gas "CF2ClBr" from NASA Glenn coefficients

**Information**

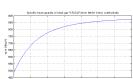


---

**Modelica.Media.IdealGases.SingleGases.CF2CL2**

Ideal gas "CF2Cl2" from NASA Glenn coefficients

**Information**

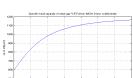


---

**Modelica.Media.IdealGases.SingleGases.CF3**

Ideal gas "CF3" from NASA Glenn coefficients

**Information**

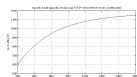


---

**Modelica.Media.IdealGases.SingleGases.CF3plus**

Ideal gas "CF3+" from NASA Glenn coefficients

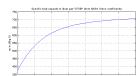
## Information



## Modelica.Media.IdealGases.SingleGases.CF3Br

Ideal gas "CF3Br" from NASA Glenn coefficients

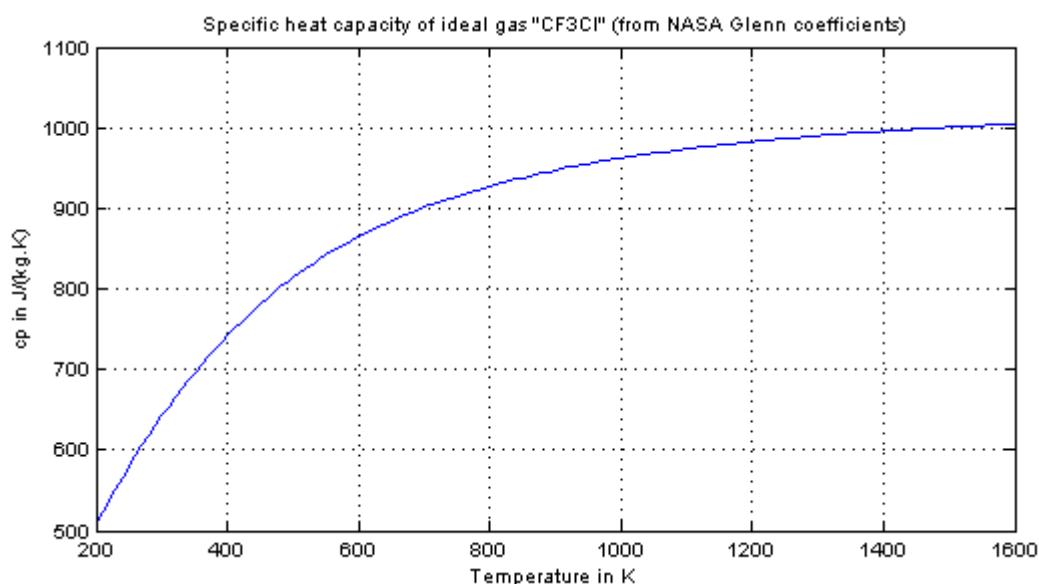
## Information



## Modelica.Media.IdealGases.SingleGases.CF3CL

Ideal gas "CF3Cl" from NASA Glenn coefficients

## Information



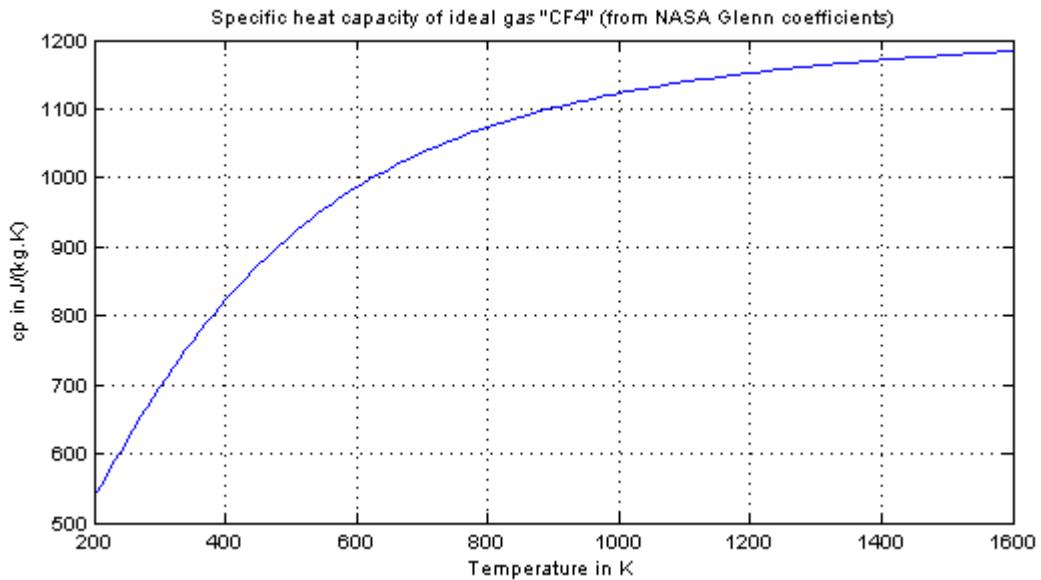
## Modelica.Media.IdealGases.SingleGases.CF4

Ideal gas "CF4" from NASA Glenn coefficients

## 1336 Modelica.Media.IdealGases.SingleGases.CF4

---

### Information

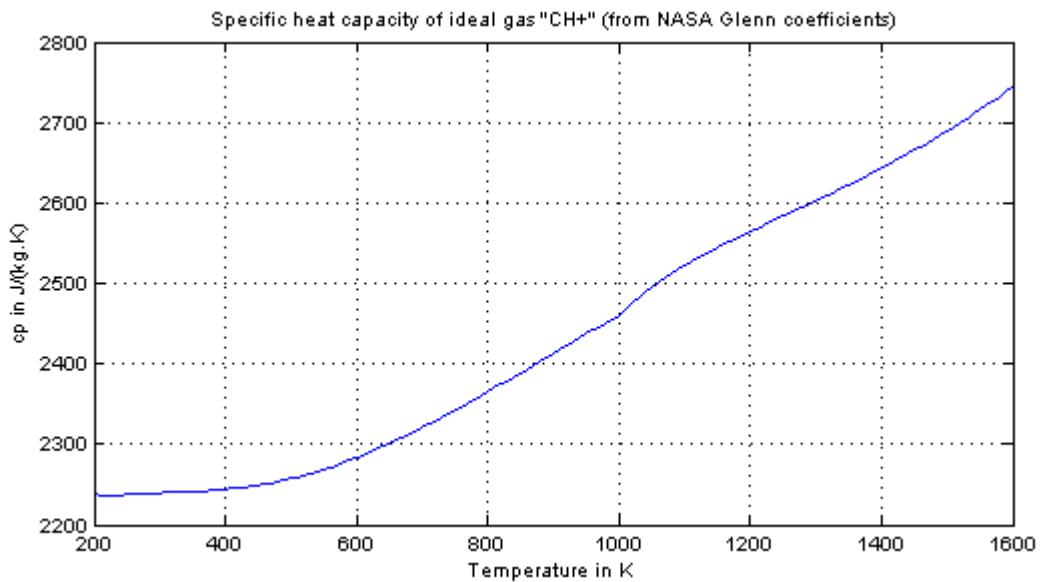


---

## Modelica.Media.IdealGases.SingleGases.CHplus

Ideal gas "CH+" from NASA Glenn coefficients

### Information

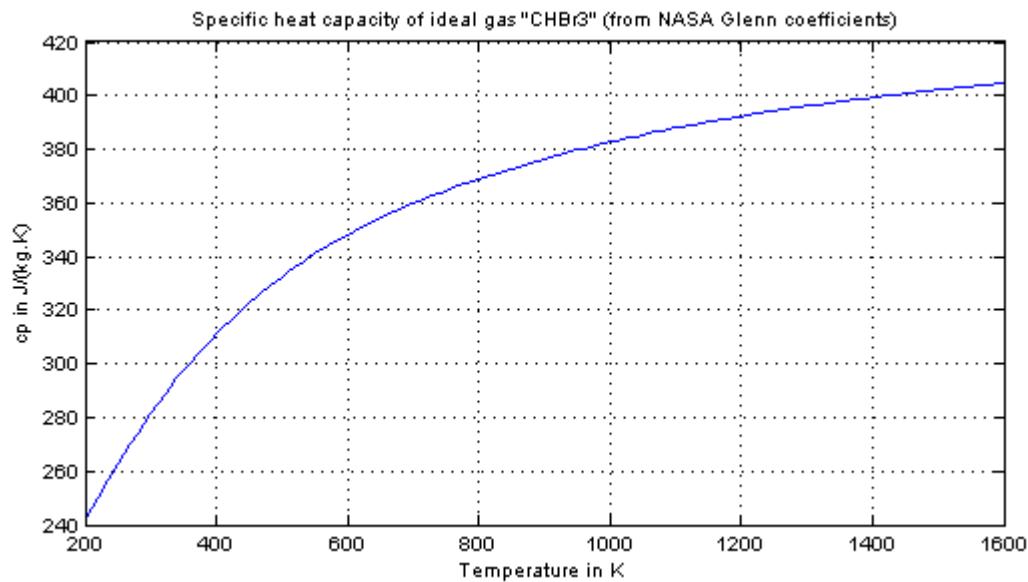


---

## Modelica.Media.IdealGases.SingleGases.CHBr3

Ideal gas "CHBr3" from NASA Glenn coefficients

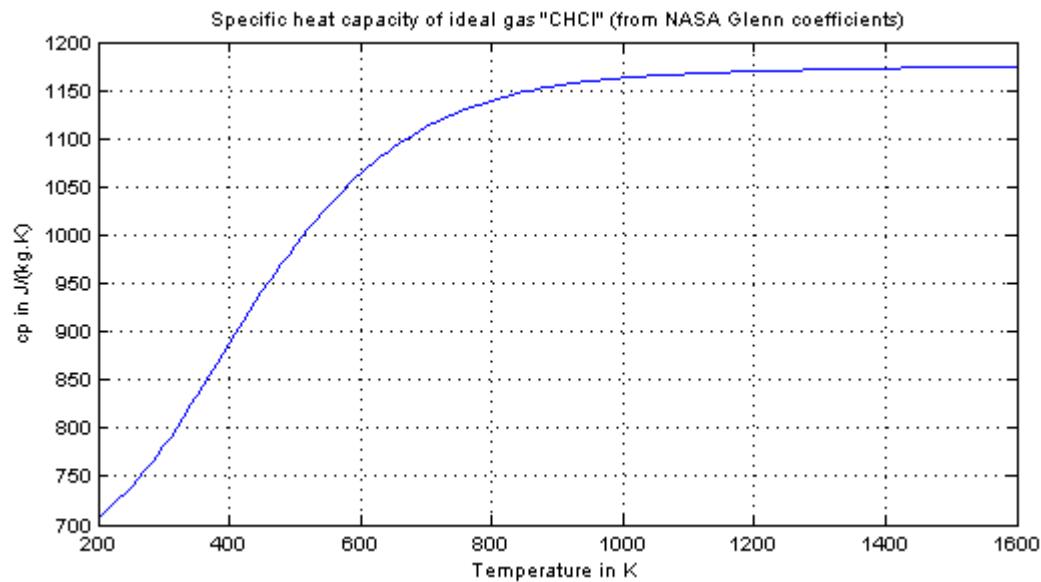
## Information



## Modelica.Media.IdealGases.SingleGases.CHCL

Ideal gas "CHCl" from NASA Glenn coefficients

## Information



## Modelica.Media.IdealGases.SingleGases.CHCLBr2

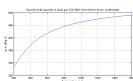
Ideal gas "CHClBr2" from NASA Glenn coefficients

---

## 1338 Modelica.Media.IdealGases.SingleGases.CHCLBr2

---

### Information

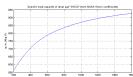


---

## Modelica.Media.IdealGases.SingleGases.CHCL2

Ideal gas "CHCl2" from NASA Glenn coefficients

### Information

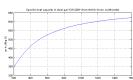


---

## Modelica.Media.IdealGases.SingleGases.CHCL2Br

Ideal gas "CHCl2Br" from NASA Glenn coefficients

### Information

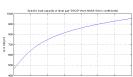


---

## Modelica.Media.IdealGases.SingleGases.CHCL3

Ideal gas "CHCl3" from NASA Glenn coefficients

### Information

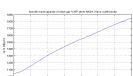


---

## Modelica.Media.IdealGases.SingleGases.CHF

Ideal gas "CHF" from NASA Glenn coefficients

### Information

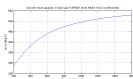


---

## Modelica.Media.IdealGases.SingleGases.CHFBr2

Ideal gas "CHFBr2" from NASA Glenn coefficients

**Information**

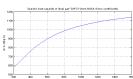


---

**Modelica.Media.IdealGases.SingleGases.CHFCI**

Ideal gas "CHFCI" from NASA Glenn coefficients

**Information**

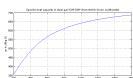


---

**Modelica.Media.IdealGases.SingleGases.CHFCIBr**

Ideal gas "CHFCIBr" from NASA Glenn coefficients

**Information**

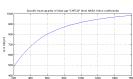


---

**Modelica.Media.IdealGases.SingleGases.CHFCI2**

Ideal gas "CHFCI2" from NASA Glenn coefficients

**Information**

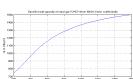


---

**Modelica.Media.IdealGases.SingleGases.CHF2**

Ideal gas "CHF2" from NASA Glenn coefficients

**Information**

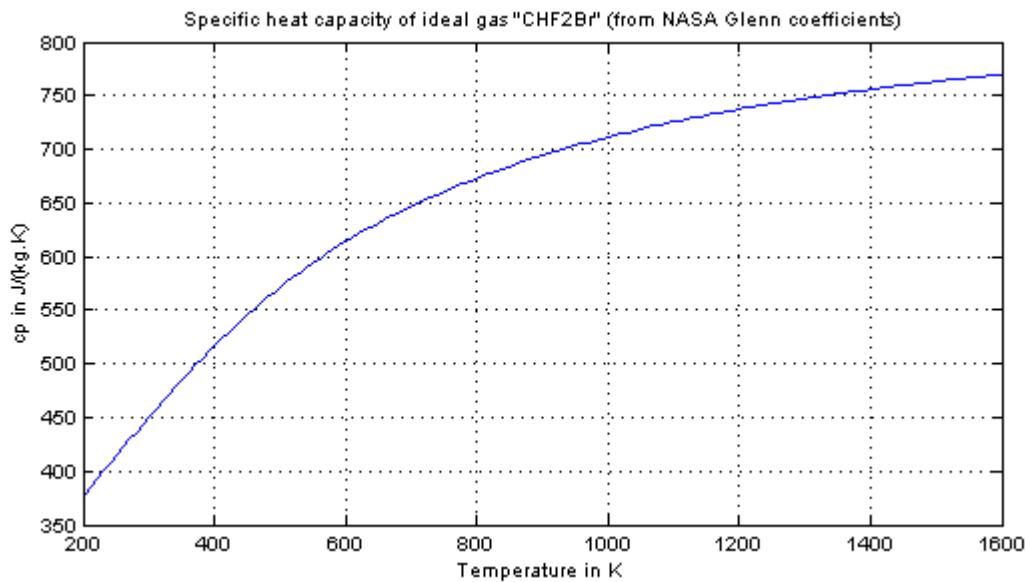


---

**Modelica.Media.IdealGases.SingleGases.CHF2Br**

Ideal gas "CHF2Br" from NASA Glenn coefficients

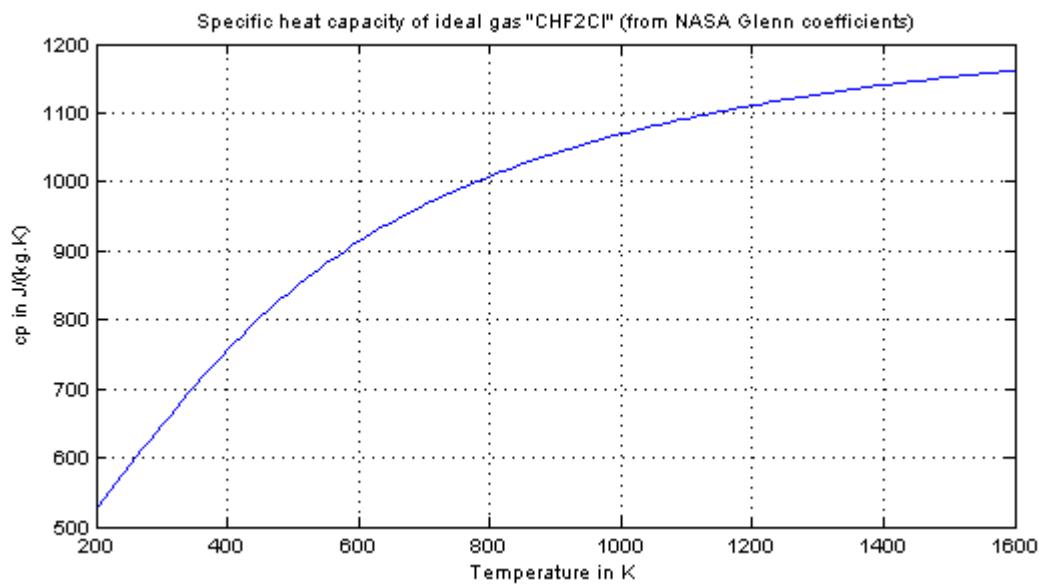
## Information



## Modelica.Media.IdealGases.SingleGases.CHF2CL

Ideal gas "CHF2Cl" from NASA Glenn coefficients

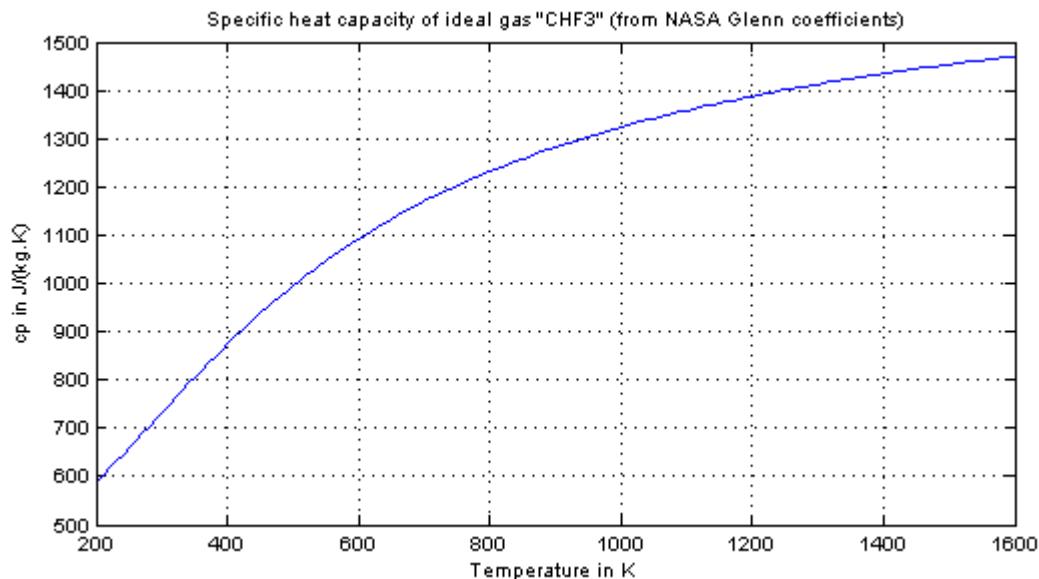
## Information



## Modelica.Media.IdealGases.SingleGases.CHF3

Ideal gas "CHF3" from NASA Glenn coefficients

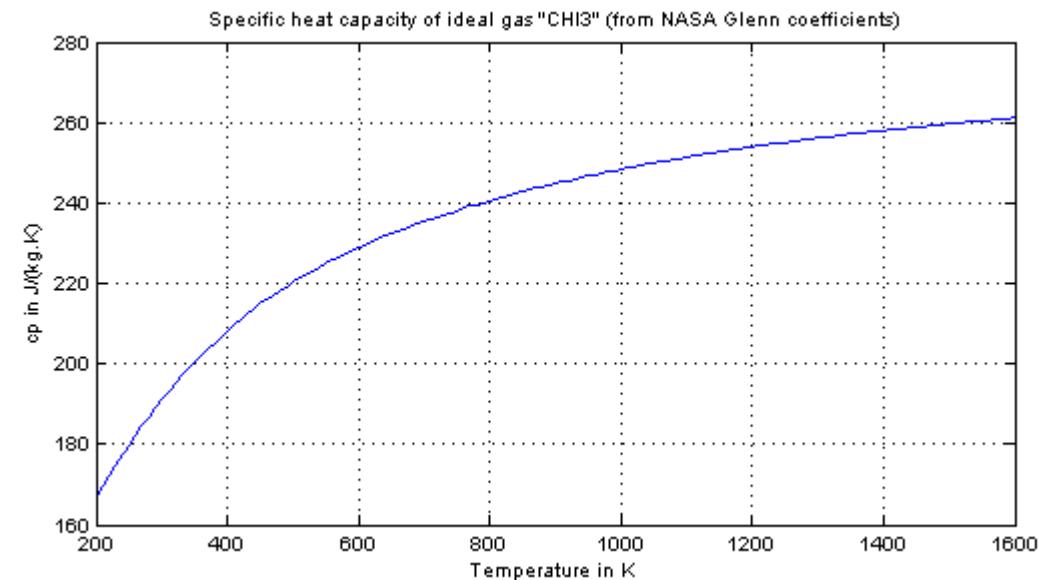
## Information



## Modelica.Media.IdealGases.SingleGases.CHI3

Ideal gas "CHI3" from NASA Glenn coefficients

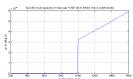
## Information



## Modelica.Media.IdealGases.SingleGases.CH2

Ideal gas "CH2" from NASA Glenn coefficients

**Information**

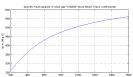


---

**Modelica.Media.IdealGases.SingleGases.CH2Br2**

Ideal gas "CH2Br2" from NASA Glenn coefficients

**Information**



---

**Modelica.Media.IdealGases.SingleGases.CH2CL**

Ideal gas "CH2Cl" from NASA Glenn coefficients

**Information**

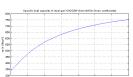


---

**Modelica.Media.IdealGases.SingleGases.CH2CLBr**

Ideal gas "CH2ClBr" from NASA Glenn coefficients

**Information**

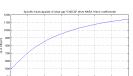


---

**Modelica.Media.IdealGases.SingleGases.CH2CL2**

Ideal gas "CH2Cl2" from NASA Glenn coefficients

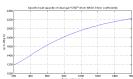
**Information**



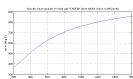
---

**Modelica.Media.IdealGases.SingleGases.CH2F**

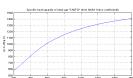
Ideal gas "CH2F" from NASA Glenn coefficients

**Information****Modelica.Media.IdealGases.SingleGases.CH2FBr**

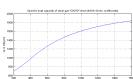
Ideal gas "CH2FBr" from NASA Glenn coefficients

**Information****Modelica.Media.IdealGases.SingleGases.CH2FCl**

Ideal gas "CH2FCI" from NASA Glenn coefficients

**Information****Modelica.Media.IdealGases.SingleGases.CH2F2**

Ideal gas "CH2F2" from NASA Glenn coefficients

**Information****Modelica.Media.IdealGases.SingleGases.CH2I2**

Ideal gas "CH2I2" from NASA Glenn coefficients

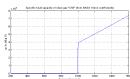
**Information****Modelica.Media.IdealGases.SingleGases.CH3**

Ideal gas "CH3" from NASA Glenn coefficients

## 1344 Modelica.Media.IdealGases.SingleGases.CH3

---

### Information

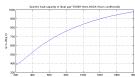


---

## Modelica.Media.IdealGases.SingleGases.CH3Br

Ideal gas "CH3Br" from NASA Glenn coefficients

### Information

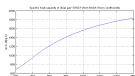


---

## Modelica.Media.IdealGases.SingleGases.CH3Cl

Ideal gas "CH3Cl" from NASA Glenn coefficients

### Information

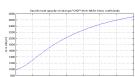


---

## Modelica.Media.IdealGases.SingleGases.CH3F

Ideal gas "CH3F" from NASA Glenn coefficients

### Information

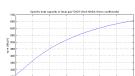


---

## Modelica.Media.IdealGases.SingleGases.CH3I

Ideal gas "CH3I" from NASA Glenn coefficients

### Information

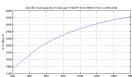


---

## Modelica.Media.IdealGases.SingleGases.CH2OH

Ideal gas "CH2OH" from NASA Glenn coefficients

## Information

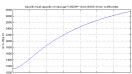


---

## Modelica.Media.IdealGases.SingleGases.CH2OHplus

Ideal gas "CH2OH+" from NASA Glenn coefficients

## Information

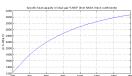


---

## Modelica.Media.IdealGases.SingleGases.CH3O

Ideal gas "CH3O" from NASA Glenn coefficients

## Information

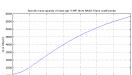


---

## Modelica.Media.IdealGases.SingleGases.CH4

Ideal gas "CH4" from NASA Glenn coefficients

## Information

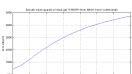


---

## Modelica.Media.IdealGases.SingleGases.CH3OH

Ideal gas "CH3OH" from NASA Glenn coefficients

## Information

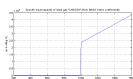


---

## Modelica.Media.IdealGases.SingleGases.CH3OOH

Ideal gas "CH3OOH" from NASA Glenn coefficients

## Information

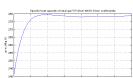


---

## Modelica.Media.IdealGases.SingleGases.CI

Ideal gas "CI" from NASA Glenn coefficients

## Information

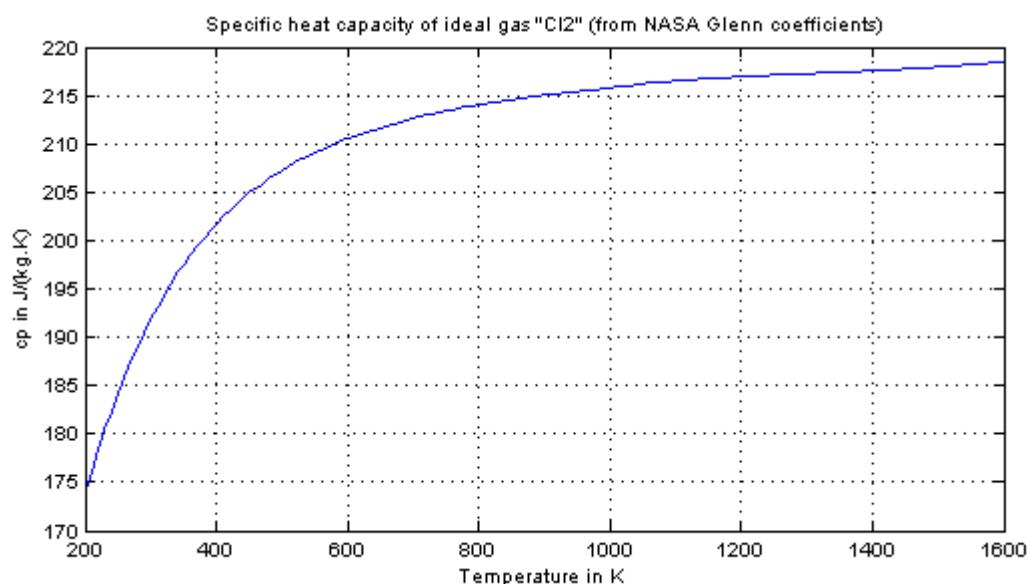


---

## Modelica.Media.IdealGases.SingleGases.CI2

Ideal gas "CI2" from NASA Glenn coefficients

## Information

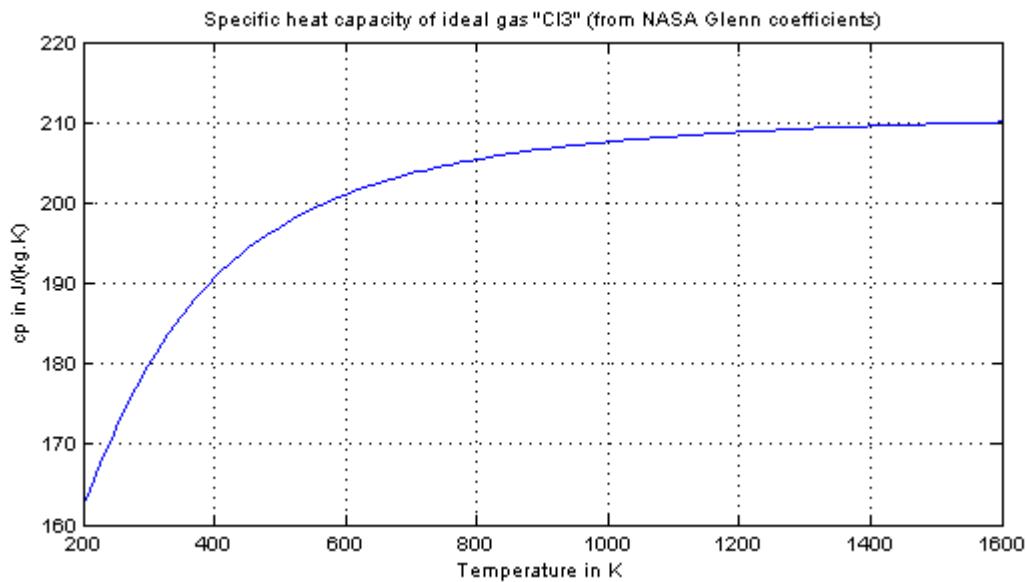


---

## Modelica.Media.IdealGases.SingleGases.CI3

Ideal gas "CI3" from NASA Glenn coefficients

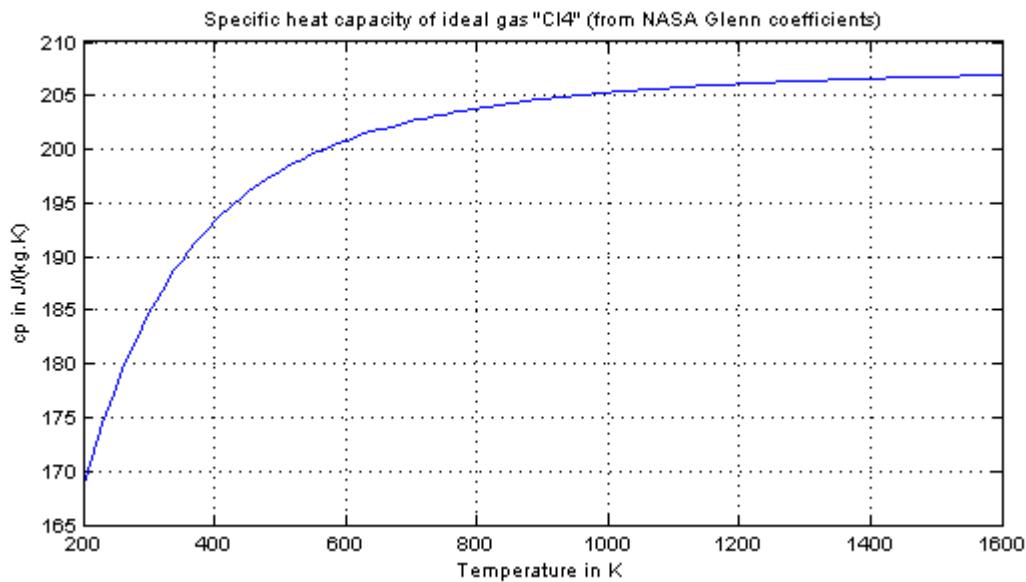
## Information



## Modelica.Media.IdealGases.SingleGases.CI4

Ideal gas "CI4" from NASA Glenn coefficients

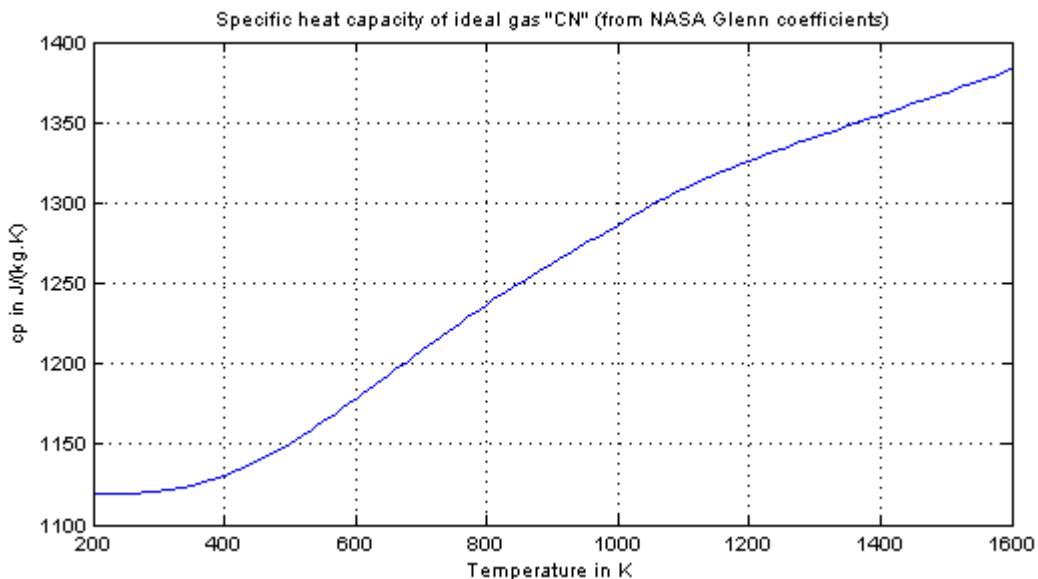
## Information



## Modelica.Media.IdealGases.SingleGases.CN

Ideal gas "CN" from NASA Glenn coefficients

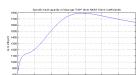
## Information



## Modelica.Media.IdealGases.SingleGases.CNplus

Ideal gas "CN+" from NASA Glenn coefficients

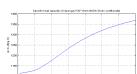
## Information



## Modelica.Media.IdealGases.SingleGases.CNminus

Ideal gas "CN-" from NASA Glenn coefficients

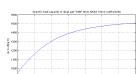
## Information



## Modelica.Media.IdealGases.SingleGases.CNN

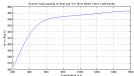
Ideal gas "CNN" from NASA Glenn coefficients

## Information

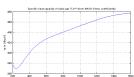


**Modelica.Media.IdealGases.SingleGases.CO**

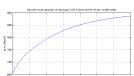
Ideal gas "CO" from NASA Glenn coefficients

**Information****Modelica.Media.IdealGases.SingleGases.COplus**

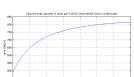
Ideal gas "CO+" from NASA Glenn coefficients

**Information****Modelica.Media.IdealGases.SingleGases.COCL**

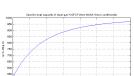
Ideal gas "COCl" from NASA Glenn coefficients

**Information****Modelica.Media.IdealGases.SingleGases.COCL2**

Ideal gas "COCl2" from NASA Glenn coefficients

**Information****Modelica.Media.IdealGases.SingleGases.COFCL**

Ideal gas "COFCl" from NASA Glenn coefficients

**Information**

---

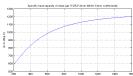
## 1350 Modelica.Media.IdealGases.SingleGases.COF2

---

### Modelica.Media.IdealGases.SingleGases.COF2

Ideal gas "COF2" from NASA Glenn coefficients

#### Information

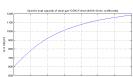


---

### Modelica.Media.IdealGases.SingleGases.COHCl

Ideal gas "COHCl" from NASA Glenn coefficients

#### Information

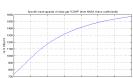


---

### Modelica.Media.IdealGases.SingleGases.COHF

Ideal gas "COHF" from NASA Glenn coefficients

#### Information

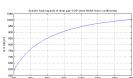


---

### Modelica.Media.IdealGases.SingleGases.COS

Ideal gas "COS" from NASA Glenn coefficients

#### Information

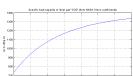


---

### Modelica.Media.IdealGases.SingleGases.CO2

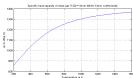
Ideal gas "CO2" from NASA Glenn coefficients

#### Information

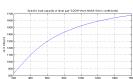


**Modelica.Media.IdealGases.SingleGases.CO2plus**

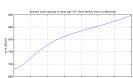
Ideal gas "CO2+" from NASA Glenn coefficients

**Information****Modelica.Media.IdealGases.SingleGases.COOH**

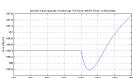
Ideal gas "COOH" from NASA Glenn coefficients

**Information****Modelica.Media.IdealGases.SingleGases.CP**

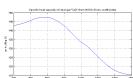
Ideal gas "CP" from NASA Glenn coefficients

**Information****Modelica.Media.IdealGases.SingleGases.CS**

Ideal gas "CS" from NASA Glenn coefficients

**Information****Modelica.Media.IdealGases.SingleGases.CS2**

Ideal gas "CS2" from NASA Glenn coefficients

**Information**

**Modelica.Media.IdealGases.SingleGases.C2**

Ideal gas "C2" from NASA Glenn coefficients

**Information**

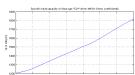


---

**Modelica.Media.IdealGases.SingleGases.C2plus**

Ideal gas "C2+" from NASA Glenn coefficients

**Information**

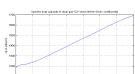


---

**Modelica.Media.IdealGases.SingleGases.C2minus**

Ideal gas "C2-" from NASA Glenn coefficients

**Information**

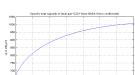


---

**Modelica.Media.IdealGases.SingleGases.C2CL**

Ideal gas "C2Cl" from NASA Glenn coefficients

**Information**



---

**Modelica.Media.IdealGases.SingleGases.C2CL2**

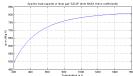
Ideal gas "C2Cl2" from NASA Glenn coefficients

**Information**

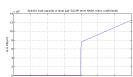


**Modelica.Media.IdealGases.SingleGases.C2CL3**

Ideal gas "C2Cl3" from NASA Glenn coefficients

**Information****Modelica.Media.IdealGases.SingleGases.C2CL4**

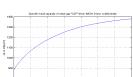
Ideal gas "C2Cl4" from NASA Glenn coefficients

**Information****Modelica.Media.IdealGases.SingleGases.C2CL6**

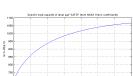
Ideal gas "C2Cl6" from NASA Glenn coefficients

**Information****Modelica.Media.IdealGases.SingleGases.C2F**

Ideal gas "C2F" from NASA Glenn coefficients

**Information****Modelica.Media.IdealGases.SingleGases.C2FCL**

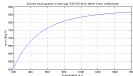
Ideal gas "C2FCI" from NASA Glenn coefficients

**Information**

**Modelica.Media.IdealGases.SingleGases.C2FCL3**

Ideal gas "C2FCI3" from NASA Glenn coefficients

**Information**

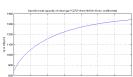


---

**Modelica.Media.IdealGases.SingleGases.C2F2**

Ideal gas "C2F2" from NASA Glenn coefficients

**Information**

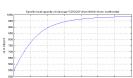


---

**Modelica.Media.IdealGases.SingleGases.C2F2CL2**

Ideal gas "C2F2CI2" from NASA Glenn coefficients

**Information**

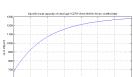


---

**Modelica.Media.IdealGases.SingleGases.C2F3**

Ideal gas "C2F3" from NASA Glenn coefficients

**Information**

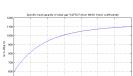


---

**Modelica.Media.IdealGases.SingleGases.C2F3CL**

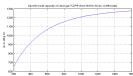
Ideal gas "C2F3CI" from NASA Glenn coefficients

**Information**

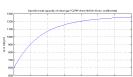


**Modelica.Media.IdealGases.SingleGases.C2F4**

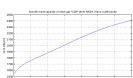
Ideal gas "C2F4" from NASA Glenn coefficients

**Information****Modelica.Media.IdealGases.SingleGases.C2F6**

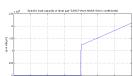
Ideal gas "C2F6" from NASA Glenn coefficients

**Information****Modelica.Media.IdealGases.SingleGases.C2H**

Ideal gas "C2H" from NASA Glenn coefficients

**Information****Modelica.Media.IdealGases.SingleGases.C2HCl**

Ideal gas "C2HCl" from NASA Glenn coefficients

**Information****Modelica.Media.IdealGases.SingleGases.C2HCl3**

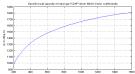
Ideal gas "C2HCl3" from NASA Glenn coefficients

**Information**

## Modelica.Media.IdealGases.SingleGases.C2HF

Ideal gas "C2HF" from NASA Glenn coefficients

### Information

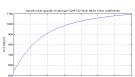


---

## Modelica.Media.IdealGases.SingleGases.C2HFCL2

Ideal gas "C2HFCl2" from NASA Glenn coefficients

### Information

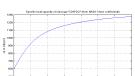


---

## Modelica.Media.IdealGases.SingleGases.C2HF2CL

Ideal gas "C2HF2Cl" from NASA Glenn coefficients

### Information

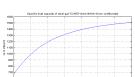


---

## Modelica.Media.IdealGases.SingleGases.C2HF3

Ideal gas "C2HF3" from NASA Glenn coefficients

### Information

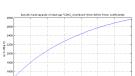


---

## Modelica.Media.IdealGases.SingleGases.C2H2\_vinylidene

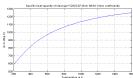
Ideal gas "C2H2\_vinylidene" from NASA Glenn coefficients

### Information

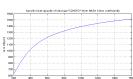


**Modelica.Media.IdealGases.SingleGases.C2H2CL2**

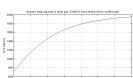
Ideal gas "C2H2Cl2" from NASA Glenn coefficients

**Information****Modelica.Media.IdealGases.SingleGases.C2H2FCL**

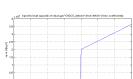
Ideal gas "C2H2FCI" from NASA Glenn coefficients

**Information****Modelica.Media.IdealGases.SingleGases.C2H2F2**

Ideal gas "C2H2F2" from NASA Glenn coefficients

**Information****Modelica.Media.IdealGases.SingleGases.CH2CO\_ketene**

Ideal gas "CH2CO\_ketene" from NASA Glenn coefficients

**Information****Modelica.Media.IdealGases.SingleGases.O\_CH\_2O**

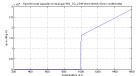
Ideal gas "O\_CH\_2O" from NASA Glenn coefficients

**Information**

**Modelica.Media.IdealGases.SingleGases.HO\_CO\_2OH**

Ideal gas "HO\_CO\_2OH" from NASA Glenn coefficients

**Information**

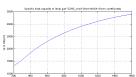


---

**Modelica.Media.IdealGases.SingleGases.C2H3\_vinyl**

Ideal gas "C2H3\_vinyl" from NASA Glenn coefficients

**Information**

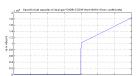


---

**Modelica.Media.IdealGases.SingleGases.CH2BrminusCOOH**

Ideal gas "CH2Br-COOH" from NASA Glenn coefficients

**Information**

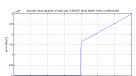


---

**Modelica.Media.IdealGases.SingleGases.C2H3CL**

Ideal gas "C2H3Cl" from NASA Glenn coefficients

**Information**

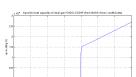


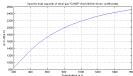
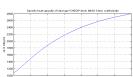
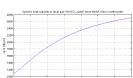
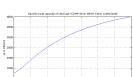
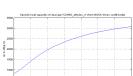
---

**Modelica.Media.IdealGases.SingleGases.CH2CLminusCOOH**

Ideal gas "CH2Cl-COOH" from NASA Glenn coefficients

**Information**

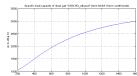


**Modelica.Media.IdealGases.SingleGases.C2H3F****Ideal gas "C2H3F" from NASA Glenn coefficients****Information****Modelica.Media.IdealGases.SingleGases.CH3CN****Ideal gas "CH3CN" from NASA Glenn coefficients****Information****Modelica.Media.IdealGases.SingleGases.CH3CO\_acetyl****Ideal gas "CH3CO\_acetyl" from NASA Glenn coefficients****Information****Modelica.Media.IdealGases.SingleGases.C2H4****Ideal gas "C2H4" from NASA Glenn coefficients****Information****Modelica.Media.IdealGases.SingleGases.C2H4O\_ethyl\_o****Ideal gas "C2H4O\_ethyl\_o" from NASA Glenn coefficients****Information**

**Modelica.Media.IdealGases.SingleGases.CH3CHO\_ethanal**

Ideal gas "CH3CHO\_ethanal" from NASA Glenn coefficients

**Information**

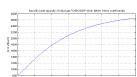


---

**Modelica.Media.IdealGases.SingleGases.CH3COOH**

Ideal gas "CH3COOH" from NASA Glenn coefficients

**Information**

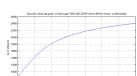


---

**Modelica.Media.IdealGases.SingleGases.OHCH2COOH**

Ideal gas "OHCH2COOH" from NASA Glenn coefficients

**Information**

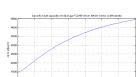


---

**Modelica.Media.IdealGases.SingleGases.C2H5**

Ideal gas "C2H5" from NASA Glenn coefficients

**Information**

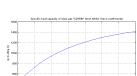


---

**Modelica.Media.IdealGases.SingleGases.C2H5Br**

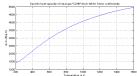
Ideal gas "C2H5Br" from NASA Glenn coefficients

**Information**

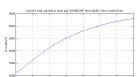


**Modelica.Media.IdealGases.SingleGases.C2H6**

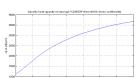
Ideal gas "C2H6" from NASA Glenn coefficients

**Information****Modelica.Media.IdealGases.SingleGases.CH3N2CH3**

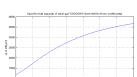
Ideal gas "CH3N2CH3" from NASA Glenn coefficients

**Information****Modelica.Media.IdealGases.SingleGases.C2H5OH**

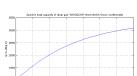
Ideal gas "C2H5OH" from NASA Glenn coefficients

**Information****Modelica.Media.IdealGases.SingleGases.CH3OCH3**

Ideal gas "CH3OCH3" from NASA Glenn coefficients

**Information****Modelica.Media.IdealGases.SingleGases.CH3O2CH3**

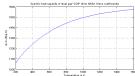
Ideal gas "CH3O2CH3" from NASA Glenn coefficients

**Information**

**Modelica.Media.IdealGases.SingleGases.CCN**

Ideal gas "CCN" from NASA Glenn coefficients

**Information**

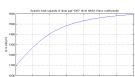


---

**Modelica.Media.IdealGases.SingleGases.CNC**

Ideal gas "CNC" from NASA Glenn coefficients

**Information**

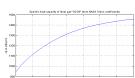


---

**Modelica.Media.IdealGases.SingleGases.OCCN**

Ideal gas "OCCN" from NASA Glenn coefficients

**Information**

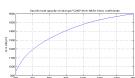


---

**Modelica.Media.IdealGases.SingleGases.C2N2**

Ideal gas "C2N2" from NASA Glenn coefficients

**Information**

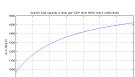


---

**Modelica.Media.IdealGases.SingleGases.C2O**

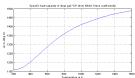
Ideal gas "C2O" from NASA Glenn coefficients

**Information**

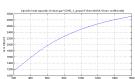


**Modelica.Media.IdealGases.SingleGases.C3**

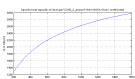
Ideal gas "C3" from NASA Glenn coefficients

**Information****Modelica.Media.IdealGases.SingleGases.C3H3\_1\_propynl**

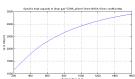
Ideal gas "C3H3\_1\_propynl" from NASA Glenn coefficients

**Information****Modelica.Media.IdealGases.SingleGases.C3H3\_2\_propynl**

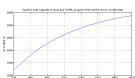
Ideal gas "C3H3\_2\_propynl" from NASA Glenn coefficients

**Information****Modelica.Media.IdealGases.SingleGases.C3H4\_allene**

Ideal gas "C3H4\_allene" from NASA Glenn coefficients

**Information****Modelica.Media.IdealGases.SingleGases.C3H4\_propyne**

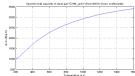
Ideal gas "C3H4\_propyne" from NASA Glenn coefficients

**Information**

**Modelica.Media.IdealGases.SingleGases.C3H4\_cyclo**

Ideal gas "C3H4\_cyclo" from NASA Glenn coefficients

**Information**

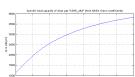


---

**Modelica.Media.IdealGases.SingleGases.C3H5\_allyl**

Ideal gas "C3H5\_allyl" from NASA Glenn coefficients

**Information**

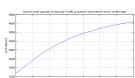


---

**Modelica.Media.IdealGases.SingleGases.C3H6\_propylene**

Ideal gas "C3H6\_propylene" from NASA Glenn coefficients

**Information**

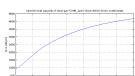


---

**Modelica.Media.IdealGases.SingleGases.C3H6\_cyclo**

Ideal gas "C3H6\_cyclo" from NASA Glenn coefficients

**Information**

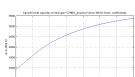


---

**Modelica.Media.IdealGases.SingleGases.C3H6O\_propylox**

Ideal gas "C3H6O\_propylox" from NASA Glenn coefficients

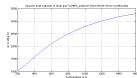
**Information**



### Modelica.Media.IdealGases.SingleGases.C3H6O\_acetone

Ideal gas "C3H6O\_acetone" from NASA Glenn coefficients

#### Information

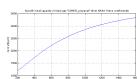


---

### Modelica.Media.IdealGases.SingleGases.C3H6O\_propanal

Ideal gas "C3H6O\_propanal" from NASA Glenn coefficients

#### Information

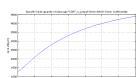


---

### Modelica.Media.IdealGases.SingleGases.C3H7\_n\_propyl

Ideal gas "C3H7\_n\_propyl" from NASA Glenn coefficients

#### Information

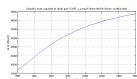


---

### Modelica.Media.IdealGases.SingleGases.C3H7\_i\_propyl

Ideal gas "C3H7\_i\_propyl" from NASA Glenn coefficients

#### Information

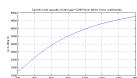


---

### Modelica.Media.IdealGases.SingleGases.C3H8

Ideal gas "C3H8" from NASA Glenn coefficients

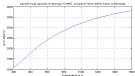
#### Information



### Modelica.Media.IdealGases.SingleGases.C3H8O\_1propanol

Ideal gas "C3H8O\_1propanol" from NASA Glenn coefficients

#### Information

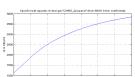


---

### Modelica.Media.IdealGases.SingleGases.C3H8O\_2propanol

Ideal gas "C3H8O\_2propanol" from NASA Glenn coefficients

#### Information

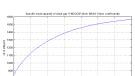


---

### Modelica.Media.IdealGases.SingleGases.CNCOCN

Ideal gas "CNCOCN" from NASA Glenn coefficients

#### Information

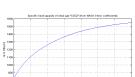


---

### Modelica.Media.IdealGases.SingleGases.C3O2

Ideal gas "C3O2" from NASA Glenn coefficients

#### Information

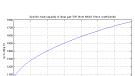


---

### Modelica.Media.IdealGases.SingleGases.C4

Ideal gas "C4" from NASA Glenn coefficients

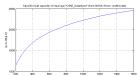
#### Information



### Modelica.Media.IdealGases.SingleGases.C4H2\_butadiyne

Ideal gas "C4H2\_butadiyne" from NASA Glenn coefficients

#### Information

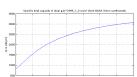


---

### Modelica.Media.IdealGases.SingleGases.C4H4\_1\_3minuscyclo

Ideal gas "C4H4\_1\_3-cyclo" from NASA Glenn coefficients

#### Information

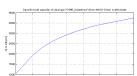


---

### Modelica.Media.IdealGases.SingleGases.C4H6\_butadiene

Ideal gas "C4H6\_butadiene" from NASA Glenn coefficients

#### Information

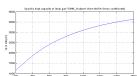


---

### Modelica.Media.IdealGases.SingleGases.C4H6\_1butyne

Ideal gas "C4H6\_1butyne" from NASA Glenn coefficients

#### Information

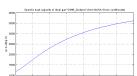


---

### Modelica.Media.IdealGases.SingleGases.C4H6\_2butyne

Ideal gas "C4H6\_2butyne" from NASA Glenn coefficients

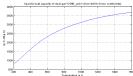
#### Information



**Modelica.Media.IdealGases.SingleGases.C4H6\_cyclo**

Ideal gas "C4H6\_cyclo" from NASA Glenn coefficients

**Information**

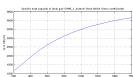


---

**Modelica.Media.IdealGases.SingleGases.C4H8\_1\_butene**

Ideal gas "C4H8\_1\_butene" from NASA Glenn coefficients

**Information**

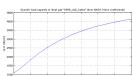


---

**Modelica.Media.IdealGases.SingleGases.C4H8\_cis2\_buten**

Ideal gas "C4H8\_cis2\_buten" from NASA Glenn coefficients

**Information**

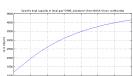


---

**Modelica.Media.IdealGases.SingleGases.C4H8\_isobutene**

Ideal gas "C4H8\_isobutene" from NASA Glenn coefficients

**Information**

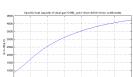


---

**Modelica.Media.IdealGases.SingleGases.C4H8\_cyclo**

Ideal gas "C4H8\_cyclo" from NASA Glenn coefficients

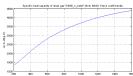
**Information**



### Modelica.Media.IdealGases.SingleGases.C4H9\_n\_butyl

Ideal gas "C4H9\_n\_butyl" from NASA Glenn coefficients

#### Information

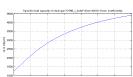


---

### Modelica.Media.IdealGases.SingleGases.C4H9\_i\_butyl

Ideal gas "C4H9\_i\_butyl" from NASA Glenn coefficients

#### Information

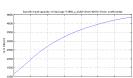


---

### Modelica.Media.IdealGases.SingleGases.C4H9\_s\_butyl

Ideal gas "C4H9\_s\_butyl" from NASA Glenn coefficients

#### Information

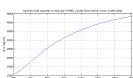


---

### Modelica.Media.IdealGases.SingleGases.C4H9\_t\_butyl

Ideal gas "C4H9\_t\_butyl" from NASA Glenn coefficients

#### Information

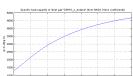


---

### Modelica.Media.IdealGases.SingleGases.C4H10\_n\_butane

Ideal gas "C4H10\_n\_butane" from NASA Glenn coefficients

#### Information



---

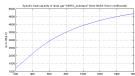
## 1370 Modelica.Media.IdealGases.SingleGases.C4H10\_isobutane

---

### Modelica.Media.IdealGases.SingleGases.C4H10\_isobutane

Ideal gas "C4H10\_isobutane" from NASA Glenn coefficients

#### Information

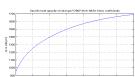


---

### Modelica.Media.IdealGases.SingleGases.C4N2

Ideal gas "C4N2" from NASA Glenn coefficients

#### Information

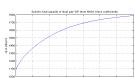


---

### Modelica.Media.IdealGases.SingleGases.C5

Ideal gas "C5" from NASA Glenn coefficients

#### Information

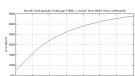


---

### Modelica.Media.IdealGases.SingleGases.C5H6\_1\_3cyclo

Ideal gas "C5H6\_1\_3cyclo" from NASA Glenn coefficients

#### Information

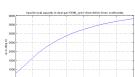


---

### Modelica.Media.IdealGases.SingleGases.C5H8\_cyclo

Ideal gas "C5H8\_cyclo" from NASA Glenn coefficients

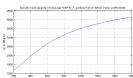
#### Information



### **Modelica.Media.IdealGases.SingleGases.C5H10\_1\_pentene**

Ideal gas "C5H10\_1\_pentene" from NASA Glenn coefficients

#### **Information**

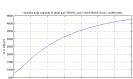


---

### **Modelica.Media.IdealGases.SingleGases.C5H10\_cyclo**

Ideal gas "C5H10\_cyclo" from NASA Glenn coefficients

#### **Information**

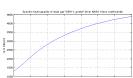


---

### **Modelica.Media.IdealGases.SingleGases.C5H11\_pentyl**

Ideal gas "C5H11\_pentyl" from NASA Glenn coefficients

#### **Information**

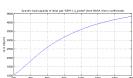


---

### **Modelica.Media.IdealGases.SingleGases.C5H11\_t\_pentyl**

Ideal gas "C5H11\_t\_pentyl" from NASA Glenn coefficients

#### **Information**

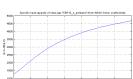


---

### **Modelica.Media.IdealGases.SingleGases.C5H12\_n\_pentane**

Ideal gas "C5H12\_n\_pentane" from NASA Glenn coefficients

#### **Information**



---

## 1372 Modelica.Media.IdealGases.SingleGases.C5H12\_i\_pentane

---

### Modelica.Media.IdealGases.SingleGases.C5H12\_i\_pentane

Ideal gas "C5H12\_i\_pentane" from NASA Glenn coefficients

#### Information

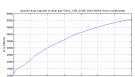


---

### Modelica.Media.IdealGases.SingleGases.CH3C\_CH3\_2CH3

Ideal gas "CH3C\_CH3\_2CH3" from NASA Glenn coefficients

#### Information

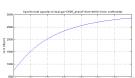


---

### Modelica.Media.IdealGases.SingleGases.C6D5\_phenyl

Ideal gas "C6D5\_phenyl" from NASA Glenn coefficients

#### Information

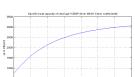


---

### Modelica.Media.IdealGases.SingleGases.C6D6

Ideal gas "C6D6" from NASA Glenn coefficients

#### Information

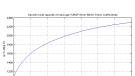


---

### Modelica.Media.IdealGases.SingleGases.C6H2

Ideal gas "C6H2" from NASA Glenn coefficients

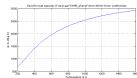
#### Information



## Modelica.Media.IdealGases.SingleGases.C6H5\_phenyl

Ideal gas "C6H5\_phenyl" from NASA Glenn coefficients

### Information

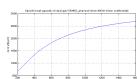


---

## Modelica.Media.IdealGases.SingleGases.C6H5O\_phenoxy

Ideal gas "C6H5O\_phenoxy" from NASA Glenn coefficients

### Information

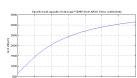


---

## Modelica.Media.IdealGases.SingleGases.C6H6

Ideal gas "C6H6" from NASA Glenn coefficients

### Information

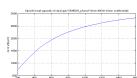


---

## Modelica.Media.IdealGases.SingleGases.C6H5OH\_phenol

Ideal gas "C6H5OH\_phenol" from NASA Glenn coefficients

### Information

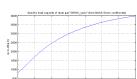


---

## Modelica.Media.IdealGases.SingleGases.C6H10\_cyclo

Ideal gas "C6H10\_cyclo" from NASA Glenn coefficients

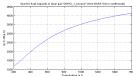
### Information



**Modelica.Media.IdealGases.SingleGases.C6H12\_1\_hexene**

Ideal gas "C6H12\_1\_hexene" from NASA Glenn coefficients

**Information**

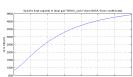


---

**Modelica.Media.IdealGases.SingleGases.C6H12\_cyclo**

Ideal gas "C6H12\_cyclo" from NASA Glenn coefficients

**Information**

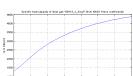


---

**Modelica.Media.IdealGases.SingleGases.C6H13\_n\_hexyl**

Ideal gas "C6H13\_n\_hexyl" from NASA Glenn coefficients

**Information**

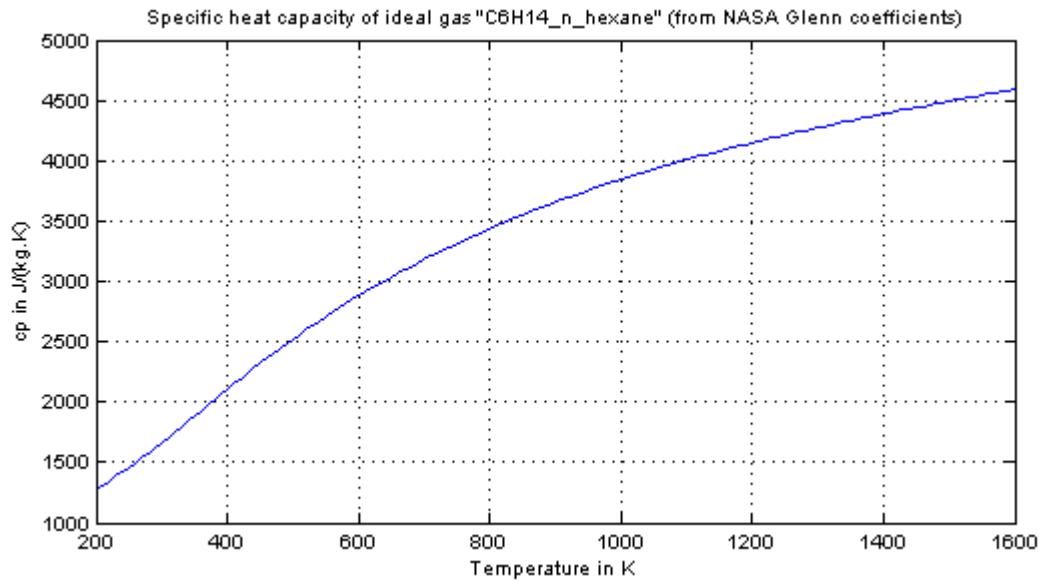


---

**Modelica.Media.IdealGases.SingleGases.C6H14\_n\_hexane**

Ideal gas "C6H14\_n\_hexane" from NASA Glenn coefficients

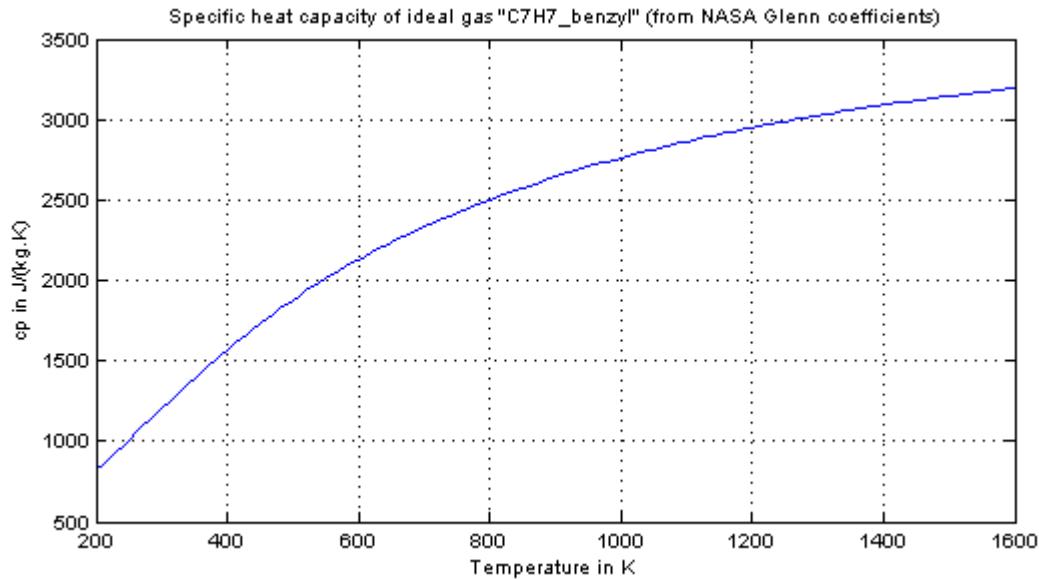
## Information



## Modelica.Media.IdealGases.SingleGases.C7H7\_benzyl

Ideal gas "C7H7\_benzyl" from NASA Glenn coefficients

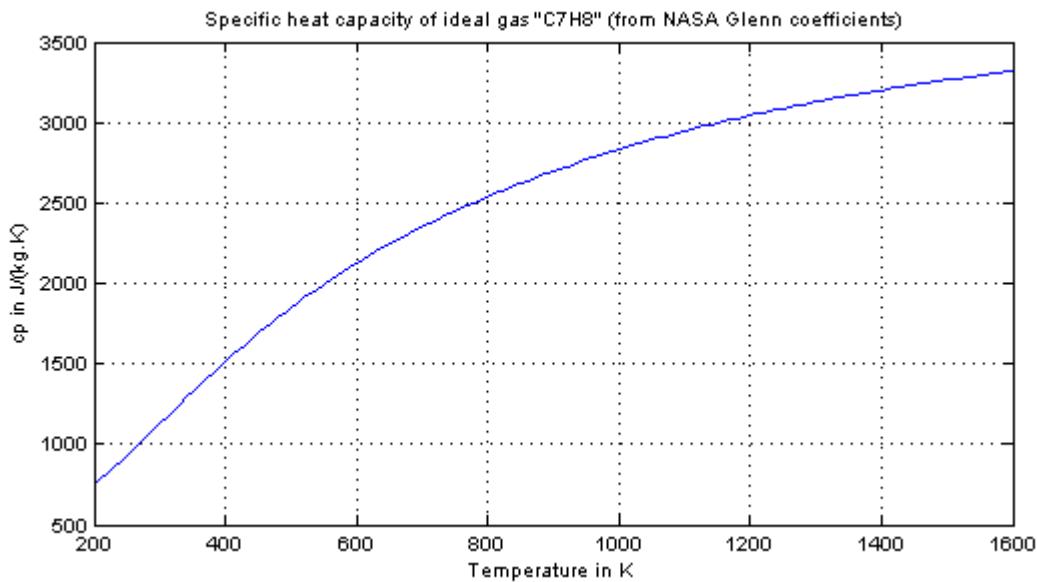
## Information



## Modelica.Media.IdealGases.SingleGases.C7H8

Ideal gas "C7H8" from NASA Glenn coefficients

## Information

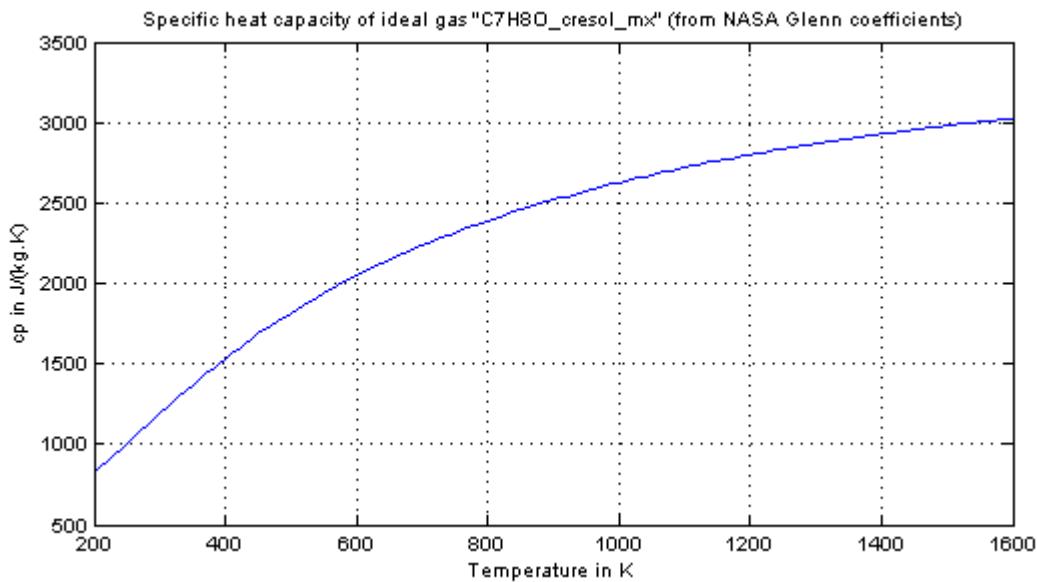


---

## Modelica.Media.IdealGases.SingleGases.C7H8O\_cresol\_mx

Ideal gas "C7H8O\_cresol\_mx" from NASA Glenn coefficients

## Information

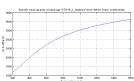


---

## Modelica.Media.IdealGases.SingleGases.C7H14\_1\_heptene

Ideal gas "C7H14\_1\_heptene" from NASA Glenn coefficients

**Information**

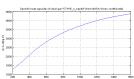


---

**Modelica.Media.IdealGases.SingleGases.C7H15\_n\_heptyl**

Ideal gas "C7H15\_n\_heptyl" from NASA Glenn coefficients

**Information**

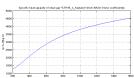


---

**Modelica.Media.IdealGases.SingleGases.C7H16\_n\_heptane**

Ideal gas "C7H16\_n\_heptane" from NASA Glenn coefficients

**Information**

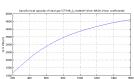


---

**Modelica.Media.IdealGases.SingleGases.C7H16\_2\_methylh**

Ideal gas "C7H16\_2\_methylh" from NASA Glenn coefficients

**Information**

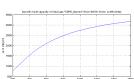


---

**Modelica.Media.IdealGases.SingleGases.C8H8\_styrene**

Ideal gas "C8H8\_styrene" from NASA Glenn coefficients

**Information**



---

**Modelica.Media.IdealGases.SingleGases.C8H10\_ethylbenz**

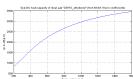
Ideal gas "C8H10\_ethylbenz" from NASA Glenn coefficients

---

## **1378 Modelica.Media.IdealGases.SingleGases.C8H10\_ethylbenz**

---

### **Information**

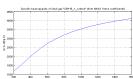


---

## **Modelica.Media.IdealGases.SingleGases.C8H16\_1\_octene**

Ideal gas "C8H16\_1\_octene" from NASA Glenn coefficients

### **Information**

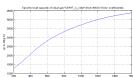


---

## **Modelica.Media.IdealGases.SingleGases.C8H17\_n\_octyl**

Ideal gas "C8H17\_n\_octyl" from NASA Glenn coefficients

### **Information**

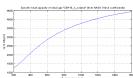


---

## **Modelica.Media.IdealGases.SingleGases.C8H18\_n\_octane**

Ideal gas "C8H18\_n\_octane" from NASA Glenn coefficients

### **Information**

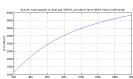


---

## **Modelica.Media.IdealGases.SingleGases.C8H18\_isooctane**

Ideal gas "C8H18\_isooctane" from NASA Glenn coefficients

### **Information**

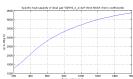


---

## **Modelica.Media.IdealGases.SingleGases.C9H19\_n\_nonyl**

Ideal gas "C9H19\_n\_nonyl" from NASA Glenn coefficients

**Information**

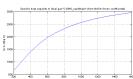


---

**Modelica.Media.IdealGases.SingleGases.C10H8\_naphthalene**

Ideal gas "C10H8\_naphthalene" from NASA Glenn coefficients

**Information**

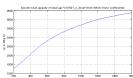


---

**Modelica.Media.IdealGases.SingleGases.C10H21\_n\_decyl**

Ideal gas "C10H21\_n\_decyl" from NASA Glenn coefficients

**Information**

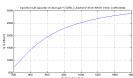


---

**Modelica.Media.IdealGases.SingleGases.C12H9\_o\_biphenyl**

Ideal gas "C12H9\_o\_biphenyl" from NASA Glenn coefficients

**Information**

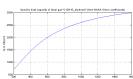


---

**Modelica.Media.IdealGases.SingleGases.C12H10\_biphenyl**

Ideal gas "C12H10\_biphenyl" from NASA Glenn coefficients

**Information**

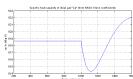


---

**Modelica.Media.IdealGases.SingleGases.Ca**

Ideal gas "Ca" from NASA Glenn coefficients

**Information**

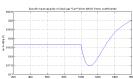


---

**Modelica.Media.IdealGases.SingleGases.CaPlus**

Ideal gas "Ca+" from NASA Glenn coefficients

**Information**

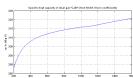


---

**Modelica.Media.IdealGases.SingleGases.CaBr**

Ideal gas "CaBr" from NASA Glenn coefficients

**Information**

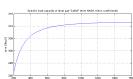


---

**Modelica.Media.IdealGases.SingleGases.CaBr2**

Ideal gas "CaBr2" from NASA Glenn coefficients

**Information**



---

**Modelica.Media.IdealGases.SingleGases.CaCL**

Ideal gas "CaCl" from NASA Glenn coefficients

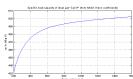
**Information**



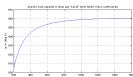
---

**Modelica.Media.IdealGases.SingleGases.CaCLplus**

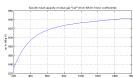
Ideal gas "CaCl+" from NASA Glenn coefficients

**Information****Modelica.Media.IdealGases.SingleGases.CaCL2**

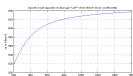
Ideal gas "CaCl2" from NASA Glenn coefficients

**Information****Modelica.Media.IdealGases.SingleGases.CaF**

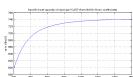
Ideal gas "CaF" from NASA Glenn coefficients

**Information****Modelica.Media.IdealGases.SingleGases.CaFplus**

Ideal gas "CaF+" from NASA Glenn coefficients

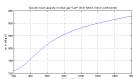
**Information****Modelica.Media.IdealGases.SingleGases.CaF2**

Ideal gas "CaF2" from NASA Glenn coefficients

**Information****Modelica.Media.IdealGases.SingleGases.CaH**

Ideal gas "CaH" from NASA Glenn coefficients

**Information**

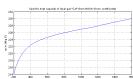


---

**Modelica.Media.IdealGases.SingleGases.Cal**

Ideal gas "Cal" from NASA Glenn coefficients

**Information**

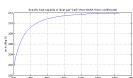


---

**Modelica.Media.IdealGases.SingleGases.Cal2**

Ideal gas "Cal2" from NASA Glenn coefficients

**Information**



---

**Modelica.Media.IdealGases.SingleGases.CaO**

Ideal gas "CaO" from NASA Glenn coefficients

**Information**

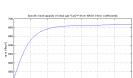


---

**Modelica.Media.IdealGases.SingleGases.CaOplus**

Ideal gas "CaO+" from NASA Glenn coefficients

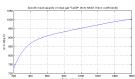
**Information**



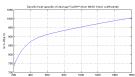
---

**Modelica.Media.IdealGases.SingleGases.CaOH**

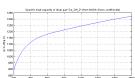
Ideal gas "CaOH" from NASA Glenn coefficients

**Information****Modelica.Media.IdealGases.SingleGases.CaOHplus**

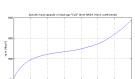
Ideal gas "CaOH+" from NASA Glenn coefficients

**Information****Modelica.Media.IdealGases.SingleGases.Ca\_OH\_2**

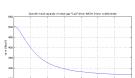
Ideal gas "Ca\_OH\_2" from NASA Glenn coefficients

**Information****Modelica.Media.IdealGases.SingleGases.CaS**

Ideal gas "CaS" from NASA Glenn coefficients

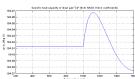
**Information****Modelica.Media.IdealGases.SingleGases.Ca2**

Ideal gas "Ca2" from NASA Glenn coefficients

**Information****Modelica.Media.IdealGases.SingleGases.Cd**

Ideal gas "Cd" from NASA Glenn coefficients

**Information**

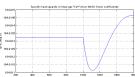


---

**Modelica.Media.IdealGases.SingleGases.Cdplus**

Ideal gas "Cd+" from NASA Glenn coefficients

**Information**

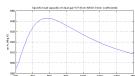


---

**Modelica.Media.IdealGases.SingleGases.CL**

Ideal gas "Cl" from NASA Glenn coefficients

**Information**

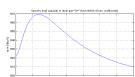


---

**Modelica.Media.IdealGases.SingleGases.CLplus**

Ideal gas "Cl+" from NASA Glenn coefficients

**Information**



---

**Modelica.Media.IdealGases.SingleGases.CLminus**

Ideal gas "Cl-" from NASA Glenn coefficients

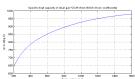
**Information**



---

**Modelica.Media.IdealGases.SingleGases.CLCN**

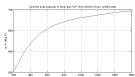
Ideal gas "CICN" from NASA Glenn coefficients

**Information**

---

**Modelica.Media.IdealGases.SingleGases.CLF**

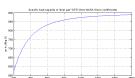
Ideal gas "CIF" from NASA Glenn coefficients

**Information**

---

**Modelica.Media.IdealGases.SingleGases.CLF3**

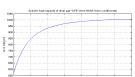
Ideal gas "CIF3" from NASA Glenn coefficients

**Information**

---

**Modelica.Media.IdealGases.SingleGases.CLF5**

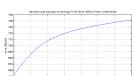
Ideal gas "CIF5" from NASA Glenn coefficients

**Information**

---

**Modelica.Media.IdealGases.SingleGases.CLO**

Ideal gas "CIO" from NASA Glenn coefficients

**Information**

---

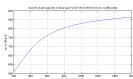
**Modelica.Media.IdealGases.SingleGases.CLO2**

Ideal gas "CIO2" from NASA Glenn coefficients

## 1386 Modelica.Media.IdealGases.SingleGases.CLO2

---

### Information

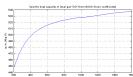


---

## Modelica.Media.IdealGases.SingleGases.CL2

Ideal gas "CL2" from NASA Glenn coefficients

### Information

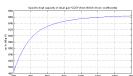


---

## Modelica.Media.IdealGases.SingleGases.CL2O

Ideal gas "CL2O" from NASA Glenn coefficients

### Information

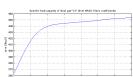


---

## Modelica.Media.IdealGases.SingleGases.Co

Ideal gas "Co" from NASA Glenn coefficients

### Information

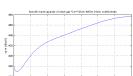


---

## Modelica.Media.IdealGases.SingleGases.Coplus

Ideal gas "Co+" from NASA Glenn coefficients

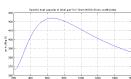
### Information



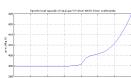
---

## Modelica.Media.IdealGases.SingleGases.Cominus

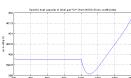
Ideal gas "Co-" from NASA Glenn coefficients

**Information****Modelica.Media.IdealGases.SingleGases.Cr**

Ideal gas "Cr" from NASA Glenn coefficients

**Information****Modelica.Media.IdealGases.SingleGases.Crplus**

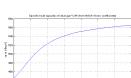
Ideal gas "Cr+" from NASA Glenn coefficients

**Information****Modelica.Media.IdealGases.SingleGases.Crminus**

Ideal gas "Cr-" from NASA Glenn coefficients

**Information****Modelica.Media.IdealGases.SingleGases.CrN**

Ideal gas "CrN" from NASA Glenn coefficients

**Information****Modelica.Media.IdealGases.SingleGases.CrO**

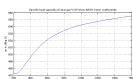
Ideal gas "CrO" from NASA Glenn coefficients

---

## 1388 Modelica.Media.IdealGases.SingleGases.CrO

---

### Information

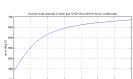


---

## Modelica.Media.IdealGases.SingleGases.CrO2

Ideal gas "CrO2" from NASA Glenn coefficients

### Information

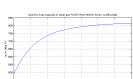


---

## Modelica.Media.IdealGases.SingleGases.CrO3

Ideal gas "CrO3" from NASA Glenn coefficients

### Information

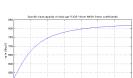


---

## Modelica.Media.IdealGases.SingleGases.CrO3minus

Ideal gas "CrO3-" from NASA Glenn coefficients

### Information

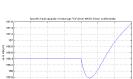


---

## Modelica.Media.IdealGases.SingleGases.Cs

Ideal gas "Cs" from NASA Glenn coefficients

### Information



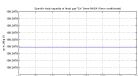
---

## Modelica.Media.IdealGases.SingleGases.Csplus

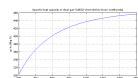
Ideal gas "Cs+" from NASA Glenn coefficients

**Information****Modelica.Media.IdealGases.SingleGases.Csminus**

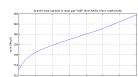
Ideal gas "Cs-" from NASA Glenn coefficients

**Information****Modelica.Media.IdealGases.SingleGases.CsBO2**

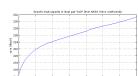
Ideal gas "CsBO2" from NASA Glenn coefficients

**Information****Modelica.Media.IdealGases.SingleGases.CsBr**

Ideal gas "CsBr" from NASA Glenn coefficients

**Information****Modelica.Media.IdealGases.SingleGases.CsCL**

Ideal gas "CsCl" from NASA Glenn coefficients

**Information****Modelica.Media.IdealGases.SingleGases.CsF**

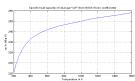
Ideal gas "CsF" from NASA Glenn coefficients

---

## 1390 Modelica.Media.IdealGases.SingleGases.CsF

---

### Information

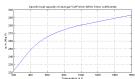


---

## Modelica.Media.IdealGases.SingleGases.CsH

Ideal gas "CsH" from NASA Glenn coefficients

### Information

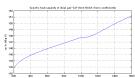


---

## Modelica.Media.IdealGases.SingleGases.CsI

Ideal gas "CsI" from NASA Glenn coefficients

### Information

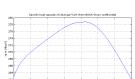


---

## Modelica.Media.IdealGases.SingleGases.CsLi

Ideal gas "CsLi" from NASA Glenn coefficients

### Information

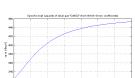


---

## Modelica.Media.IdealGases.SingleGases.CsNO2

Ideal gas "CsNO2" from NASA Glenn coefficients

### Information

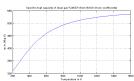


---

## Modelica.Media.IdealGases.SingleGases.CsNO3

Ideal gas "CsNO3" from NASA Glenn coefficients

**Information**

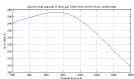


---

**Modelica.Media.IdealGases.SingleGases.CsNa**

Ideal gas "CsNa" from NASA Glenn coefficients

**Information**

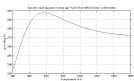


---

**Modelica.Media.IdealGases.SingleGases.CsO**

Ideal gas "CsO" from NASA Glenn coefficients

**Information**

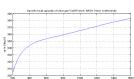


---

**Modelica.Media.IdealGases.SingleGases.CsOH**

Ideal gas "CsOH" from NASA Glenn coefficients

**Information**



---

**Modelica.Media.IdealGases.SingleGases.CsRb**

Ideal gas "CsRb" from NASA Glenn coefficients

**Information**



---

**Modelica.Media.IdealGases.SingleGases.Cs2**

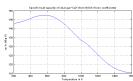
Ideal gas "Cs2" from NASA Glenn coefficients

---

## 1392 Modelica.Media.IdealGases.SingleGases.Cs2

---

### Information

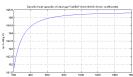


---

## Modelica.Media.IdealGases.SingleGases.Cs2Br2

Ideal gas "Cs2Br2" from NASA Glenn coefficients

### Information

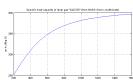


---

## Modelica.Media.IdealGases.SingleGases.Cs2CO3

Ideal gas "Cs2CO3" from NASA Glenn coefficients

### Information

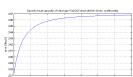


---

## Modelica.Media.IdealGases.SingleGases.Cs2Cl2

Ideal gas "Cs2Cl2" from NASA Glenn coefficients

### Information

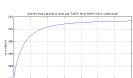


---

## Modelica.Media.IdealGases.SingleGases.Cs2F2

Ideal gas "Cs2F2" from NASA Glenn coefficients

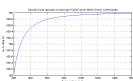
### Information



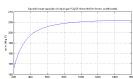
---

## Modelica.Media.IdealGases.SingleGases.Cs2I2

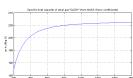
Ideal gas "Cs2I2" from NASA Glenn coefficients

**Information****Modelica.Media.IdealGases.SingleGases.Cs2O**

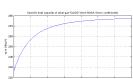
Ideal gas "Cs2O" from NASA Glenn coefficients

**Information****Modelica.Media.IdealGases.SingleGases.Cs2Oplus**

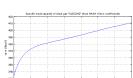
Ideal gas "Cs2O+" from NASA Glenn coefficients

**Information****Modelica.Media.IdealGases.SingleGases.Cs2O2**

Ideal gas "Cs2O2" from NASA Glenn coefficients

**Information****Modelica.Media.IdealGases.SingleGases.Cs2O2H2**

Ideal gas "Cs2O2H2" from NASA Glenn coefficients

**Information****Modelica.Media.IdealGases.SingleGases.Cs2SO4**

Ideal gas "Cs2SO4" from NASA Glenn coefficients

## 1394 Modelica.Media.IdealGases.SingleGases.Cs2SO4

---

### Information



---

## Modelica.Media.IdealGases.SingleGases.Cu

Ideal gas "Cu" from NASA Glenn coefficients

### Information

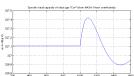


---

## Modelica.Media.IdealGases.SingleGases.Cuplus

Ideal gas "Cu+" from NASA Glenn coefficients

### Information

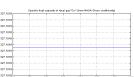


---

## Modelica.Media.IdealGases.SingleGases.Cuminus

Ideal gas "Cu-" from NASA Glenn coefficients

### Information

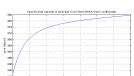


---

## Modelica.Media.IdealGases.SingleGases.CuCL

Ideal gas "CuCl" from NASA Glenn coefficients

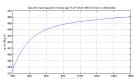
### Information



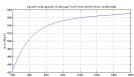
---

## Modelica.Media.IdealGases.SingleGases.CuF

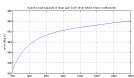
Ideal gas "CuF" from NASA Glenn coefficients

**Information****Modelica.Media.IdealGases.SingleGases.CuF2**

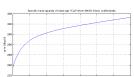
Ideal gas "CuF2" from NASA Glenn coefficients

**Information****Modelica.Media.IdealGases.SingleGases.CuO**

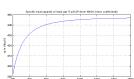
Ideal gas "CuO" from NASA Glenn coefficients

**Information****Modelica.Media.IdealGases.SingleGases.Cu2**

Ideal gas "Cu2" from NASA Glenn coefficients

**Information****Modelica.Media.IdealGases.SingleGases.Cu3Cl3**

Ideal gas "Cu3Cl3" from NASA Glenn coefficients

**Information****Modelica.Media.IdealGases.SingleGases.D**

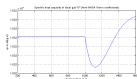
Ideal gas "D" from NASA Glenn coefficients

---

## 1396 Modelica.Media.IdealGases.SingleGases.D

---

### Information

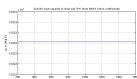


---

## Modelica.Media.IdealGases.SingleGases.Dplus

Ideal gas "D+" from NASA Glenn coefficients

### Information



---

## Modelica.Media.IdealGases.SingleGases.Dminus

Ideal gas "D-" from NASA Glenn coefficients

### Information

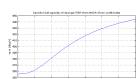


---

## Modelica.Media.IdealGases.SingleGases.DBr

Ideal gas "DBr" from NASA Glenn coefficients

### Information

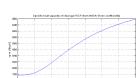


---

## Modelica.Media.IdealGases.SingleGases.DCL

Ideal gas "DCI" from NASA Glenn coefficients

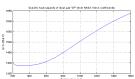
### Information



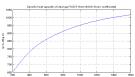
---

## Modelica.Media.IdealGases.SingleGases.DF

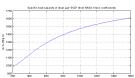
Ideal gas "DF" from NASA Glenn coefficients

**Information****Modelica.Media.IdealGases.SingleGases.DOCL**

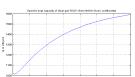
Ideal gas "DOCI" from NASA Glenn coefficients

**Information****Modelica.Media.IdealGases.SingleGases.DO2**

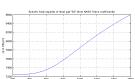
Ideal gas "DO2" from NASA Glenn coefficients

**Information****Modelica.Media.IdealGases.SingleGases.DO2minus**

Ideal gas "DO2-" from NASA Glenn coefficients

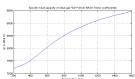
**Information****Modelica.Media.IdealGases.SingleGases.D2**

Ideal gas "D2" from NASA Glenn coefficients

**Information****Modelica.Media.IdealGases.SingleGases.D2plus**

Ideal gas "D2+" from NASA Glenn coefficients

## Information

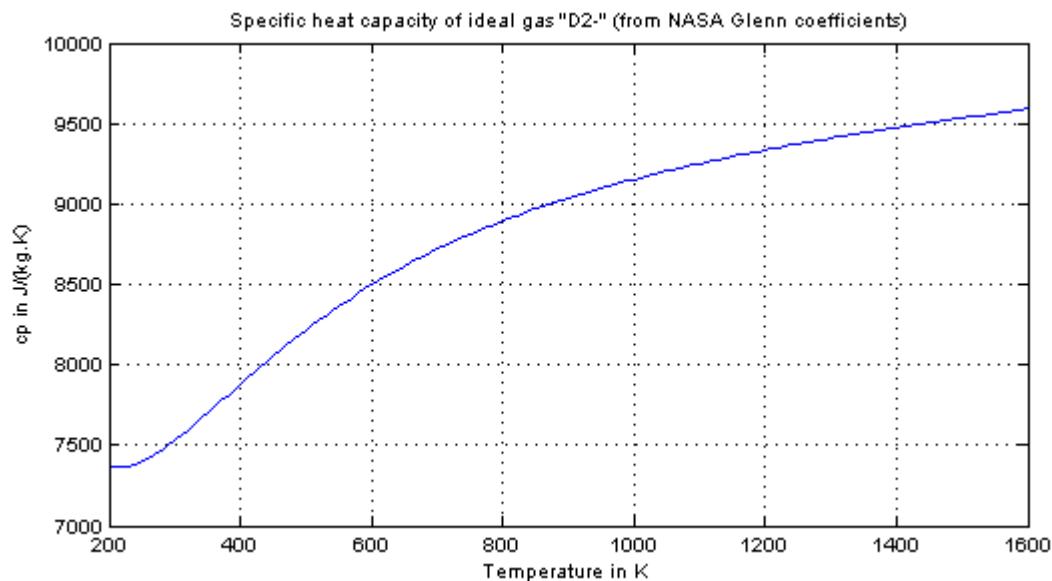


---

## Modelica.Media.IdealGases.SingleGases.D2minus

Ideal gas "D2-" from NASA Glenn coefficients

## Information

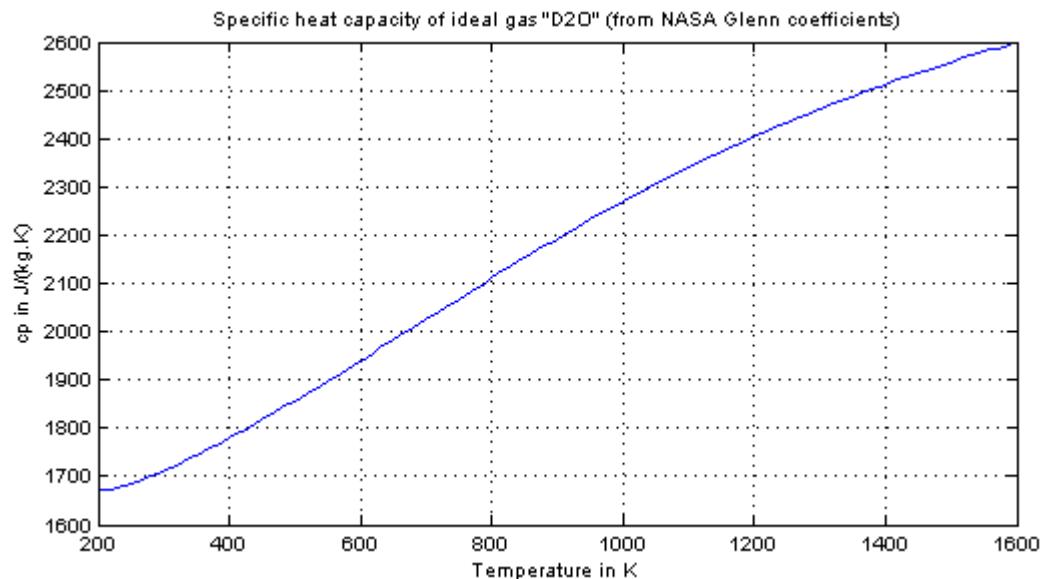


---

## Modelica.Media.IdealGases.SingleGases.D2O

Ideal gas "D2O" from NASA Glenn coefficients

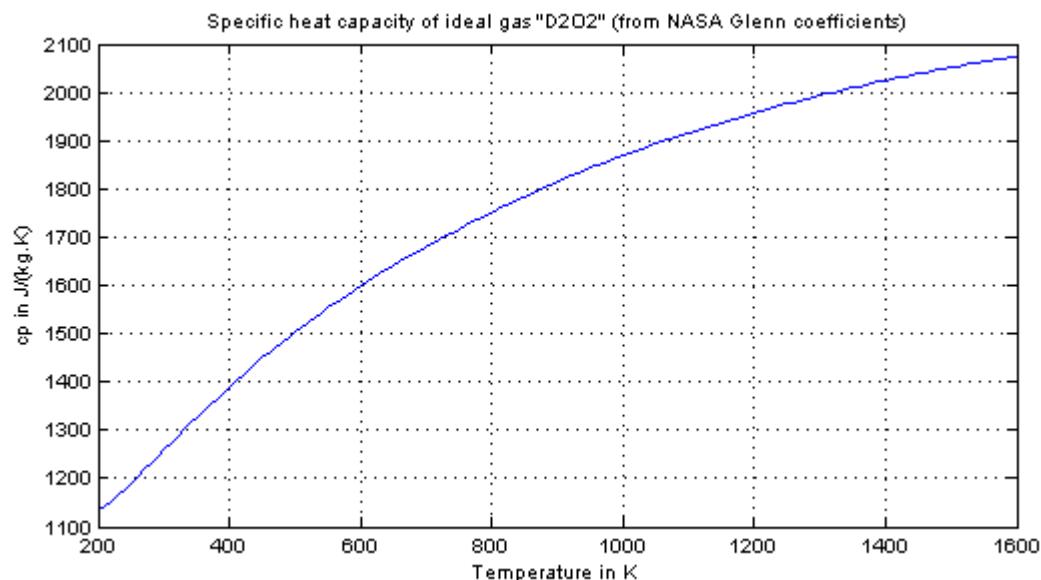
## Information



## Modelica.Media.IdealGases.SingleGases.D2O2

Ideal gas "D2O2" from NASA Glenn coefficients

## Information



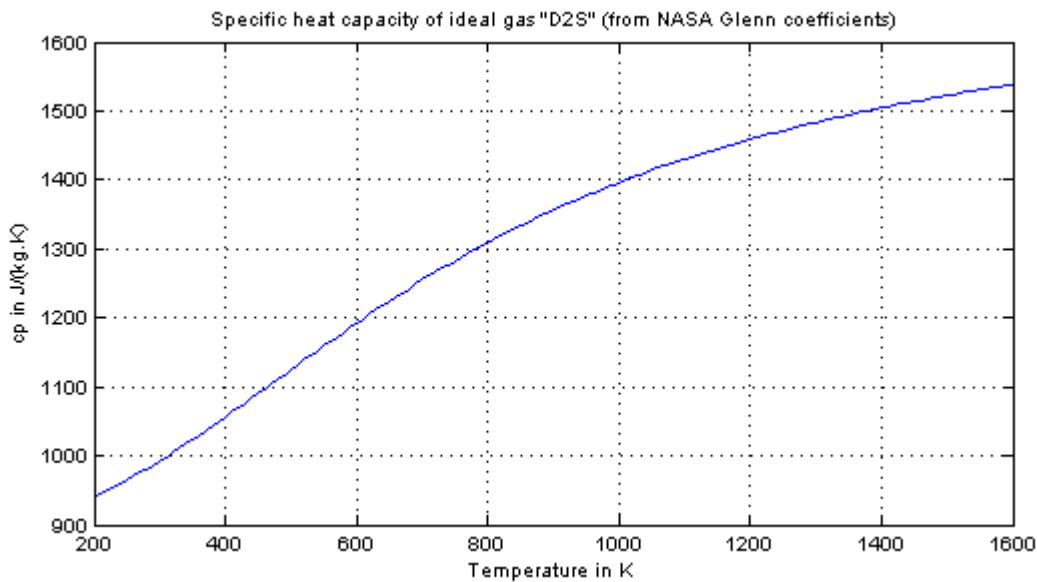
## Modelica.Media.IdealGases.SingleGases.D2S

Ideal gas "D2S" from NASA Glenn coefficients

## 1400 Modelica.Media.IdealGases.SingleGases.D2S

---

### Information



## Modelica.Media.IdealGases.SingleGases.eminus

Ideal gas "e-" from NASA Glenn coefficients

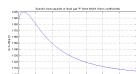
### Information



## Modelica.Media.IdealGases.SingleGases.F

Ideal gas "F" from NASA Glenn coefficients

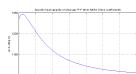
### Information



## Modelica.Media.IdealGases.SingleGases.Fplus

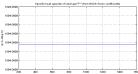
Ideal gas "F+" from NASA Glenn coefficients

### Information

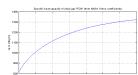


**Modelica.Media.IdealGases.SingleGases.Fminus**

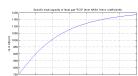
Ideal gas "F-" from NASA Glenn coefficients

**Information****Modelica.Media.IdealGases.SingleGases.FCN**

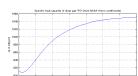
Ideal gas "FCN" from NASA Glenn coefficients

**Information****Modelica.Media.IdealGases.SingleGases.FCO**

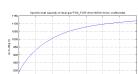
Ideal gas "FCO" from NASA Glenn coefficients

**Information****Modelica.Media.IdealGases.SingleGases.FO**

Ideal gas "FO" from NASA Glenn coefficients

**Information****Modelica.Media.IdealGases.SingleGases.FO2\_FOO**

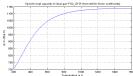
Ideal gas "FO2\_FOO" from NASA Glenn coefficients

**Information**

## Modelica.Media.IdealGases.SingleGases.FO2\_OFO

Ideal gas "FO2\_OFO" from NASA Glenn coefficients

### Information

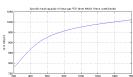


---

## Modelica.Media.IdealGases.SingleGases.F2

Ideal gas "F2" from NASA Glenn coefficients

### Information

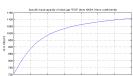


---

## Modelica.Media.IdealGases.SingleGases.F2O

Ideal gas "F2O" from NASA Glenn coefficients

### Information

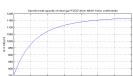


---

## Modelica.Media.IdealGases.SingleGases.F2O2

Ideal gas "F2O2" from NASA Glenn coefficients

### Information

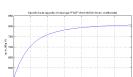


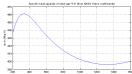
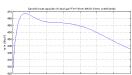
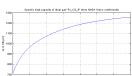
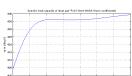
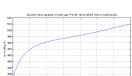
---

## Modelica.Media.IdealGases.SingleGases.FS2F

Ideal gas "FS2F" from NASA Glenn coefficients

### Information

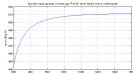


**Modelica.Media.IdealGases.SingleGases.Fe****Ideal gas "Fe" from NASA Glenn coefficients****Information****Modelica.Media.IdealGases.SingleGases.Feplus****Ideal gas "Fe+" from NASA Glenn coefficients****Information****Modelica.Media.IdealGases.SingleGases.Fe\_CO\_5****Ideal gas "Fe\_CO\_5" from NASA Glenn coefficients****Information****Modelica.Media.IdealGases.SingleGases.FeCL****Ideal gas "FeCl" from NASA Glenn coefficients****Information****Modelica.Media.IdealGases.SingleGases.FeCL2****Ideal gas "FeCl2" from NASA Glenn coefficients****Information**

### Modelica.Media.IdealGases.SingleGases.FeCL3

Ideal gas "FeCl3" from NASA Glenn coefficients

#### Information

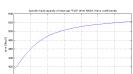


---

### Modelica.Media.IdealGases.SingleGases.FeO

Ideal gas "FeO" from NASA Glenn coefficients

#### Information

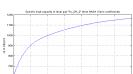


---

### Modelica.Media.IdealGases.SingleGases.Fe\_OH\_2

Ideal gas "Fe\_OH\_2" from NASA Glenn coefficients

#### Information

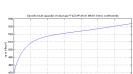


---

### Modelica.Media.IdealGases.SingleGases.Fe2CL4

Ideal gas "Fe2Cl4" from NASA Glenn coefficients

#### Information



---

### Modelica.Media.IdealGases.SingleGases.Fe2CL6

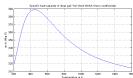
Ideal gas "Fe2Cl6" from NASA Glenn coefficients

#### Information

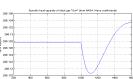


**Modelica.Media.IdealGases.SingleGases.Ga**

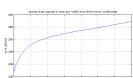
Ideal gas "Ga" from NASA Glenn coefficients

**Information****Modelica.Media.IdealGases.SingleGases.Gaplus**

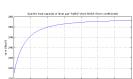
Ideal gas "Ga+" from NASA Glenn coefficients

**Information****Modelica.Media.IdealGases.SingleGases.GaBr**

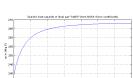
Ideal gas "GaBr" from NASA Glenn coefficients

**Information****Modelica.Media.IdealGases.SingleGases.GaBr2**

Ideal gas "GaBr2" from NASA Glenn coefficients

**Information****Modelica.Media.IdealGases.SingleGases.GaBr3**

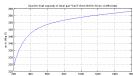
Ideal gas "GaBr3" from NASA Glenn coefficients

**Information**

### Modelica.Media.IdealGases.SingleGases.GaCL

Ideal gas "GaCl" from NASA Glenn coefficients

#### Information

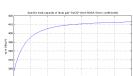


---

### Modelica.Media.IdealGases.SingleGases.GaCL2

Ideal gas "GaCl2" from NASA Glenn coefficients

#### Information

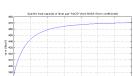


---

### Modelica.Media.IdealGases.SingleGases.GaCL3

Ideal gas "GaCl3" from NASA Glenn coefficients

#### Information

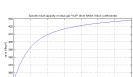


---

### Modelica.Media.IdealGases.SingleGases.GaF

Ideal gas "GaF" from NASA Glenn coefficients

#### Information

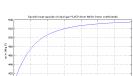


---

### Modelica.Media.IdealGases.SingleGases.GaF2

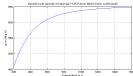
Ideal gas "GaF2" from NASA Glenn coefficients

#### Information

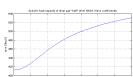


**Modelica.Media.IdealGases.SingleGases.GaF3**

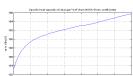
Ideal gas "GaF3" from NASA Glenn coefficients

**Information****Modelica.Media.IdealGases.SingleGases.GaH**

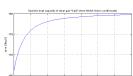
Ideal gas "GaH" from NASA Glenn coefficients

**Information****Modelica.Media.IdealGases.SingleGases.GaI**

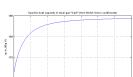
Ideal gas "GaI" from NASA Glenn coefficients

**Information****Modelica.Media.IdealGases.SingleGases.GaI2**

Ideal gas "GaI2" from NASA Glenn coefficients

**Information****Modelica.Media.IdealGases.SingleGases.GaI3**

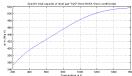
Ideal gas "GaI3" from NASA Glenn coefficients

**Information**

### Modelica.Media.IdealGases.SingleGases.GaO

Ideal gas "GaO" from NASA Glenn coefficients

#### Information

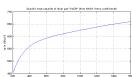


---

### Modelica.Media.IdealGases.SingleGases.GaOH

Ideal gas "GaOH" from NASA Glenn coefficients

#### Information

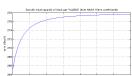


---

### Modelica.Media.IdealGases.SingleGases.Ga2Br2

Ideal gas "Ga2Br2" from NASA Glenn coefficients

#### Information

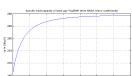


---

### Modelica.Media.IdealGases.SingleGases.Ga2Br4

Ideal gas "Ga2Br4" from NASA Glenn coefficients

#### Information

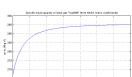


---

### Modelica.Media.IdealGases.SingleGases.Ga2Br6

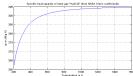
Ideal gas "Ga2Br6" from NASA Glenn coefficients

#### Information

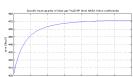


**Modelica.Media.IdealGases.SingleGases.Ga2CL2**

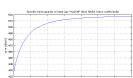
Ideal gas "Ga2Cl2" from NASA Glenn coefficients

**Information****Modelica.Media.IdealGases.SingleGases.Ga2CL4**

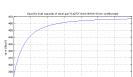
Ideal gas "Ga2Cl4" from NASA Glenn coefficients

**Information****Modelica.Media.IdealGases.SingleGases.Ga2CL6**

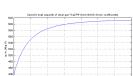
Ideal gas "Ga2Cl6" from NASA Glenn coefficients

**Information****Modelica.Media.IdealGases.SingleGases.Ga2F2**

Ideal gas "Ga2F2" from NASA Glenn coefficients

**Information****Modelica.Media.IdealGases.SingleGases.Ga2F4**

Ideal gas "Ga2F4" from NASA Glenn coefficients

**Information**

---

## 1410 Modelica.Media.IdealGases.SingleGases.Ga2F6

---

### Modelica.Media.IdealGases.SingleGases.Ga2F6

Ideal gas "Ga2F6" from NASA Glenn coefficients

#### Information

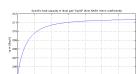


---

### Modelica.Media.IdealGases.SingleGases.Ga2I2

Ideal gas "Ga2I2" from NASA Glenn coefficients

#### Information

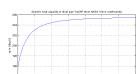


---

### Modelica.Media.IdealGases.SingleGases.Ga2I4

Ideal gas "Ga2I4" from NASA Glenn coefficients

#### Information

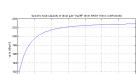


---

### Modelica.Media.IdealGases.SingleGases.Ga2I6

Ideal gas "Ga2I6" from NASA Glenn coefficients

#### Information

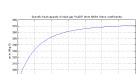


---

### Modelica.Media.IdealGases.SingleGases.Ga2O

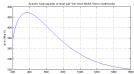
Ideal gas "Ga2O" from NASA Glenn coefficients

#### Information

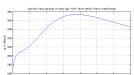


**Modelica.Media.IdealGases.SingleGases.Ge**

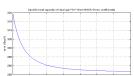
Ideal gas "Ge" from NASA Glenn coefficients

**Information****Modelica.Media.IdealGases.SingleGases.Geplus**

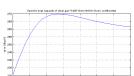
Ideal gas "Ge+" from NASA Glenn coefficients

**Information****Modelica.Media.IdealGases.SingleGases.Geminus**

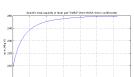
Ideal gas "Ge-" from NASA Glenn coefficients

**Information****Modelica.Media.IdealGases.SingleGases.GeBr**

Ideal gas "GeBr" from NASA Glenn coefficients

**Information****Modelica.Media.IdealGases.SingleGases.GeBr2**

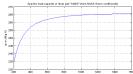
Ideal gas "GeBr2" from NASA Glenn coefficients

**Information**

### Modelica.Media.IdealGases.SingleGases.GeBr3

Ideal gas "GeBr3" from NASA Glenn coefficients

#### Information

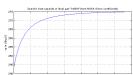


---

### Modelica.Media.IdealGases.SingleGases.GeBr4

Ideal gas "GeBr4" from NASA Glenn coefficients

#### Information

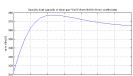


---

### Modelica.Media.IdealGases.SingleGases.GeCL

Ideal gas "GeCl" from NASA Glenn coefficients

#### Information

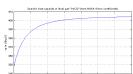


---

### Modelica.Media.IdealGases.SingleGases.GeCL2

Ideal gas "GeCl2" from NASA Glenn coefficients

#### Information

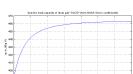


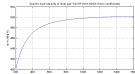
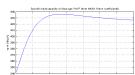
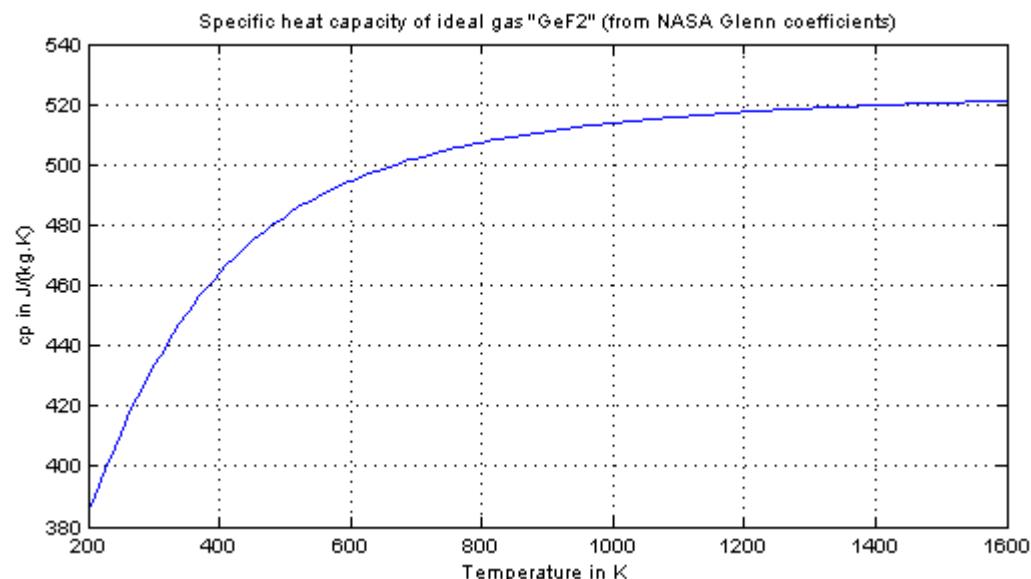
---

### Modelica.Media.IdealGases.SingleGases.GeCL3

Ideal gas "GeCl3" from NASA Glenn coefficients

#### Information

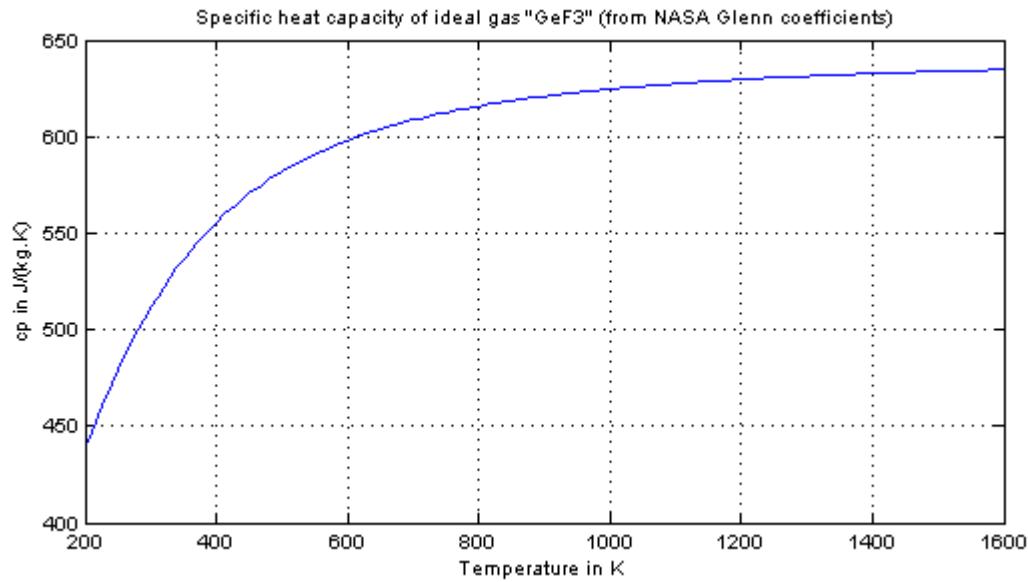


**Modelica.Media.IdealGases.SingleGases.GeCl4****Ideal gas "GeCl4" from NASA Glenn coefficients****Information****Modelica.Media.IdealGases.SingleGases.GeF****Ideal gas "GeF" from NASA Glenn coefficients****Information****Modelica.Media.IdealGases.SingleGases.GeF2****Ideal gas "GeF2" from NASA Glenn coefficients****Information****Modelica.Media.IdealGases.SingleGases.GeF3****Ideal gas "GeF3" from NASA Glenn coefficients**

## 1414 Modelica.Media.IdealGases.SingleGases.GeF3

---

### Information

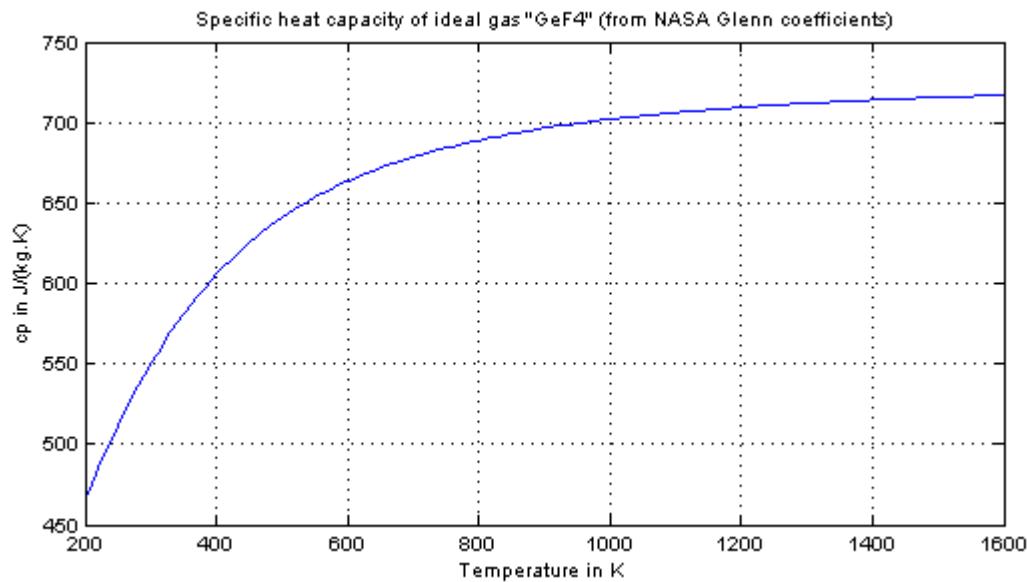


---

## Modelica.Media.IdealGases.SingleGases.GeF4

Ideal gas "GeF4" from NASA Glenn coefficients

### Information

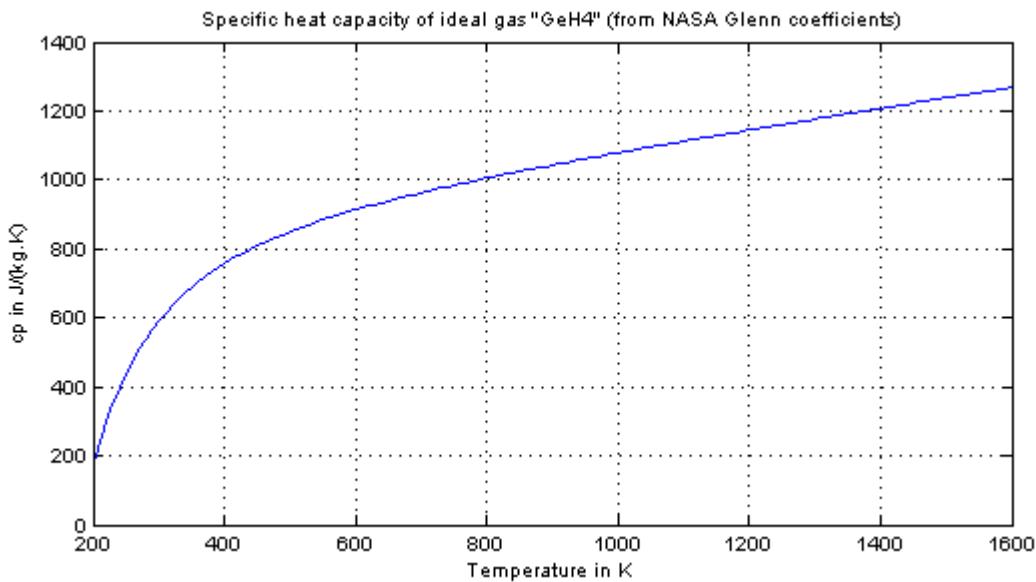


---

## Modelica.Media.IdealGases.SingleGases.GeH4

Ideal gas "GeH4" from NASA Glenn coefficients

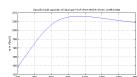
## Information



## Modelica.Media.IdealGases.SingleGases.Gel

Ideal gas "Gel" from NASA Glenn coefficients

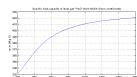
## Information



## Modelica.Media.IdealGases.SingleGases.GeO

Ideal gas "GeO" from NASA Glenn coefficients

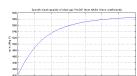
## Information



## Modelica.Media.IdealGases.SingleGases.GeO2

Ideal gas "GeO2" from NASA Glenn coefficients

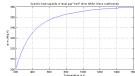
## Information



**Modelica.Media.IdealGases.SingleGases.GeS**

Ideal gas "GeS" from NASA Glenn coefficients

**Information**

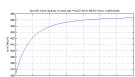


---

**Modelica.Media.IdealGases.SingleGases.GeS2**

Ideal gas "GeS2" from NASA Glenn coefficients

**Information**

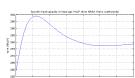


---

**Modelica.Media.IdealGases.SingleGases.Ge2**

Ideal gas "Ge2" from NASA Glenn coefficients

**Information**

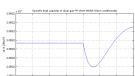


---

**Modelica.Media.IdealGases.SingleGases.H**

Ideal gas "H" from NASA Glenn coefficients

**Information**

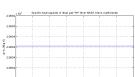


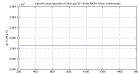
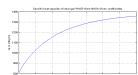
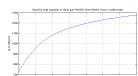
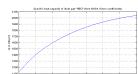
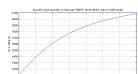
---

**Modelica.Media.IdealGases.SingleGases.Hplus**

Ideal gas "H+" from NASA Glenn coefficients

**Information**



**Modelica.Media.IdealGases.SingleGases.Hminus****Ideal gas "H-" from NASA Glenn coefficients****Information****Modelica.Media.IdealGases.SingleGases.HALO****Ideal gas "HALO" from NASA Glenn coefficients****Information****Modelica.Media.IdealGases.SingleGases.HALO2****Ideal gas "HALO2" from NASA Glenn coefficients****Information****Modelica.Media.IdealGases.SingleGases.HBO****Ideal gas "HBO" from NASA Glenn coefficients****Information****Modelica.Media.IdealGases.SingleGases.HBOplus****Ideal gas "HBO+" from NASA Glenn coefficients****Information**

---

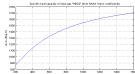
## 1418 Modelica.Media.IdealGases.SingleGases.HBO2

---

### Modelica.Media.IdealGases.SingleGases.HBO2

Ideal gas "HBO2" from NASA Glenn coefficients

#### Information

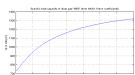


---

### Modelica.Media.IdealGases.SingleGases.HBS

Ideal gas "HBS" from NASA Glenn coefficients

#### Information

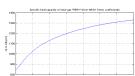


---

### Modelica.Media.IdealGases.SingleGases.HBSplus

Ideal gas "HBS+" from NASA Glenn coefficients

#### Information

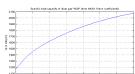


---

### Modelica.Media.IdealGases.SingleGases.HCN

Ideal gas "HCN" from NASA Glenn coefficients

#### Information

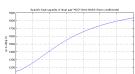


---

### Modelica.Media.IdealGases.SingleGases.HCO

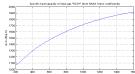
Ideal gas "HCO" from NASA Glenn coefficients

#### Information

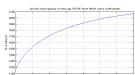


**Modelica.Media.IdealGases.SingleGases.HCOplus**

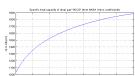
Ideal gas "HCO+" from NASA Glenn coefficients

**Information****Modelica.Media.IdealGases.SingleGases.HCCN**

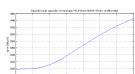
Ideal gas "HCCN" from NASA Glenn coefficients

**Information****Modelica.Media.IdealGases.SingleGases.HCCO**

Ideal gas "HCCO" from NASA Glenn coefficients

**Information****Modelica.Media.IdealGases.SingleGases.HCL**

Ideal gas "HCl" from NASA Glenn coefficients

**Information****Modelica.Media.IdealGases.SingleGases.HD**

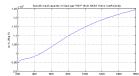
Ideal gas "HD" from NASA Glenn coefficients

**Information**

**Modelica.Media.IdealGases.SingleGases.HDplus**

Ideal gas "HD+" from NASA Glenn coefficients

**Information**

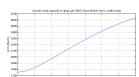


---

**Modelica.Media.IdealGases.SingleGases.HDO**

Ideal gas "HDO" from NASA Glenn coefficients

**Information**

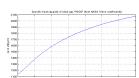


---

**Modelica.Media.IdealGases.SingleGases.HDO2**

Ideal gas "HDO2" from NASA Glenn coefficients

**Information**

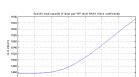


---

**Modelica.Media.IdealGases.SingleGases.HF**

Ideal gas "HF" from NASA Glenn coefficients

**Information**

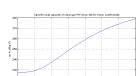


---

**Modelica.Media.IdealGases.SingleGases.HI**

Ideal gas "HI" from NASA Glenn coefficients

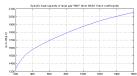
**Information**



## Modelica.Media.IdealGases.SingleGases.HNC

Ideal gas "HNC" from NASA Glenn coefficients

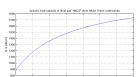
### Information



## Modelica.Media.IdealGases.SingleGases.HNCO

Ideal gas "HNCO" from NASA Glenn coefficients

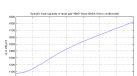
### Information



## Modelica.Media.IdealGases.SingleGases.HNO

Ideal gas "HNO" from NASA Glenn coefficients

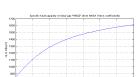
### Information



## Modelica.Media.IdealGases.SingleGases.HNO2

Ideal gas "HNO2" from NASA Glenn coefficients

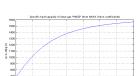
### Information



## Modelica.Media.IdealGases.SingleGases.HNO3

Ideal gas "HNO3" from NASA Glenn coefficients

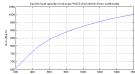
### Information



## Modelica.Media.IdealGases.SingleGases.HOCL

Ideal gas "HOCl" from NASA Glenn coefficients

### Information

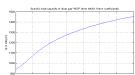


---

## Modelica.Media.IdealGases.SingleGases.HOF

Ideal gas "HOF" from NASA Glenn coefficients

### Information

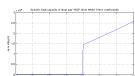


---

## Modelica.Media.IdealGases.SingleGases.HO2

Ideal gas "HO2" from NASA Glenn coefficients

### Information

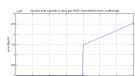


---

## Modelica.Media.IdealGases.SingleGases.HO2minus

Ideal gas "HO2-" from NASA Glenn coefficients

### Information

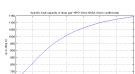


---

## Modelica.Media.IdealGases.SingleGases.HPO

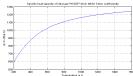
Ideal gas "HPO" from NASA Glenn coefficients

### Information

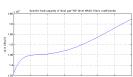


**Modelica.Media.IdealGases.SingleGases.HSO3F**

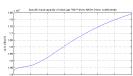
Ideal gas "HSO3F" from NASA Glenn coefficients

**Information****Modelica.Media.IdealGases.SingleGases.H2**

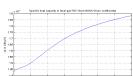
Ideal gas "H2" from NASA Glenn coefficients

**Information****Modelica.Media.IdealGases.SingleGases.H2plus**

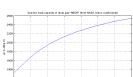
Ideal gas "H2+" from NASA Glenn coefficients

**Information****Modelica.Media.IdealGases.SingleGases.H2minus**

Ideal gas "H2-" from NASA Glenn coefficients

**Information****Modelica.Media.IdealGases.SingleGases.HBOH**

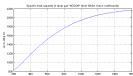
Ideal gas "HBOH" from NASA Glenn coefficients

**Information**

### Modelica.Media.IdealGases.SingleGases.HCOOH

Ideal gas "HCOOH" from NASA Glenn coefficients

#### Information

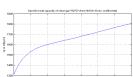


---

### Modelica.Media.IdealGases.SingleGases.H2F2

Ideal gas "H2F2" from NASA Glenn coefficients

#### Information

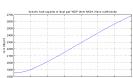


---

### Modelica.Media.IdealGases.SingleGases.H2O

Ideal gas "H2O" from NASA Glenn coefficients

#### Information

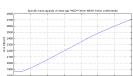


---

### Modelica.Media.IdealGases.SingleGases.H2Oplus

Ideal gas "H2O+" from NASA Glenn coefficients

#### Information

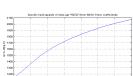


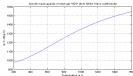
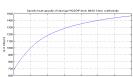
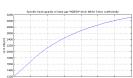
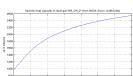
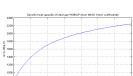
---

### Modelica.Media.IdealGases.SingleGases.H2O2

Ideal gas "H2O2" from NASA Glenn coefficients

#### Information

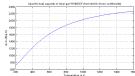


**Modelica.Media.IdealGases.SingleGases.H2S****Ideal gas "H2S" from NASA Glenn coefficients****Information****Modelica.Media.IdealGases.SingleGases.H2SO4****Ideal gas "H2SO4" from NASA Glenn coefficients****Information****Modelica.Media.IdealGases.SingleGases.H2BOH****Ideal gas "H2BOH" from NASA Glenn coefficients****Information****Modelica.Media.IdealGases.SingleGases.HB\_OH\_2****Ideal gas "HB\_OH\_2" from NASA Glenn coefficients****Information****Modelica.Media.IdealGases.SingleGases.H3BO3****Ideal gas "H3BO3" from NASA Glenn coefficients****Information**

### Modelica.Media.IdealGases.SingleGases.H3B3O3

Ideal gas "H3B3O3" from NASA Glenn coefficients

#### Information

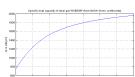


---

### Modelica.Media.IdealGases.SingleGases.H3B3O6

Ideal gas "H3B3O6" from NASA Glenn coefficients

#### Information

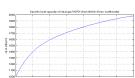


---

### Modelica.Media.IdealGases.SingleGases.H3F3

Ideal gas "H3F3" from NASA Glenn coefficients

#### Information

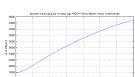


---

### Modelica.Media.IdealGases.SingleGases.H3Oplus

Ideal gas "H3O+" from NASA Glenn coefficients

#### Information

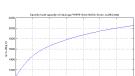


---

### Modelica.Media.IdealGases.SingleGases.H4F4

Ideal gas "H4F4" from NASA Glenn coefficients

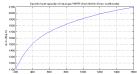
#### Information



## Modelica.Media.IdealGases.SingleGases.H5F5

Ideal gas "H5F5" from NASA Glenn coefficients

### Information

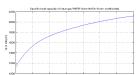


---

## Modelica.Media.IdealGases.SingleGases.H6F6

Ideal gas "H6F6" from NASA Glenn coefficients

### Information

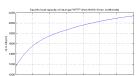


---

## Modelica.Media.IdealGases.SingleGases.H7F7

Ideal gas "H7F7" from NASA Glenn coefficients

### Information

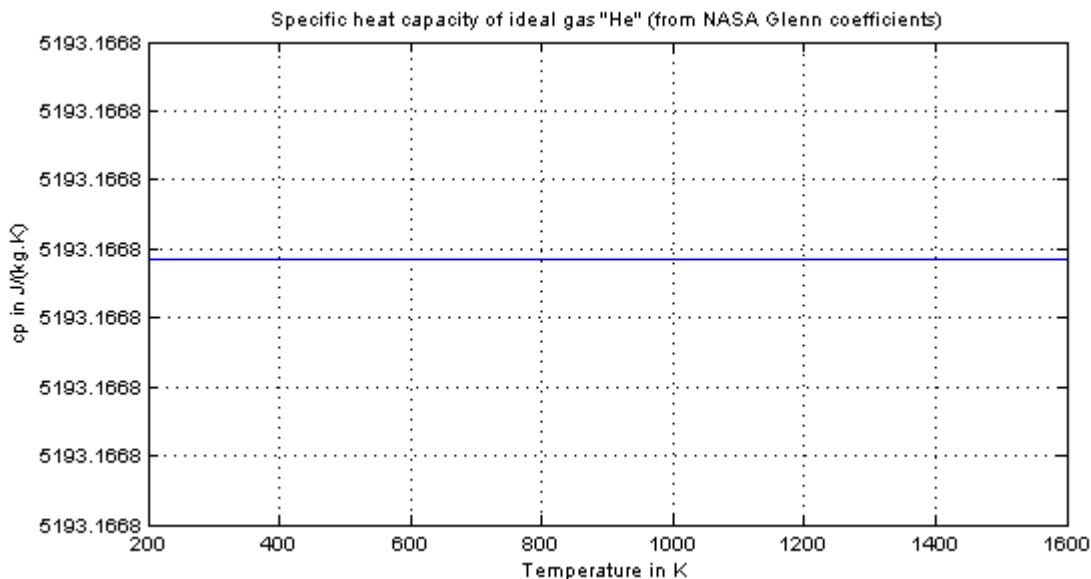


---

## Modelica.Media.IdealGases.SingleGases.He

Ideal gas "He" from NASA Glenn coefficients

## Information

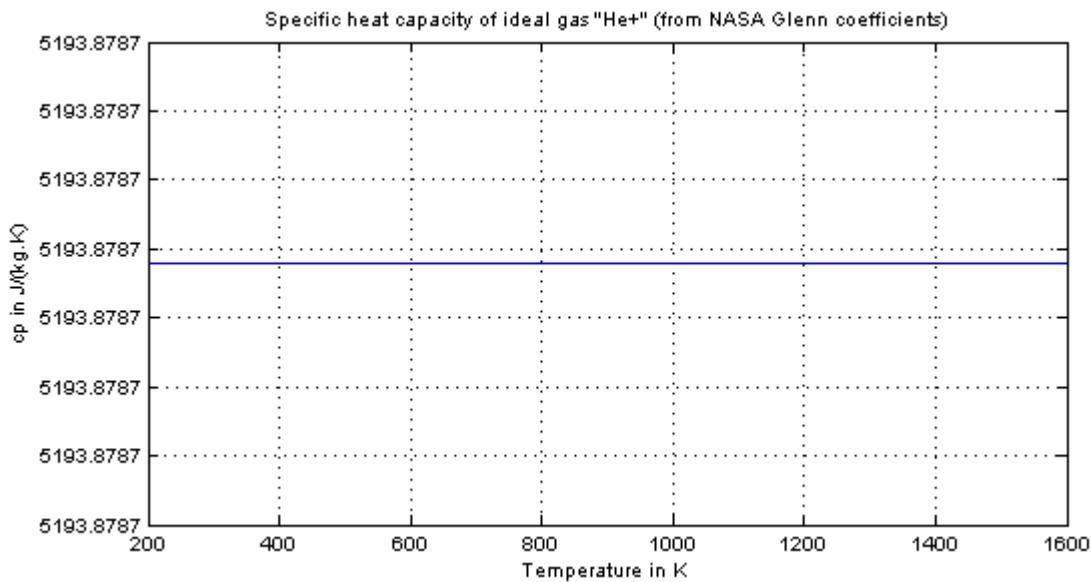


---

## Modelica.Media.IdealGases.SingleGases.Heplus

Ideal gas "He+" from NASA Glenn coefficients

## Information

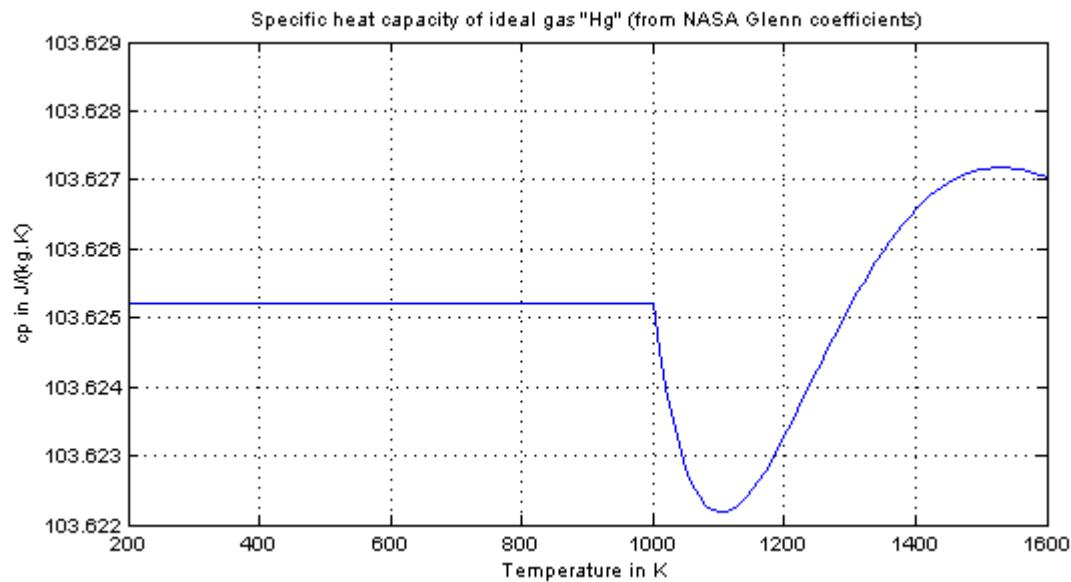


---

## Modelica.Media.IdealGases.SingleGases.Hg

Ideal gas "Hg" from NASA Glenn coefficients

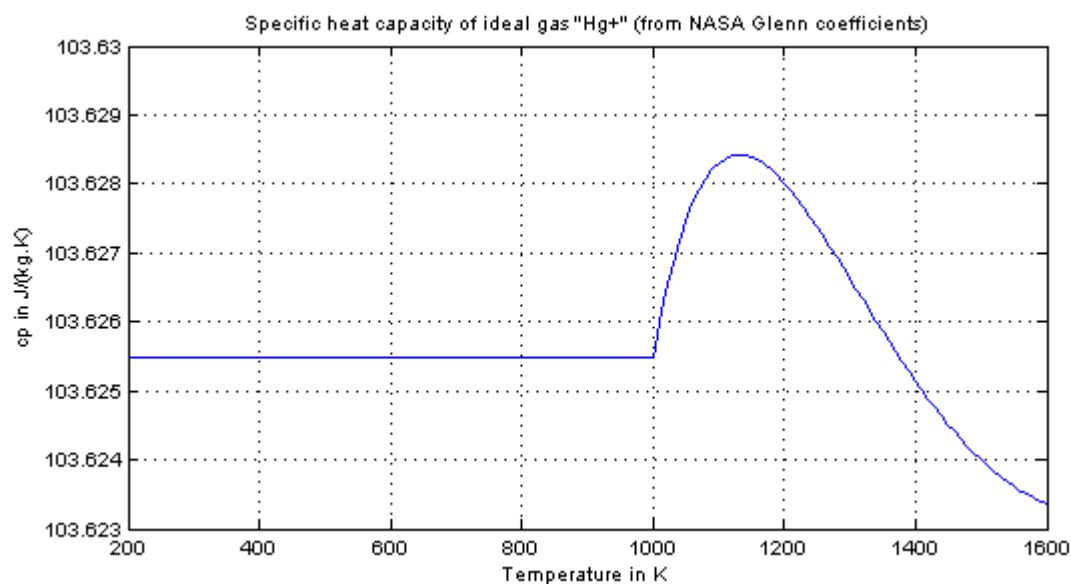
## Information



## Modelica.Media.IdealGases.SingleGases.Hgplus

Ideal gas "Hg+" from NASA Glenn coefficients

## Information



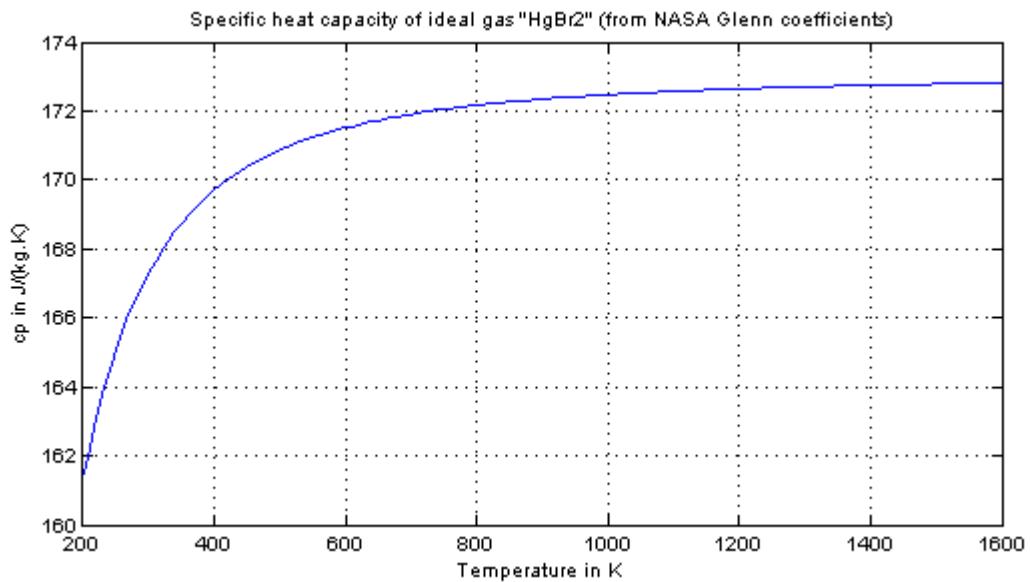
## Modelica.Media.IdealGases.SingleGases.HgBr2

Ideal gas "HgBr2" from NASA Glenn coefficients

## 1430 Modelica.Media.IdealGases.SingleGases.HgBr2

---

### Information

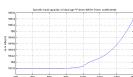


---

## Modelica.Media.IdealGases.SingleGases.I

Ideal gas "I" from NASA Glenn coefficients

### Information

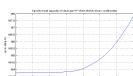


---

## Modelica.Media.IdealGases.SingleGases.Iplus

Ideal gas "I+" from NASA Glenn coefficients

### Information



---

## Modelica.Media.IdealGases.SingleGases.Iminus

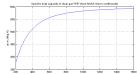
Ideal gas "I-" from NASA Glenn coefficients

### Information

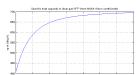


**Modelica.Media.IdealGases.SingleGases.IF5**

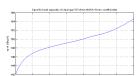
Ideal gas "IF5" from NASA Glenn coefficients

**Information****Modelica.Media.IdealGases.SingleGases.IF7**

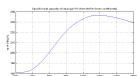
Ideal gas "IF7" from NASA Glenn coefficients

**Information****Modelica.Media.IdealGases.SingleGases.I2**

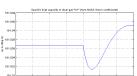
Ideal gas "I2" from NASA Glenn coefficients

**Information****Modelica.Media.IdealGases.SingleGases.In**

Ideal gas "In" from NASA Glenn coefficients

**Information****Modelica.Media.IdealGases.SingleGases.Inplus**

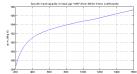
Ideal gas "In+" from NASA Glenn coefficients

**Information**

**Modelica.Media.IdealGases.SingleGases.InBr**

Ideal gas "InBr" from NASA Glenn coefficients

**Information**

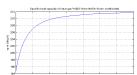


---

**Modelica.Media.IdealGases.SingleGases.InBr2**

Ideal gas "InBr2" from NASA Glenn coefficients

**Information**

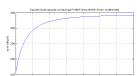


---

**Modelica.Media.IdealGases.SingleGases.InBr3**

Ideal gas "InBr3" from NASA Glenn coefficients

**Information**

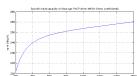


---

**Modelica.Media.IdealGases.SingleGases.InCL**

Ideal gas "InCL" from NASA Glenn coefficients

**Information**

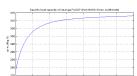


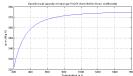
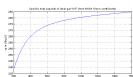
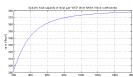
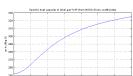
---

**Modelica.Media.IdealGases.SingleGases.InCL2**

Ideal gas "InCL2" from NASA Glenn coefficients

**Information**

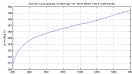


**Modelica.Media.IdealGases.SingleGases.InCL3****Ideal gas "InCL3" from NASA Glenn coefficients****Information****Modelica.Media.IdealGases.SingleGases.InF****Ideal gas "InF" from NASA Glenn coefficients****Information****Modelica.Media.IdealGases.SingleGases.InF2****Ideal gas "InF2" from NASA Glenn coefficients****Information****Modelica.Media.IdealGases.SingleGases.InF3****Ideal gas "InF3" from NASA Glenn coefficients****Information****Modelica.Media.IdealGases.SingleGases.InH****Ideal gas "InH" from NASA Glenn coefficients****Information**

**Modelica.Media.IdealGases.SingleGases.Inl**

Ideal gas "Inl" from NASA Glenn coefficients

**Information**

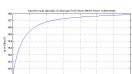


---

**Modelica.Media.IdealGases.SingleGases.Inl2**

Ideal gas "Inl2" from NASA Glenn coefficients

**Information**

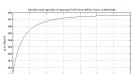


---

**Modelica.Media.IdealGases.SingleGases.Inl3**

Ideal gas "Inl3" from NASA Glenn coefficients

**Information**

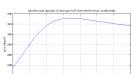


---

**Modelica.Media.IdealGases.SingleGases.InO**

Ideal gas "InO" from NASA Glenn coefficients

**Information**

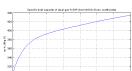


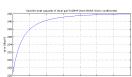
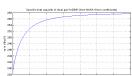
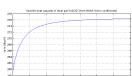
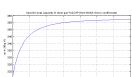
---

**Modelica.Media.IdealGases.SingleGases.InOH**

Ideal gas "InOH" from NASA Glenn coefficients

**Information**

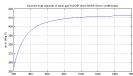


**Modelica.Media.IdealGases.SingleGases.In2Br2****Ideal gas "In2Br2" from NASA Glenn coefficients****Information****Modelica.Media.IdealGases.SingleGases.In2Br4****Ideal gas "In2Br4" from NASA Glenn coefficients****Information****Modelica.Media.IdealGases.SingleGases.In2Br6****Ideal gas "In2Br6" from NASA Glenn coefficients****Information****Modelica.Media.IdealGases.SingleGases.In2Cl2****Ideal gas "In2Cl2" from NASA Glenn coefficients****Information****Modelica.Media.IdealGases.SingleGases.In2Cl4****Ideal gas "In2Cl4" from NASA Glenn coefficients****Information**

**Modelica.Media.IdealGases.SingleGases.In2CL6**

Ideal gas "In2Cl6" from NASA Glenn coefficients

**Information**

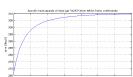


---

**Modelica.Media.IdealGases.SingleGases.In2F2**

Ideal gas "In2F2" from NASA Glenn coefficients

**Information**

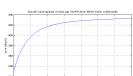


---

**Modelica.Media.IdealGases.SingleGases.In2F4**

Ideal gas "In2F4" from NASA Glenn coefficients

**Information**

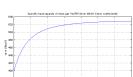


---

**Modelica.Media.IdealGases.SingleGases.In2F6**

Ideal gas "In2F6" from NASA Glenn coefficients

**Information**

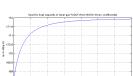


---

**Modelica.Media.IdealGases.SingleGases.In2I2**

Ideal gas "In2I2" from NASA Glenn coefficients

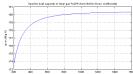
**Information**



## Modelica.Media.IdealGases.SingleGases.In2I4

Ideal gas "In2I4" from NASA Glenn coefficients

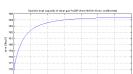
### Information



## Modelica.Media.IdealGases.SingleGases.In2I6

Ideal gas "In2I6" from NASA Glenn coefficients

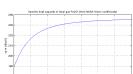
### Information



## Modelica.Media.IdealGases.SingleGases.In2O

Ideal gas "In2O" from NASA Glenn coefficients

### Information



## Modelica.Media.IdealGases.SingleGases.K

Ideal gas "K" from NASA Glenn coefficients

### Information



## Modelica.Media.IdealGases.SingleGases.Kplus

Ideal gas "K+" from NASA Glenn coefficients

### Information



---

## 1438 Modelica.Media.IdealGases.SingleGases.Kminus

---

### Modelica.Media.IdealGases.SingleGases.Kminus

Ideal gas "K-" from NASA Glenn coefficients

#### Information

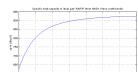


---

### Modelica.Media.IdealGases.SingleGases.KALF4

Ideal gas "KAIF4" from NASA Glenn coefficients

#### Information

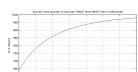


---

### Modelica.Media.IdealGases.SingleGases.KBO2

Ideal gas "KBO2" from NASA Glenn coefficients

#### Information

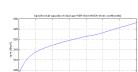


---

### Modelica.Media.IdealGases.SingleGases.KBr

Ideal gas "KBr" from NASA Glenn coefficients

#### Information

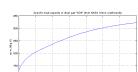


---

### Modelica.Media.IdealGases.SingleGases.KCN

Ideal gas "KCN" from NASA Glenn coefficients

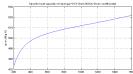
#### Information



## Modelica.Media.IdealGases.SingleGases.KCL

Ideal gas "KCl" from NASA Glenn coefficients

### Information

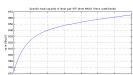


---

## Modelica.Media.IdealGases.SingleGases.KF

Ideal gas "KF" from NASA Glenn coefficients

### Information

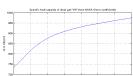


---

## Modelica.Media.IdealGases.SingleGases.KH

Ideal gas "KH" from NASA Glenn coefficients

### Information

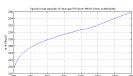


---

## Modelica.Media.IdealGases.SingleGases.KI

Ideal gas "KI" from NASA Glenn coefficients

### Information

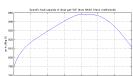


---

## Modelica.Media.IdealGases.SingleGases.KLi

Ideal gas "KLi" from NASA Glenn coefficients

### Information



---

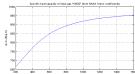
## 1440 Modelica.Media.IdealGases.SingleGases.KNO2

---

### Modelica.Media.IdealGases.SingleGases.KNO2

Ideal gas "KNO2" from NASA Glenn coefficients

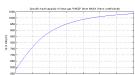
#### Information



### Modelica.Media.IdealGases.SingleGases.KNO3

Ideal gas "KNO3" from NASA Glenn coefficients

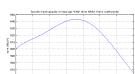
#### Information



### Modelica.Media.IdealGases.SingleGases.KNa

Ideal gas "KNa" from NASA Glenn coefficients

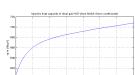
#### Information



### Modelica.Media.IdealGases.SingleGases.KO

Ideal gas "KO" from NASA Glenn coefficients

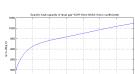
#### Information

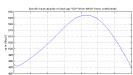
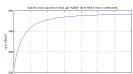
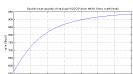
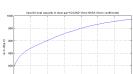


### Modelica.Media.IdealGases.SingleGases.KOH

Ideal gas "KOH" from NASA Glenn coefficients

#### Information

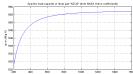


**Modelica.Media.IdealGases.SingleGases.K2****Ideal gas "K2" from NASA Glenn coefficients****Information****Modelica.Media.IdealGases.SingleGases.K2plus****Ideal gas "K2+" from NASA Glenn coefficients****Information****Modelica.Media.IdealGases.SingleGases.K2Br2****Ideal gas "K2Br2" from NASA Glenn coefficients****Information****Modelica.Media.IdealGases.SingleGases.K2CO3****Ideal gas "K2CO3" from NASA Glenn coefficients****Information****Modelica.Media.IdealGases.SingleGases.K2C2N2****Ideal gas "K2C2N2" from NASA Glenn coefficients****Information**

**Modelica.Media.IdealGases.SingleGases.K2CL2**

Ideal gas "K2Cl2" from NASA Glenn coefficients

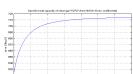
**Information**



**Modelica.Media.IdealGases.SingleGases.K2F2**

Ideal gas "K2F2" from NASA Glenn coefficients

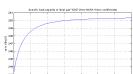
**Information**



**Modelica.Media.IdealGases.SingleGases.K2I2**

Ideal gas "K2I2" from NASA Glenn coefficients

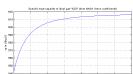
**Information**



**Modelica.Media.IdealGases.SingleGases.K2O**

Ideal gas "K2O" from NASA Glenn coefficients

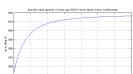
**Information**



**Modelica.Media.IdealGases.SingleGases.K2Oplus**

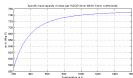
Ideal gas "K2O+" from NASA Glenn coefficients

**Information**

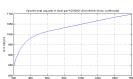


**Modelica.Media.IdealGases.SingleGases.K2O2**

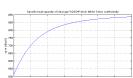
Ideal gas "K2O2" from NASA Glenn coefficients

**Information****Modelica.Media.IdealGases.SingleGases.K2O2H2**

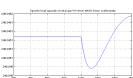
Ideal gas "K2O2H2" from NASA Glenn coefficients

**Information****Modelica.Media.IdealGases.SingleGases.K2SO4**

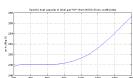
Ideal gas "K2SO4" from NASA Glenn coefficients

**Information****Modelica.Media.IdealGases.SingleGases.Kr**

Ideal gas "Kr" from NASA Glenn coefficients

**Information****Modelica.Media.IdealGases.SingleGases.Krplus**

Ideal gas "Kr+" from NASA Glenn coefficients

**Information**

### Modelica.Media.IdealGases.SingleGases.Li

Ideal gas "li" from NASA Glenn coefficients

#### Information

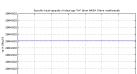


---

### Modelica.Media.IdealGases.SingleGases.Liplus

Ideal gas "li+" from NASA Glenn coefficients

#### Information



---

### Modelica.Media.IdealGases.SingleGases.Liminus

Ideal gas "li-" from NASA Glenn coefficients

#### Information

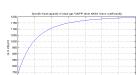


---

### Modelica.Media.IdealGases.SingleGases.LiAlF4

Ideal gas "liAlF4" from NASA Glenn coefficients

#### Information

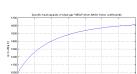


---

### Modelica.Media.IdealGases.SingleGases.LiBO2

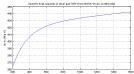
Ideal gas "liBO2" from NASA Glenn coefficients

#### Information

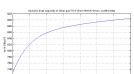


**Modelica.Media.IdealGases.SingleGases.LiBr**

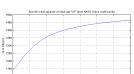
Ideal gas "LiBr" from NASA Glenn coefficients

**Information****Modelica.Media.IdealGases.SingleGases.LiCL**

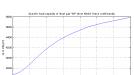
Ideal gas "LiCl" from NASA Glenn coefficients

**Information****Modelica.Media.IdealGases.SingleGases.LiF**

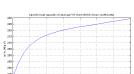
Ideal gas "LiF" from NASA Glenn coefficients

**Information****Modelica.Media.IdealGases.SingleGases.LiH**

Ideal gas "LiH" from NASA Glenn coefficients

**Information****Modelica.Media.IdealGases.SingleGases.LiI**

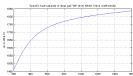
Ideal gas "LiI" from NASA Glenn coefficients

**Information**

**Modelica.Media.IdealGases.SingleGases.LiN**

Ideal gas "LiN" from NASA Glenn coefficients

**Information**

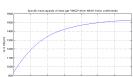


---

**Modelica.Media.IdealGases.SingleGases.LiNO2**

Ideal gas "LiNO2" from NASA Glenn coefficients

**Information**

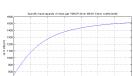


---

**Modelica.Media.IdealGases.SingleGases.LiNO3**

Ideal gas "LiNO3" from NASA Glenn coefficients

**Information**

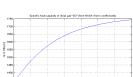


---

**Modelica.Media.IdealGases.SingleGases.LiO**

Ideal gas "LiO" from NASA Glenn coefficients

**Information**

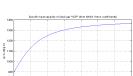


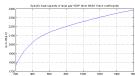
---

**Modelica.Media.IdealGases.SingleGases.LiOF**

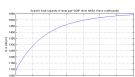
Ideal gas "LiOF" from NASA Glenn coefficients

**Information**

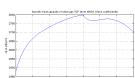


**Modelica.Media.IdealGases.SingleGases.LiOH****Ideal gas "LiOH" from NASA Glenn coefficients****Information**

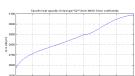
---

**Modelica.Media.IdealGases.SingleGases.LiON****Ideal gas "LiON" from NASA Glenn coefficients****Information**

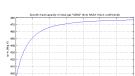
---

**Modelica.Media.IdealGases.SingleGases.Li2****Ideal gas "Li2" from NASA Glenn coefficients****Information**

---

**Modelica.Media.IdealGases.SingleGases.Li2plus****Ideal gas "Li2+" from NASA Glenn coefficients****Information**

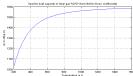
---

**Modelica.Media.IdealGases.SingleGases.Li2Br2****Ideal gas "Li2Br2" from NASA Glenn coefficients****Information**

## Modelica.Media.IdealGases.SingleGases.Li2F2

Ideal gas "li2F2" from NASA Glenn coefficients

### Information

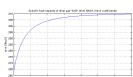


---

## Modelica.Media.IdealGases.SingleGases.Li2I2

Ideal gas "li2I2" from NASA Glenn coefficients

### Information

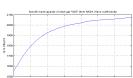


---

## Modelica.Media.IdealGases.SingleGases.Li2O

Ideal gas "li2O" from NASA Glenn coefficients

### Information

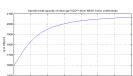


---

## Modelica.Media.IdealGases.SingleGases.Li2Oplus

Ideal gas "li2O+" from NASA Glenn coefficients

### Information

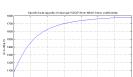


---

## Modelica.Media.IdealGases.SingleGases.Li2O2

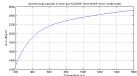
Ideal gas "li2O2" from NASA Glenn coefficients

### Information



**Modelica.Media.IdealGases.SingleGases.Li2O2H2**

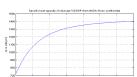
Ideal gas "li2O2H2" from NASA Glenn coefficients

**Information**

---

**Modelica.Media.IdealGases.SingleGases.Li2SO4**

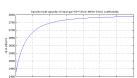
Ideal gas "li2SO4" from NASA Glenn coefficients

**Information**

---

**Modelica.Media.IdealGases.SingleGases.Li3plus**

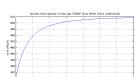
Ideal gas "li3+" from NASA Glenn coefficients

**Information**

---

**Modelica.Media.IdealGases.SingleGases.Li3Br3**

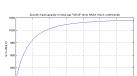
Ideal gas "li3Br3" from NASA Glenn coefficients

**Information**

---

**Modelica.Media.IdealGases.SingleGases.Li3CL3**

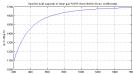
Ideal gas "li3Cl3" from NASA Glenn coefficients

**Information**

### Modelica.Media.IdealGases.SingleGases.Li3F3

Ideal gas "Li3F3" from NASA Glenn coefficients

#### Information

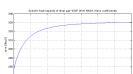


---

### Modelica.Media.IdealGases.SingleGases.Li3I3

Ideal gas "Li3I3" from NASA Glenn coefficients

#### Information

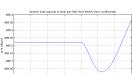


---

### Modelica.Media.IdealGases.SingleGases.Mg

Ideal gas "Mg" from NASA Glenn coefficients

#### Information



---

### Modelica.Media.IdealGases.SingleGases.Mgplus

Ideal gas "Mg+" from NASA Glenn coefficients

#### Information

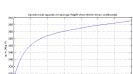


---

### Modelica.Media.IdealGases.SingleGases.MgBr

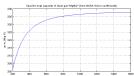
Ideal gas "MgBr" from NASA Glenn coefficients

#### Information

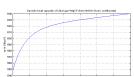


**Modelica.Media.IdealGases.SingleGases.MgBr2**

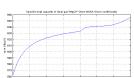
Ideal gas "MgBr2" from NASA Glenn coefficients

**Information****Modelica.Media.IdealGases.SingleGases.MgCL**

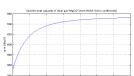
Ideal gas "MgCl" from NASA Glenn coefficients

**Information****Modelica.Media.IdealGases.SingleGases.MgCLplus**

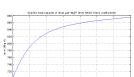
Ideal gas "MgCl+" from NASA Glenn coefficients

**Information****Modelica.Media.IdealGases.SingleGases.MgCL2**

Ideal gas "MgCl2" from NASA Glenn coefficients

**Information****Modelica.Media.IdealGases.SingleGases.MgF**

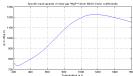
Ideal gas "MgF" from NASA Glenn coefficients

**Information**

### Modelica.Media.IdealGases.SingleGases.MgFplus

Ideal gas "MgF+" from NASA Glenn coefficients

#### Information

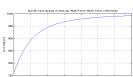


---

### Modelica.Media.IdealGases.SingleGases.MgF2

Ideal gas "MgF2" from NASA Glenn coefficients

#### Information

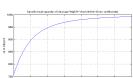


---

### Modelica.Media.IdealGases.SingleGases.MgF2plus

Ideal gas "MgF2+" from NASA Glenn coefficients

#### Information

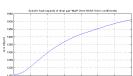


---

### Modelica.Media.IdealGases.SingleGases.MgH

Ideal gas "MgH" from NASA Glenn coefficients

#### Information

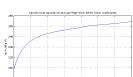


---

### Modelica.Media.IdealGases.SingleGases.Mgl

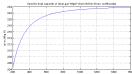
Ideal gas "Mgl" from NASA Glenn coefficients

#### Information

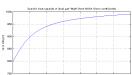


**Modelica.Media.IdealGases.SingleGases.MgI2**

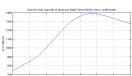
Ideal gas "MgI2" from NASA Glenn coefficients

**Information****Modelica.Media.IdealGases.SingleGases.MgN**

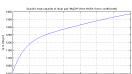
Ideal gas "MgN" from NASA Glenn coefficients

**Information****Modelica.Media.IdealGases.SingleGases.MgO**

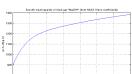
Ideal gas "MgO" from NASA Glenn coefficients

**Information****Modelica.Media.IdealGases.SingleGases.MgOH**

Ideal gas "MgOH" from NASA Glenn coefficients

**Information****Modelica.Media.IdealGases.SingleGases.MgOHplus**

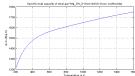
Ideal gas "MgOH+" from NASA Glenn coefficients

**Information**

## Modelica.Media.IdealGases.SingleGases.Mg\_OH\_2

Ideal gas "Mg\_OH\_2" from NASA Glenn coefficients

### Information

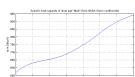


---

## Modelica.Media.IdealGases.SingleGases.MgS

Ideal gas "MgS" from NASA Glenn coefficients

### Information

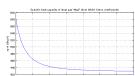


---

## Modelica.Media.IdealGases.SingleGases.Mg2

Ideal gas "Mg2" from NASA Glenn coefficients

### Information

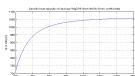


---

## Modelica.Media.IdealGases.SingleGases.Mg2F4

Ideal gas "Mg2F4" from NASA Glenn coefficients

### Information

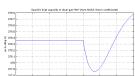


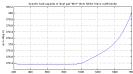
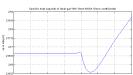
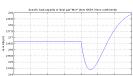
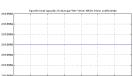
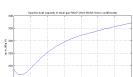
---

## Modelica.Media.IdealGases.SingleGases.Mn

Ideal gas "Mn" from NASA Glenn coefficients

### Information



**Modelica.Media.IdealGases.SingleGases.Mnplus****Ideal gas "Mn+" from NASA Glenn coefficients****Information****Modelica.Media.IdealGases.SingleGases.Mo****Ideal gas "Mo" from NASA Glenn coefficients****Information****Modelica.Media.IdealGases.SingleGases.Moplus****Ideal gas "Mo+" from NASA Glenn coefficients****Information****Modelica.Media.IdealGases.SingleGases.Mominus****Ideal gas "Mo-" from NASA Glenn coefficients****Information****Modelica.Media.IdealGases.SingleGases.MoO****Ideal gas "MoO" from NASA Glenn coefficients****Information**

---

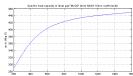
## 1456 Modelica.Media.IdealGases.SingleGases.MoO2

---

### Modelica.Media.IdealGases.SingleGases.MoO2

Ideal gas "MoO2" from NASA Glenn coefficients

#### Information

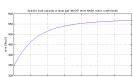


---

### Modelica.Media.IdealGases.SingleGases.MoO3

Ideal gas "MoO3" from NASA Glenn coefficients

#### Information

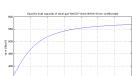


---

### Modelica.Media.IdealGases.SingleGases.MoO3minus

Ideal gas "MoO3-" from NASA Glenn coefficients

#### Information

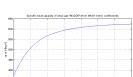


---

### Modelica.Media.IdealGases.SingleGases.Mo2O6

Ideal gas "Mo2O6" from NASA Glenn coefficients

#### Information

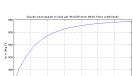


---

### Modelica.Media.IdealGases.SingleGases.Mo3O9

Ideal gas "Mo3O9" from NASA Glenn coefficients

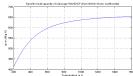
#### Information



## Modelica.Media.IdealGases.SingleGases.Mo4O12

Ideal gas "Mo4O12" from NASA Glenn coefficients

### Information

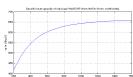


---

## Modelica.Media.IdealGases.SingleGases.Mo5O15

Ideal gas "Mo5O15" from NASA Glenn coefficients

### Information

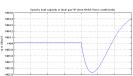


---

## Modelica.Media.IdealGases.SingleGases.N

Ideal gas "N" from NASA Glenn coefficients

### Information

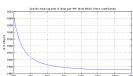


---

## Modelica.Media.IdealGases.SingleGases.Nplus

Ideal gas "N+" from NASA Glenn coefficients

### Information

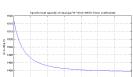


---

## Modelica.Media.IdealGases.SingleGases.Nminus

Ideal gas "N-" from NASA Glenn coefficients

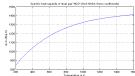
### Information



### Modelica.Media.IdealGases.SingleGases.NCO

Ideal gas "NCO" from NASA Glenn coefficients

#### Information

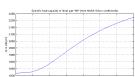


---

### Modelica.Media.IdealGases.SingleGases.ND

Ideal gas "ND" from NASA Glenn coefficients

#### Information

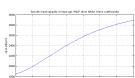


---

### Modelica.Media.IdealGases.SingleGases.ND2

Ideal gas "ND2" from NASA Glenn coefficients

#### Information

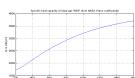


---

### Modelica.Media.IdealGases.SingleGases.ND3

Ideal gas "ND3" from NASA Glenn coefficients

#### Information

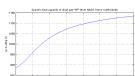


---

### Modelica.Media.IdealGases.SingleGases.NF

Ideal gas "NF" from NASA Glenn coefficients

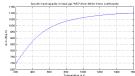
#### Information



## Modelica.Media.IdealGases.SingleGases.NF2

Ideal gas "NF2" from NASA Glenn coefficients

### Information

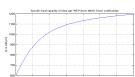


---

## Modelica.Media.IdealGases.SingleGases.NF3

Ideal gas "NF3" from NASA Glenn coefficients

### Information

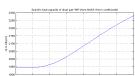


---

## Modelica.Media.IdealGases.SingleGases.NH

Ideal gas "NH" from NASA Glenn coefficients

### Information

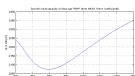


---

## Modelica.Media.IdealGases.SingleGases.NHplus

Ideal gas "NH+" from NASA Glenn coefficients

### Information

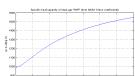


---

## Modelica.Media.IdealGases.SingleGases.NHF

Ideal gas "NHF" from NASA Glenn coefficients

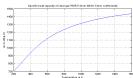
### Information



## Modelica.Media.IdealGases.SingleGases.NHF2

Ideal gas "NHF2" from NASA Glenn coefficients

### Information

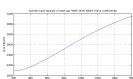


---

## Modelica.Media.IdealGases.SingleGases.NH2

Ideal gas "NH2" from NASA Glenn coefficients

### Information

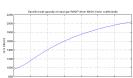


---

## Modelica.Media.IdealGases.SingleGases.NH2F

Ideal gas "NH2F" from NASA Glenn coefficients

### Information

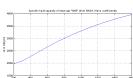


---

## Modelica.Media.IdealGases.SingleGases.NH3

Ideal gas "NH3" from NASA Glenn coefficients

### Information

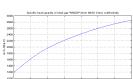


---

## Modelica.Media.IdealGases.SingleGases.NH2OH

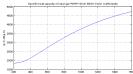
Ideal gas "NH2OH" from NASA Glenn coefficients

### Information

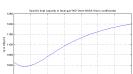


**Modelica.Media.IdealGases.SingleGases.NH4plus**

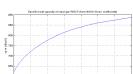
Ideal gas "NH4+" from NASA Glenn coefficients

**Information****Modelica.Media.IdealGases.SingleGases.NO**

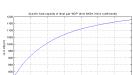
Ideal gas "NO" from NASA Glenn coefficients

**Information****Modelica.Media.IdealGases.SingleGases.NOCL**

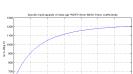
Ideal gas "NOCl" from NASA Glenn coefficients

**Information****Modelica.Media.IdealGases.SingleGases.NOF**

Ideal gas "NOF" from NASA Glenn coefficients

**Information****Modelica.Media.IdealGases.SingleGases.NOF3**

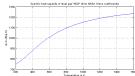
Ideal gas "NOF3" from NASA Glenn coefficients

**Information**

**Modelica.Media.IdealGases.SingleGases.NO2**

Ideal gas "NO2" from NASA Glenn coefficients

**Information**

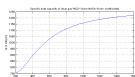


---

**Modelica.Media.IdealGases.SingleGases.NO2minus**

Ideal gas "NO2-" from NASA Glenn coefficients

**Information**

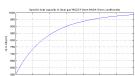


---

**Modelica.Media.IdealGases.SingleGases.NO2CL**

Ideal gas "NO2Cl" from NASA Glenn coefficients

**Information**

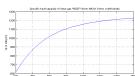


---

**Modelica.Media.IdealGases.SingleGases.NO2F**

Ideal gas "NO2F" from NASA Glenn coefficients

**Information**

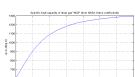


---

**Modelica.Media.IdealGases.SingleGases.NO3**

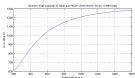
Ideal gas "NO3" from NASA Glenn coefficients

**Information**

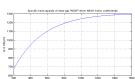


**Modelica.Media.IdealGases.SingleGases.NO3minus**

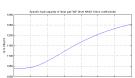
Ideal gas "NO3-" from NASA Glenn coefficients

**Information****Modelica.Media.IdealGases.SingleGases.NO3F**

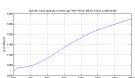
Ideal gas "NO3F" from NASA Glenn coefficients

**Information****Modelica.Media.IdealGases.SingleGases.N2**

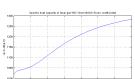
Ideal gas "N2" from NASA Glenn coefficients

**Information****Modelica.Media.IdealGases.SingleGases.N2plus**

Ideal gas "N2+" from NASA Glenn coefficients

**Information****Modelica.Media.IdealGases.SingleGases.N2minus**

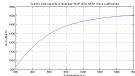
Ideal gas "N2-" from NASA Glenn coefficients

**Information**

**Modelica.Media.IdealGases.SingleGases.NCN**

Ideal gas "NCN" from NASA Glenn coefficients

**Information**

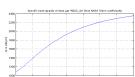


---

**Modelica.Media.IdealGases.SingleGases.N2D2\_cis**

Ideal gas "N2D2\_cis" from NASA Glenn coefficients

**Information**

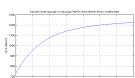


---

**Modelica.Media.IdealGases.SingleGases.N2F2**

Ideal gas "N2F2" from NASA Glenn coefficients

**Information**

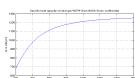


---

**Modelica.Media.IdealGases.SingleGases.N2F4**

Ideal gas "N2F4" from NASA Glenn coefficients

**Information**

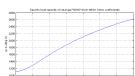


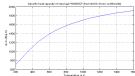
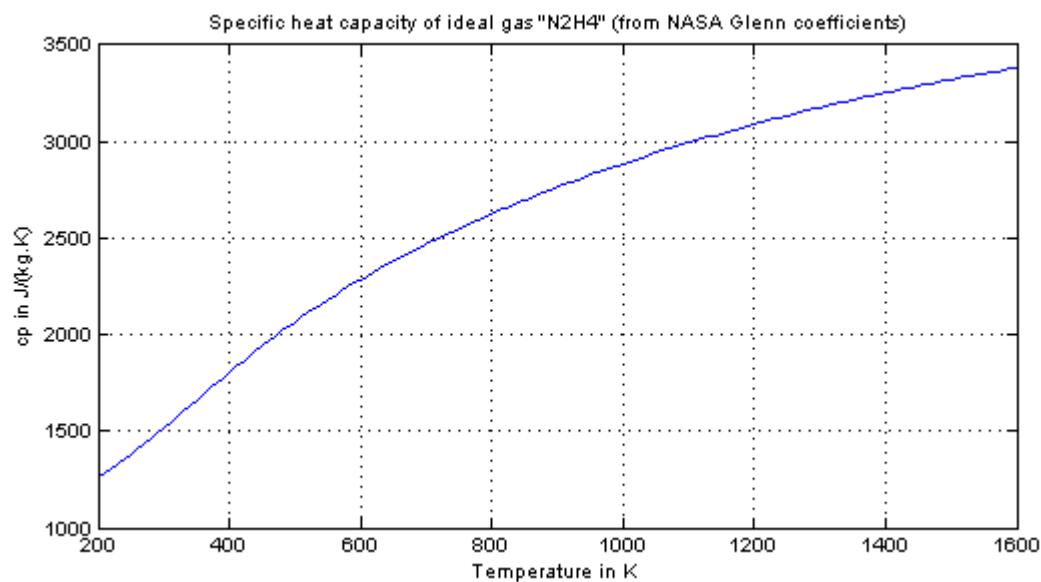
---

**Modelica.Media.IdealGases.SingleGases.N2H2**

Ideal gas "N2H2" from NASA Glenn coefficients

**Information**

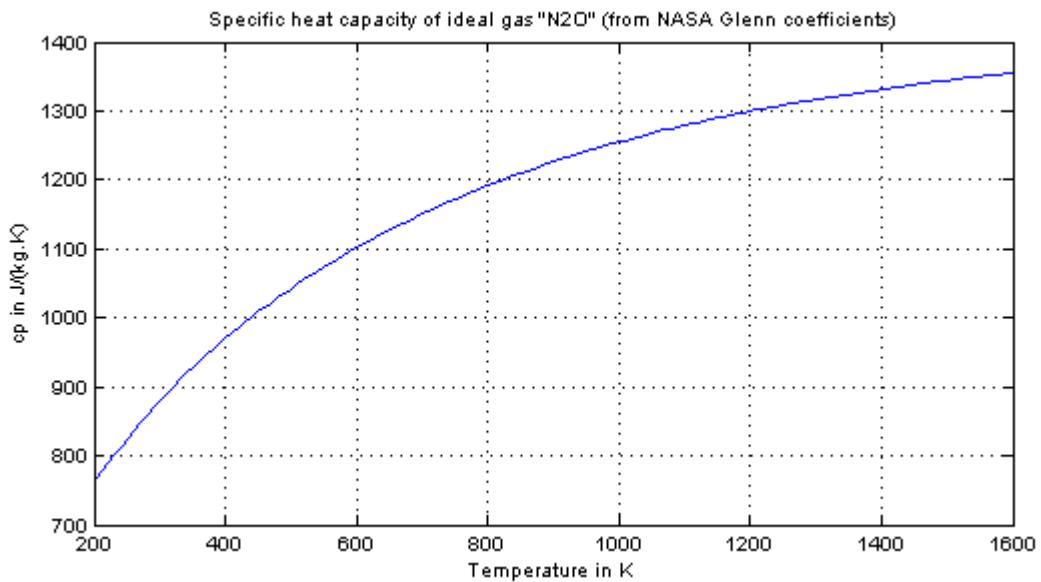


**Modelica.Media.IdealGases.SingleGases.NH2NO2****Ideal gas "NH2NO2" from NASA Glenn coefficients****Information****Modelica.Media.IdealGases.SingleGases.N2H4****Ideal gas "N2H4" from NASA Glenn coefficients****Information****Modelica.Media.IdealGases.SingleGases.N2O****Ideal gas "N2O" from NASA Glenn coefficients**

## 1466 Modelica.Media.IdealGases.SingleGases.N2O

---

### Information

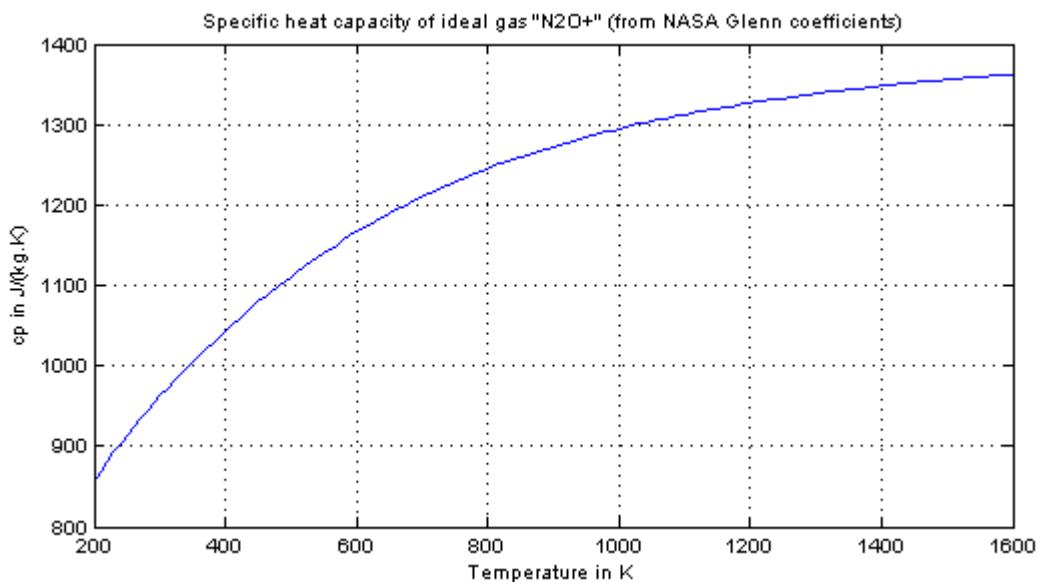


---

## Modelica.Media.IdealGases.SingleGases.N2Oplus

Ideal gas "N2O+" from NASA Glenn coefficients

### Information

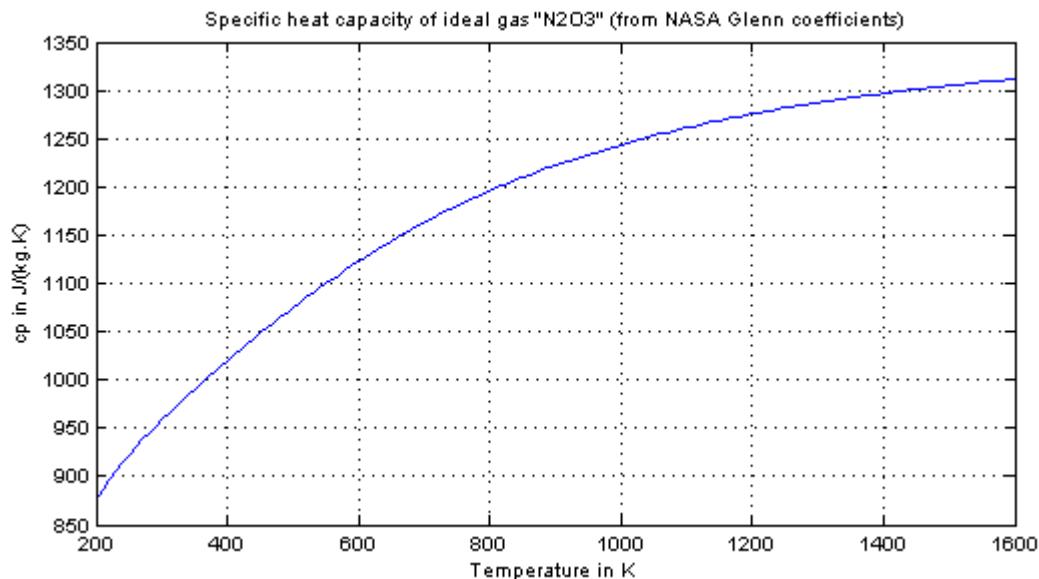


---

## Modelica.Media.IdealGases.SingleGases.N2O3

Ideal gas "N2O3" from NASA Glenn coefficients

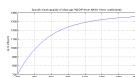
## Information



## Modelica.Media.IdealGases.SingleGases.N2O4

Ideal gas "N2O4" from NASA Glenn coefficients

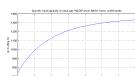
## Information



## Modelica.Media.IdealGases.SingleGases.N2O5

Ideal gas "N2O5" from NASA Glenn coefficients

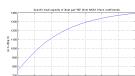
## Information



## Modelica.Media.IdealGases.SingleGases.N3

Ideal gas "N3" from NASA Glenn coefficients

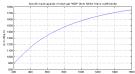
## Information



### Modelica.Media.IdealGases.SingleGases.N3H

Ideal gas "N3H" from NASA Glenn coefficients

#### Information

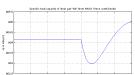


---

### Modelica.Media.IdealGases.SingleGases.Na

Ideal gas "Na" from NASA Glenn coefficients

#### Information



---

### Modelica.Media.IdealGases.SingleGases.Naplus

Ideal gas "Na+" from NASA Glenn coefficients

#### Information



---

### Modelica.Media.IdealGases.SingleGases.Naminus

Ideal gas "Na-" from NASA Glenn coefficients

#### Information

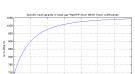


---

### Modelica.Media.IdealGases.SingleGases.NaAlF4

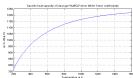
Ideal gas "NaAlF4" from NASA Glenn coefficients

#### Information

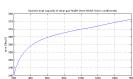


**Modelica.Media.IdealGases.SingleGases.NaBO2**

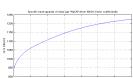
Ideal gas "NaBO2" from NASA Glenn coefficients

**Information****Modelica.Media.IdealGases.SingleGases.NaBr**

Ideal gas "NaBr" from NASA Glenn coefficients

**Information****Modelica.Media.IdealGases.SingleGases.NaCN**

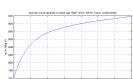
Ideal gas "NaCN" from NASA Glenn coefficients

**Information****Modelica.Media.IdealGases.SingleGases.NaCL**

Ideal gas "NaCl" from NASA Glenn coefficients

**Information****Modelica.Media.IdealGases.SingleGases.NaF**

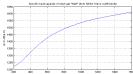
Ideal gas "NaF" from NASA Glenn coefficients

**Information**

### Modelica.Media.IdealGases.SingleGases.NaH

Ideal gas "NaH" from NASA Glenn coefficients

#### Information

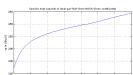


---

### Modelica.Media.IdealGases.SingleGases.NaI

Ideal gas "NaI" from NASA Glenn coefficients

#### Information

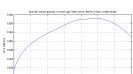


---

### Modelica.Media.IdealGases.SingleGases.NaLi

Ideal gas "NaLi" from NASA Glenn coefficients

#### Information

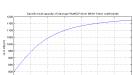


---

### Modelica.Media.IdealGases.SingleGases.NaNO2

Ideal gas "NaNO2" from NASA Glenn coefficients

#### Information

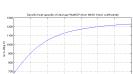


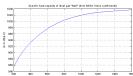
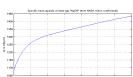
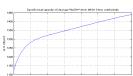
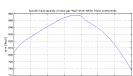
---

### Modelica.Media.IdealGases.SingleGases.NaNO3

Ideal gas "NaNO3" from NASA Glenn coefficients

#### Information

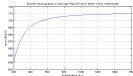


**Modelica.Media.IdealGases.SingleGases.NaO****Ideal gas "NaO" from NASA Glenn coefficients****Information****Modelica.Media.IdealGases.SingleGases.NaOH****Ideal gas "NaOH" from NASA Glenn coefficients****Information****Modelica.Media.IdealGases.SingleGases.NaOHplus****Ideal gas "NaOH+" from NASA Glenn coefficients****Information****Modelica.Media.IdealGases.SingleGases.Na2****Ideal gas "Na2" from NASA Glenn coefficients****Information****Modelica.Media.IdealGases.SingleGases.Na2Br2****Ideal gas "Na2Br2" from NASA Glenn coefficients****Information**

## Modelica.Media.IdealGases.SingleGases.Na2CL2

Ideal gas "Na2Cl2" from NASA Glenn coefficients

### Information

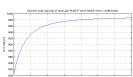


---

## Modelica.Media.IdealGases.SingleGases.Na2F2

Ideal gas "Na2F2" from NASA Glenn coefficients

### Information

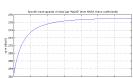


---

## Modelica.Media.IdealGases.SingleGases.Na2I2

Ideal gas "Na2I2" from NASA Glenn coefficients

### Information



---

## Modelica.Media.IdealGases.SingleGases.Na2O

Ideal gas "Na2O" from NASA Glenn coefficients

### Information

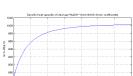


---

## Modelica.Media.IdealGases.SingleGases.Na2Oplus

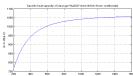
Ideal gas "Na2O+" from NASA Glenn coefficients

### Information

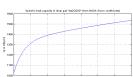


**Modelica.Media.IdealGases.SingleGases.Na2O2**

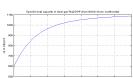
Ideal gas "Na2O2" from NASA Glenn coefficients

**Information****Modelica.Media.IdealGases.SingleGases.Na2O2H2**

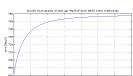
Ideal gas "Na2O2H2" from NASA Glenn coefficients

**Information****Modelica.Media.IdealGases.SingleGases.Na2SO4**

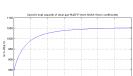
Ideal gas "Na2SO4" from NASA Glenn coefficients

**Information****Modelica.Media.IdealGases.SingleGases.Na3Cl3**

Ideal gas "Na3Cl3" from NASA Glenn coefficients

**Information****Modelica.Media.IdealGases.SingleGases.Na3F3**

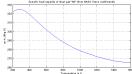
Ideal gas "Na3F3" from NASA Glenn coefficients

**Information**

**Modelica.Media.IdealGases.SingleGases.Nb**

Ideal gas "Nb" from NASA Glenn coefficients

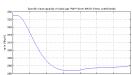
**Information**



**Modelica.Media.IdealGases.SingleGases.Nbplus**

Ideal gas "Nb+" from NASA Glenn coefficients

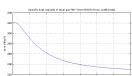
**Information**



**Modelica.Media.IdealGases.SingleGases.Nbminus**

Ideal gas "Nb-" from NASA Glenn coefficients

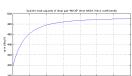
**Information**



**Modelica.Media.IdealGases.SingleGases.NbCL5**

Ideal gas "NbCl5" from NASA Glenn coefficients

**Information**

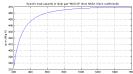
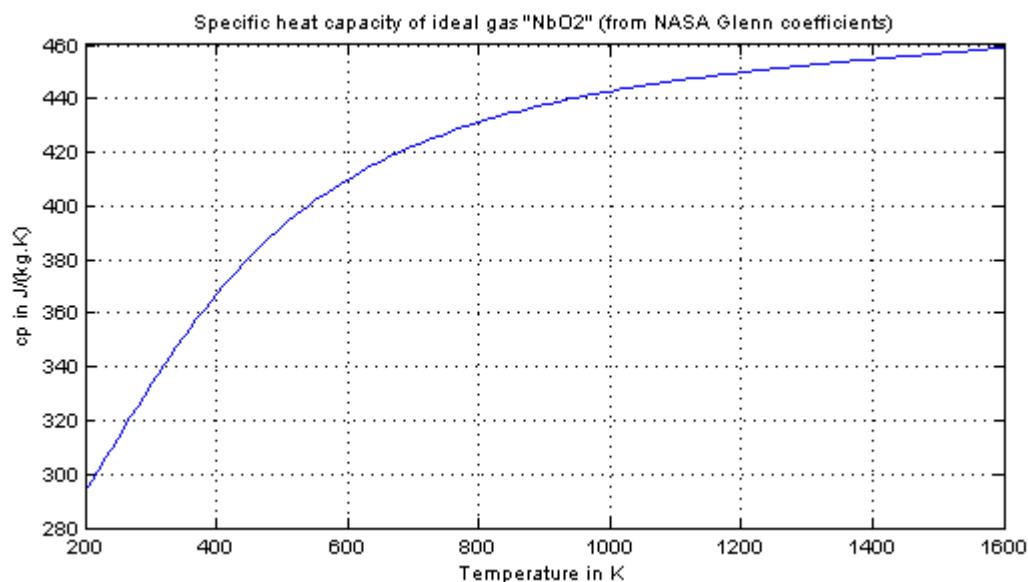


**Modelica.Media.IdealGases.SingleGases.NbO**

Ideal gas "NbO" from NASA Glenn coefficients

**Information**

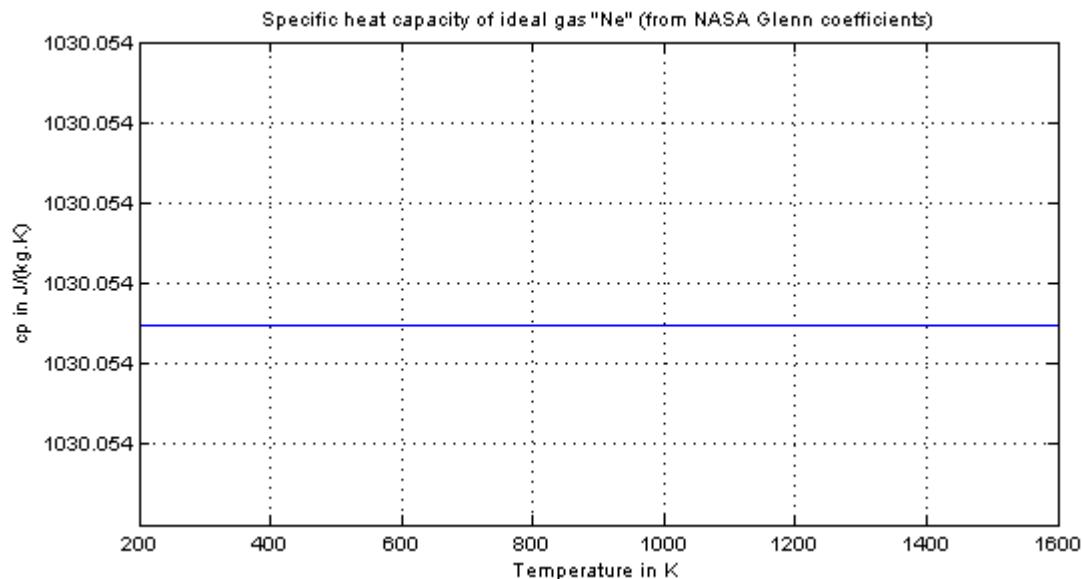


**Modelica.Media.IdealGases.SingleGases.NbOCL3****Ideal gas "NbOCl3" from NASA Glenn coefficients****Information****Modelica.Media.IdealGases.SingleGases.NbO2****Ideal gas "NbO2" from NASA Glenn coefficients****Information****Modelica.Media.IdealGases.SingleGases.Ne****Ideal gas "Ne" from NASA Glenn coefficients**

## 1476 Modelica.Media.IdealGases.SingleGases.Ne

---

### Information

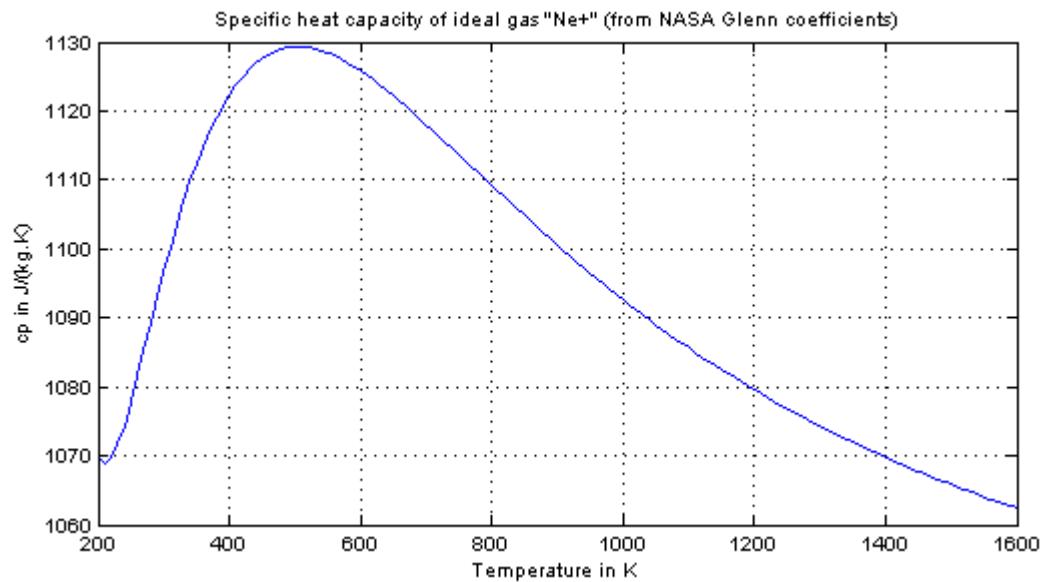


---

## Modelica.Media.IdealGases.SingleGases.Neplus

Ideal gas "Ne+" from NASA Glenn coefficients

### Information

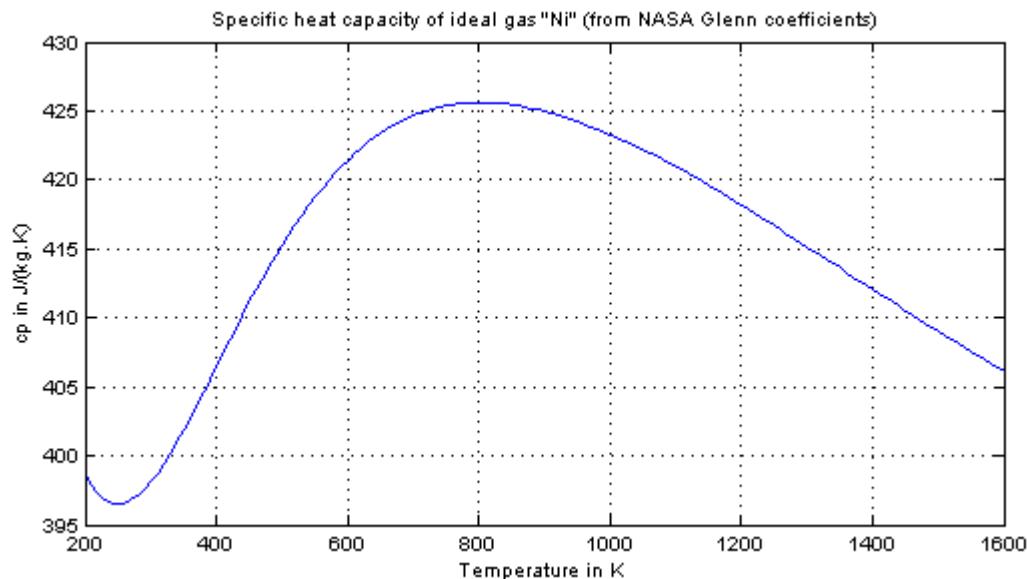


---

## Modelica.Media.IdealGases.SingleGases.Ni

Ideal gas "Ni" from NASA Glenn coefficients

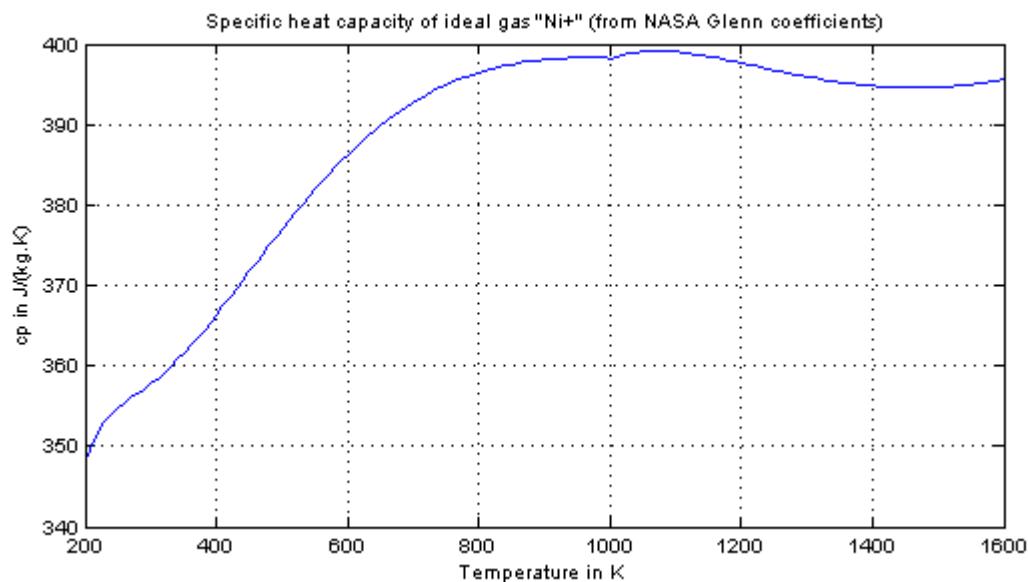
## Information



## Modelica.Media.IdealGases.SingleGases.Niplus

Ideal gas "Ni+" from NASA Glenn coefficients

## Information



## Modelica.Media.IdealGases.SingleGases.Niminus

Ideal gas "Ni-" from NASA Glenn coefficients

---

## 1478 Modelica.Media.IdealGases.SingleGases.Niminus

---

### Information

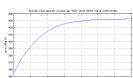


---

## Modelica.Media.IdealGases.SingleGases.NiCL

Ideal gas "NiCl" from NASA Glenn coefficients

### Information

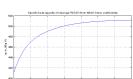


---

## Modelica.Media.IdealGases.SingleGases.NiCL2

Ideal gas "NiCl2" from NASA Glenn coefficients

### Information

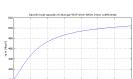


---

## Modelica.Media.IdealGases.SingleGases.NiO

Ideal gas "NiO" from NASA Glenn coefficients

### Information

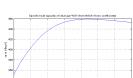


---

## Modelica.Media.IdealGases.SingleGases.NiS

Ideal gas "NiS" from NASA Glenn coefficients

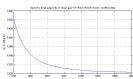
### Information



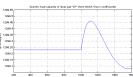
---

## Modelica.Media.IdealGases.SingleGases.O

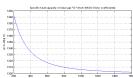
Ideal gas "O" from NASA Glenn coefficients

**Information****Modelica.Media.IdealGases.SingleGases.Oplus**

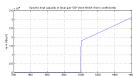
Ideal gas "O+" from NASA Glenn coefficients

**Information****Modelica.Media.IdealGases.SingleGases.Ominus**

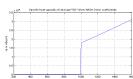
Ideal gas "O-" from NASA Glenn coefficients

**Information****Modelica.Media.IdealGases.SingleGases.OD**

Ideal gas "OD" from NASA Glenn coefficients

**Information****Modelica.Media.IdealGases.SingleGases.ODminus**

Ideal gas "OD-" from NASA Glenn coefficients

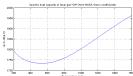
**Information****Modelica.Media.IdealGases.SingleGases.OH**

Ideal gas "OH" from NASA Glenn coefficients

## 1480 Modelica.Media.IdealGases.SingleGases.OH

---

### Information

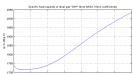


---

## Modelica.Media.IdealGases.SingleGases.OHplus

Ideal gas "OH+" from NASA Glenn coefficients

### Information

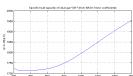


---

## Modelica.Media.IdealGases.SingleGases.OHminus

Ideal gas "OH-" from NASA Glenn coefficients

### Information

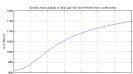


---

## Modelica.Media.IdealGases.SingleGases.O2

Ideal gas "O2" from NASA Glenn coefficients

### Information

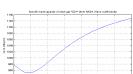


---

## Modelica.Media.IdealGases.SingleGases.O2plus

Ideal gas "O2+" from NASA Glenn coefficients

### Information

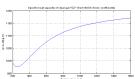


---

## Modelica.Media.IdealGases.SingleGases.O2minus

Ideal gas "O2-" from NASA Glenn coefficients

### Information

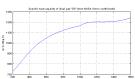


---

### Modelica.Media.IdealGases.SingleGases.O3

Ideal gas "O3" from NASA Glenn coefficients

### Information

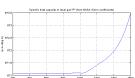


---

### Modelica.Media.IdealGases.SingleGases.P

Ideal gas "P" from NASA Glenn coefficients

### Information

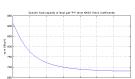


---

### Modelica.Media.IdealGases.SingleGases.Pplus

Ideal gas "P+" from NASA Glenn coefficients

### Information

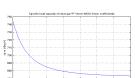


---

### Modelica.Media.IdealGases.SingleGases.Pminus

Ideal gas "P-" from NASA Glenn coefficients

### Information



---

### Modelica.Media.IdealGases.SingleGases.PCL

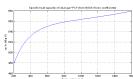
Ideal gas "PCI" from NASA Glenn coefficients

---

## 1482 Modelica.Media.IdealGases.SingleGases.PCL

---

### Information

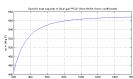


---

## Modelica.Media.IdealGases.SingleGases.PCL2

Ideal gas "PCI2" from NASA Glenn coefficients

### Information

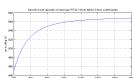


---

## Modelica.Media.IdealGases.SingleGases.PCL2minus

Ideal gas "PCI2-" from NASA Glenn coefficients

### Information

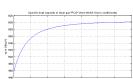


---

## Modelica.Media.IdealGases.SingleGases.PCL3

Ideal gas "PCI3" from NASA Glenn coefficients

### Information

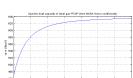


---

## Modelica.Media.IdealGases.SingleGases.PCL5

Ideal gas "PCI5" from NASA Glenn coefficients

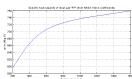
### Information



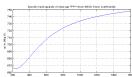
---

## Modelica.Media.IdealGases.SingleGases.PF

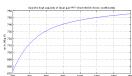
Ideal gas "PF" from NASA Glenn coefficients

**Information****Modelica.Media.IdealGases.SingleGases.PFplus**

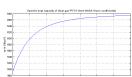
Ideal gas "PF+" from NASA Glenn coefficients

**Information****Modelica.Media.IdealGases.SingleGases.PFminus**

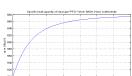
Ideal gas "PF-" from NASA Glenn coefficients

**Information****Modelica.Media.IdealGases.SingleGases.PFCL**

Ideal gas "PFCI" from NASA Glenn coefficients

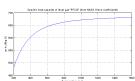
**Information****Modelica.Media.IdealGases.SingleGases.PFCLminus**

Ideal gas "PFCI-" from NASA Glenn coefficients

**Information****Modelica.Media.IdealGases.SingleGases.PFCL2**

Ideal gas "PFCI2" from NASA Glenn coefficients

**Information**

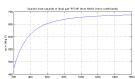


---

**Modelica.Media.IdealGases.SingleGases.PFCL4**

Ideal gas "PFCI4" from NASA Glenn coefficients

**Information**

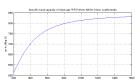


---

**Modelica.Media.IdealGases.SingleGases.PF2**

Ideal gas "PF2" from NASA Glenn coefficients

**Information**

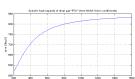


---

**Modelica.Media.IdealGases.SingleGases.PF2minus**

Ideal gas "PF2-‐" from NASA Glenn coefficients

**Information**

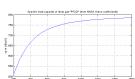


---

**Modelica.Media.IdealGases.SingleGases.PF2CL**

Ideal gas "PF2CI" from NASA Glenn coefficients

**Information**

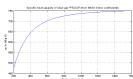


---

**Modelica.Media.IdealGases.SingleGases.PF2CL3**

Ideal gas "PF2CI3" from NASA Glenn coefficients

**Information**

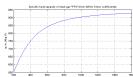


---

**Modelica.Media.IdealGases.SingleGases.PF3**

Ideal gas "PF3" from NASA Glenn coefficients

**Information**

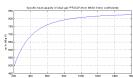


---

**Modelica.Media.IdealGases.SingleGases.PF3CL2**

Ideal gas "PF3Cl2" from NASA Glenn coefficients

**Information**

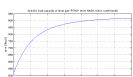


---

**Modelica.Media.IdealGases.SingleGases.PF4CL**

Ideal gas "PF4Cl" from NASA Glenn coefficients

**Information**

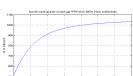


---

**Modelica.Media.IdealGases.SingleGases.PF5**

Ideal gas "PF5" from NASA Glenn coefficients

**Information**

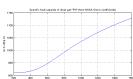


---

**Modelica.Media.IdealGases.SingleGases.PH**

Ideal gas "PH" from NASA Glenn coefficients

**Information**

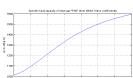


---

**Modelica.Media.IdealGases.SingleGases.PH2**

Ideal gas "PH2" from NASA Glenn coefficients

**Information**

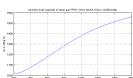


---

**Modelica.Media.IdealGases.SingleGases.PH2minus**

Ideal gas "PH2-" from NASA Glenn coefficients

**Information**

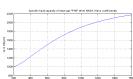


---

**Modelica.Media.IdealGases.SingleGases.PH3**

Ideal gas "PH3" from NASA Glenn coefficients

**Information**



---

**Modelica.Media.IdealGases.SingleGases.PN**

Ideal gas "PN" from NASA Glenn coefficients

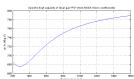
**Information**



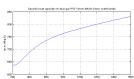
---

**Modelica.Media.IdealGases.SingleGases.PO**

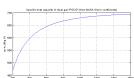
Ideal gas "PO" from NASA Glenn coefficients

**Information****Modelica.Media.IdealGases.SingleGases.POminus**

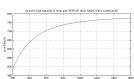
Ideal gas "PO-" from NASA Glenn coefficients

**Information****Modelica.Media.IdealGases.SingleGases.POCL3**

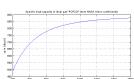
Ideal gas "POCl3" from NASA Glenn coefficients

**Information****Modelica.Media.IdealGases.SingleGases.POFCL2**

Ideal gas "POFCI2" from NASA Glenn coefficients

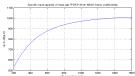
**Information****Modelica.Media.IdealGases.SingleGases.POF2CL**

Ideal gas "POF2CI" from NASA Glenn coefficients

**Information****Modelica.Media.IdealGases.SingleGases.POF3**

Ideal gas "POF3" from NASA Glenn coefficients

**Information**

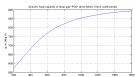


---

**Modelica.Media.IdealGases.SingleGases.PO2**

Ideal gas "PO2" from NASA Glenn coefficients

**Information**

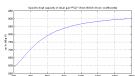


---

**Modelica.Media.IdealGases.SingleGases.PO2minus**

Ideal gas "PO2-" from NASA Glenn coefficients

**Information**

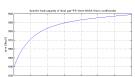


---

**Modelica.Media.IdealGases.SingleGases.PS**

Ideal gas "PS" from NASA Glenn coefficients

**Information**

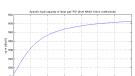


---

**Modelica.Media.IdealGases.SingleGases.P2**

Ideal gas "P2" from NASA Glenn coefficients

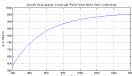
**Information**



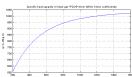
---

**Modelica.Media.IdealGases.SingleGases.P2O3**

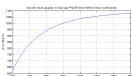
Ideal gas "P2O3" from NASA Glenn coefficients

**Information****Modelica.Media.IdealGases.SingleGases.P2O4**

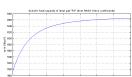
Ideal gas "P2O4" from NASA Glenn coefficients

**Information****Modelica.Media.IdealGases.SingleGases.P2O5**

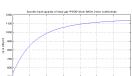
Ideal gas "P2O5" from NASA Glenn coefficients

**Information****Modelica.Media.IdealGases.SingleGases.P3**

Ideal gas "P3" from NASA Glenn coefficients

**Information****Modelica.Media.IdealGases.SingleGases.P3O6**

Ideal gas "P3O6" from NASA Glenn coefficients

**Information****Modelica.Media.IdealGases.SingleGases.P4**

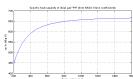
Ideal gas "P4" from NASA Glenn coefficients

---

## 1490 Modelica.Media.IdealGases.SingleGases.P4

---

### Information

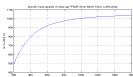


---

## Modelica.Media.IdealGases.SingleGases.P4O6

Ideal gas "P4O6" from NASA Glenn coefficients

### Information

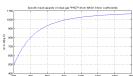


---

## Modelica.Media.IdealGases.SingleGases.P4O7

Ideal gas "P4O7" from NASA Glenn coefficients

### Information

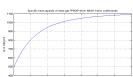


---

## Modelica.Media.IdealGases.SingleGases.P4O8

Ideal gas "P4O8" from NASA Glenn coefficients

### Information

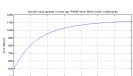


---

## Modelica.Media.IdealGases.SingleGases.P4O9

Ideal gas "P4O9" from NASA Glenn coefficients

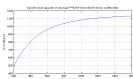
### Information



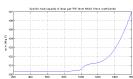
---

## Modelica.Media.IdealGases.SingleGases.P4O10

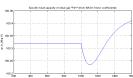
Ideal gas "P4O10" from NASA Glenn coefficients

**Information****Modelica.Media.IdealGases.SingleGases.Pb**

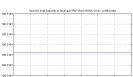
Ideal gas "Pb" from NASA Glenn coefficients

**Information****Modelica.Media.IdealGases.SingleGases.Pbplus**

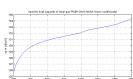
Ideal gas "Pb+" from NASA Glenn coefficients

**Information****Modelica.Media.IdealGases.SingleGases.Pbminus**

Ideal gas "Pb-" from NASA Glenn coefficients

**Information****Modelica.Media.IdealGases.SingleGases.PbBr**

Ideal gas "PbBr" from NASA Glenn coefficients

**Information****Modelica.Media.IdealGases.SingleGases.PbBr2**

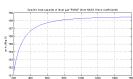
Ideal gas "PbBr2" from NASA Glenn coefficients

---

## 1492 Modelica.Media.IdealGases.SingleGases.PbBr2

---

### Information

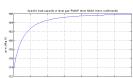


---

## Modelica.Media.IdealGases.SingleGases.PbBr3

Ideal gas "PbBr3" from NASA Glenn coefficients

### Information

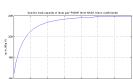


---

## Modelica.Media.IdealGases.SingleGases.PbBr4

Ideal gas "PbBr4" from NASA Glenn coefficients

### Information

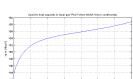


---

## Modelica.Media.IdealGases.SingleGases.PbCl

Ideal gas "PbCl" from NASA Glenn coefficients

### Information



---

## Modelica.Media.IdealGases.SingleGases.PbCl2

Ideal gas "PbCl2" from NASA Glenn coefficients

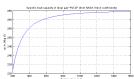
### Information



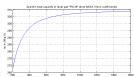
---

## Modelica.Media.IdealGases.SingleGases.PbCl3

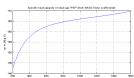
Ideal gas "PbCl3" from NASA Glenn coefficients

**Information****Modelica.Media.IdealGases.SingleGases.PbCl4**

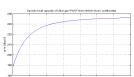
Ideal gas "PbCl4" from NASA Glenn coefficients

**Information****Modelica.Media.IdealGases.SingleGases.PbF**

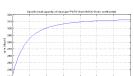
Ideal gas "PbF" from NASA Glenn coefficients

**Information****Modelica.Media.IdealGases.SingleGases.PbF2**

Ideal gas "PbF2" from NASA Glenn coefficients

**Information****Modelica.Media.IdealGases.SingleGases.PbF3**

Ideal gas "PbF3" from NASA Glenn coefficients

**Information****Modelica.Media.IdealGases.SingleGases.PbF4**

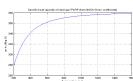
Ideal gas "PbF4" from NASA Glenn coefficients

---

## 1494 Modelica.Media.IdealGases.SingleGases.PbF4

---

### Information

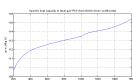


---

## Modelica.Media.IdealGases.SingleGases.PbI

Ideal gas "PbI" from NASA Glenn coefficients

### Information

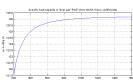


---

## Modelica.Media.IdealGases.SingleGases.PbI2

Ideal gas "PbI2" from NASA Glenn coefficients

### Information



---

## Modelica.Media.IdealGases.SingleGases.PbI3

Ideal gas "PbI3" from NASA Glenn coefficients

### Information

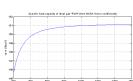


---

## Modelica.Media.IdealGases.SingleGases.PbI4

Ideal gas "PbI4" from NASA Glenn coefficients

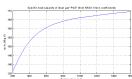
### Information



---

## Modelica.Media.IdealGases.SingleGases.PbO

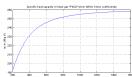
Ideal gas "PbO" from NASA Glenn coefficients

**Information**

---

**Modelica.Media.IdealGases.SingleGases.PbO2**

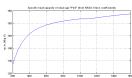
Ideal gas "PbO2" from NASA Glenn coefficients

**Information**

---

**Modelica.Media.IdealGases.SingleGases.PbS**

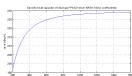
Ideal gas "PbS" from NASA Glenn coefficients

**Information**

---

**Modelica.Media.IdealGases.SingleGases.PbS2**

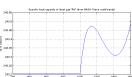
Ideal gas "PbS2" from NASA Glenn coefficients

**Information**

---

**Modelica.Media.IdealGases.SingleGases.Rb**

Ideal gas "Rb" from NASA Glenn coefficients

**Information**

---

**Modelica.Media.IdealGases.SingleGases.Rbplus**

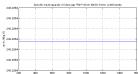
Ideal gas "Rb+" from NASA Glenn coefficients

---

## 1496 Modelica.Media.IdealGases.SingleGases.Rbplus

---

### Information

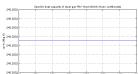


---

## Modelica.Media.IdealGases.SingleGases.Rbminus

Ideal gas "Rb-" from NASA Glenn coefficients

### Information

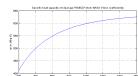


---

## Modelica.Media.IdealGases.SingleGases.RbBO2

Ideal gas "RbBO2" from NASA Glenn coefficients

### Information

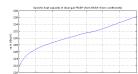


---

## Modelica.Media.IdealGases.SingleGases.RbBr

Ideal gas "RbBr" from NASA Glenn coefficients

### Information

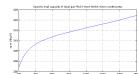


---

## Modelica.Media.IdealGases.SingleGases.RbCl

Ideal gas "RbCl" from NASA Glenn coefficients

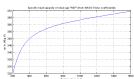
### Information



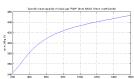
---

## Modelica.Media.IdealGases.SingleGases.RbF

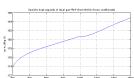
Ideal gas "RbF" from NASA Glenn coefficients

**Information****Modelica.Media.IdealGases.SingleGases.RbH**

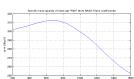
Ideal gas "RbH" from NASA Glenn coefficients

**Information****Modelica.Media.IdealGases.SingleGases.Rbl**

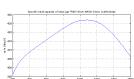
Ideal gas "Rbl" from NASA Glenn coefficients

**Information****Modelica.Media.IdealGases.SingleGases.RbK**

Ideal gas "RbK" from NASA Glenn coefficients

**Information****Modelica.Media.IdealGases.SingleGases.RbLi**

Ideal gas "RbLi" from NASA Glenn coefficients

**Information****Modelica.Media.IdealGases.SingleGases.RbNO2**

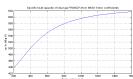
Ideal gas "RbNO2" from NASA Glenn coefficients

---

## 1498 Modelica.Media.IdealGases.SingleGases.RbNO2

---

### Information

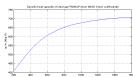


---

## Modelica.Media.IdealGases.SingleGases.RbNO3

Ideal gas "RbNO3" from NASA Glenn coefficients

### Information

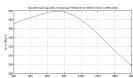


---

## Modelica.Media.IdealGases.SingleGases.RbNa

Ideal gas "RbNa" from NASA Glenn coefficients

### Information

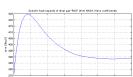


---

## Modelica.Media.IdealGases.SingleGases.RbO

Ideal gas "RbO" from NASA Glenn coefficients

### Information

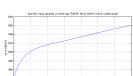


---

## Modelica.Media.IdealGases.SingleGases.RbOH

Ideal gas "RbOH" from NASA Glenn coefficients

### Information

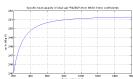


---

## Modelica.Media.IdealGases.SingleGases.Rb2Br2

Ideal gas "Rb2Br2" from NASA Glenn coefficients

**Information**

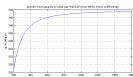


---

**Modelica.Media.IdealGases.SingleGases.Rb2Cl2**

Ideal gas "Rb2Cl2" from NASA Glenn coefficients

**Information**

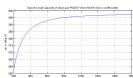


---

**Modelica.Media.IdealGases.SingleGases.Rb2F2**

Ideal gas "Rb2F2" from NASA Glenn coefficients

**Information**

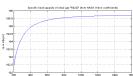


---

**Modelica.Media.IdealGases.SingleGases.Rb2I2**

Ideal gas "Rb2I2" from NASA Glenn coefficients

**Information**

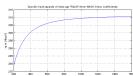


---

**Modelica.Media.IdealGases.SingleGases.Rb2O**

Ideal gas "Rb2O" from NASA Glenn coefficients

**Information**



---

**Modelica.Media.IdealGases.SingleGases.Rb2O2**

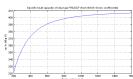
Ideal gas "Rb2O2" from NASA Glenn coefficients

---

## 1500 Modelica.Media.IdealGases.SingleGases.Rb2O2

---

### Information

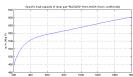


---

## Modelica.Media.IdealGases.SingleGases.Rb2O2H2

Ideal gas "Rb<sub>2</sub>O<sub>2</sub>H<sub>2</sub>" from NASA Glenn coefficients

### Information

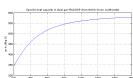


---

## Modelica.Media.IdealGases.SingleGases.Rb2SO4

Ideal gas "Rb<sub>2</sub>SO<sub>4</sub>" from NASA Glenn coefficients

### Information

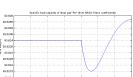


---

## Modelica.Media.IdealGases.SingleGases.Rn

Ideal gas "Rn" from NASA Glenn coefficients

### Information



---

## Modelica.Media.IdealGases.SingleGases.Rnplus

Ideal gas "Rn+" from NASA Glenn coefficients

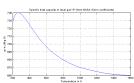
### Information



---

## Modelica.Media.IdealGases.SingleGases.S

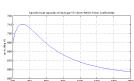
Ideal gas "S" from NASA Glenn coefficients

**Information****Modelica.Media.IdealGases.SingleGases.Splus**

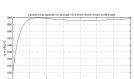
Ideal gas "S+" from NASA Glenn coefficients

**Information****Modelica.Media.IdealGases.SingleGases.Sminus**

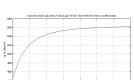
Ideal gas "S-" from NASA Glenn coefficients

**Information****Modelica.Media.IdealGases.SingleGases.SCL**

Ideal gas "SCI" from NASA Glenn coefficients

**Information****Modelica.Media.IdealGases.SingleGases.SCL2**

Ideal gas "SCI2" from NASA Glenn coefficients

**Information****Modelica.Media.IdealGases.SingleGases.SCL2plus**

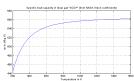
Ideal gas "SCI2+" from NASA Glenn coefficients

---

## 1502 Modelica.Media.IdealGases.SingleGases.SCL2plus

---

### Information

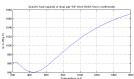


---

## Modelica.Media.IdealGases.SingleGases.SD

Ideal gas "SD" from NASA Glenn coefficients

### Information

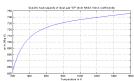


---

## Modelica.Media.IdealGases.SingleGases.SF

Ideal gas "SF" from NASA Glenn coefficients

### Information

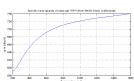


---

## Modelica.Media.IdealGases.SingleGases.SFplus

Ideal gas "SF+" from NASA Glenn coefficients

### Information

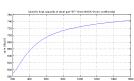


---

## Modelica.Media.IdealGases.SingleGases.SFminus

Ideal gas "SF-" from NASA Glenn coefficients

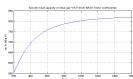
### Information



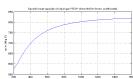
---

## Modelica.Media.IdealGases.SingleGases.SF2

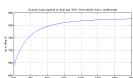
Ideal gas "SF2" from NASA Glenn coefficients

**Information****Modelica.Media.IdealGases.SingleGases.SF2plus**

Ideal gas "SF2+" from NASA Glenn coefficients

**Information****Modelica.Media.IdealGases.SingleGases.SF2minus**

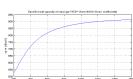
Ideal gas "SF2-" from NASA Glenn coefficients

**Information****Modelica.Media.IdealGases.SingleGases.SF3**

Ideal gas "SF3" from NASA Glenn coefficients

**Information****Modelica.Media.IdealGases.SingleGases.SF3plus**

Ideal gas "SF3+" from NASA Glenn coefficients

**Information****Modelica.Media.IdealGases.SingleGases.SF3minus**

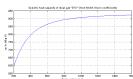
Ideal gas "SF3-" from NASA Glenn coefficients

---

## 1504 Modelica.Media.IdealGases.SingleGases.SF3minus

---

### Information

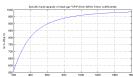


---

## Modelica.Media.IdealGases.SingleGases.SF4

Ideal gas "SF4" from NASA Glenn coefficients

### Information

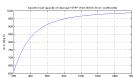


---

## Modelica.Media.IdealGases.SingleGases.SF4plus

Ideal gas "SF4+" from NASA Glenn coefficients

### Information

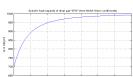


---

## Modelica.Media.IdealGases.SingleGases.SF4minus

Ideal gas "SF4-" from NASA Glenn coefficients

### Information

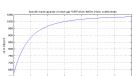


---

## Modelica.Media.IdealGases.SingleGases.SF5

Ideal gas "SF5" from NASA Glenn coefficients

### Information

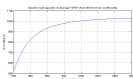


---

## Modelica.Media.IdealGases.SingleGases.SF5plus

Ideal gas "SF5+" from NASA Glenn coefficients

**Information**

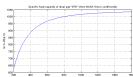


---

**Modelica.Media.IdealGases.SingleGases.SF5minus**

Ideal gas "SF5-" from NASA Glenn coefficients

**Information**

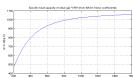


---

**Modelica.Media.IdealGases.SingleGases.SF6**

Ideal gas "SF6" from NASA Glenn coefficients

**Information**



---

**Modelica.Media.IdealGases.SingleGases.SF6minus**

Ideal gas "SF6-" from NASA Glenn coefficients

**Information**

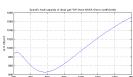


---

**Modelica.Media.IdealGases.SingleGases.SH**

Ideal gas "SH" from NASA Glenn coefficients

**Information**



---

**Modelica.Media.IdealGases.SingleGases.SHminus**

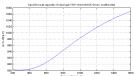
Ideal gas "SH-" from NASA Glenn coefficients

---

## 1506 Modelica.Media.IdealGases.SingleGases.SHminus

---

### Information

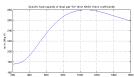


---

## Modelica.Media.IdealGases.SingleGases.SN

Ideal gas "SN" from NASA Glenn coefficients

### Information

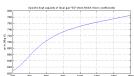


---

## Modelica.Media.IdealGases.SingleGases.SO

Ideal gas "SO" from NASA Glenn coefficients

### Information

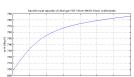


---

## Modelica.Media.IdealGases.SingleGases.SOminus

Ideal gas "SO-" from NASA Glenn coefficients

### Information

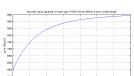


---

## Modelica.Media.IdealGases.SingleGases.SOF2

Ideal gas "SOF2" from NASA Glenn coefficients

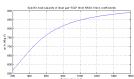
### Information



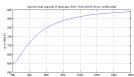
---

## Modelica.Media.IdealGases.SingleGases.SO2

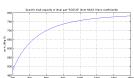
Ideal gas "SO2" from NASA Glenn coefficients

**Information****Modelica.Media.IdealGases.SingleGases.SO2minus**

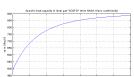
Ideal gas "SO2-" from NASA Glenn coefficients

**Information****Modelica.Media.IdealGases.SingleGases.SO2CL2**

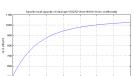
Ideal gas "SO2Cl2" from NASA Glenn coefficients

**Information****Modelica.Media.IdealGases.SingleGases.SO2FCL**

Ideal gas "SO2FCI" from NASA Glenn coefficients

**Information****Modelica.Media.IdealGases.SingleGases.SO2F2**

Ideal gas "SO2F2" from NASA Glenn coefficients

**Information****Modelica.Media.IdealGases.SingleGases.SO3**

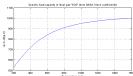
Ideal gas "SO3" from NASA Glenn coefficients

---

## 1508 Modelica.Media.IdealGases.SingleGases.SO3

---

### Information

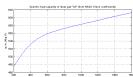


---

## Modelica.Media.IdealGases.SingleGases.S2

Ideal gas "S2" from NASA Glenn coefficients

### Information



---

## Modelica.Media.IdealGases.SingleGases.S2minus

Ideal gas "S2-" from NASA Glenn coefficients

### Information

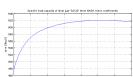


---

## Modelica.Media.IdealGases.SingleGases.S2CL2

Ideal gas "S2Cl2" from NASA Glenn coefficients

### Information

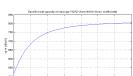


---

## Modelica.Media.IdealGases.SingleGases.S2F2

Ideal gas "S2F2" from NASA Glenn coefficients

### Information



---

## Modelica.Media.IdealGases.SingleGases.S2O

Ideal gas "S2O" from NASA Glenn coefficients

## Information

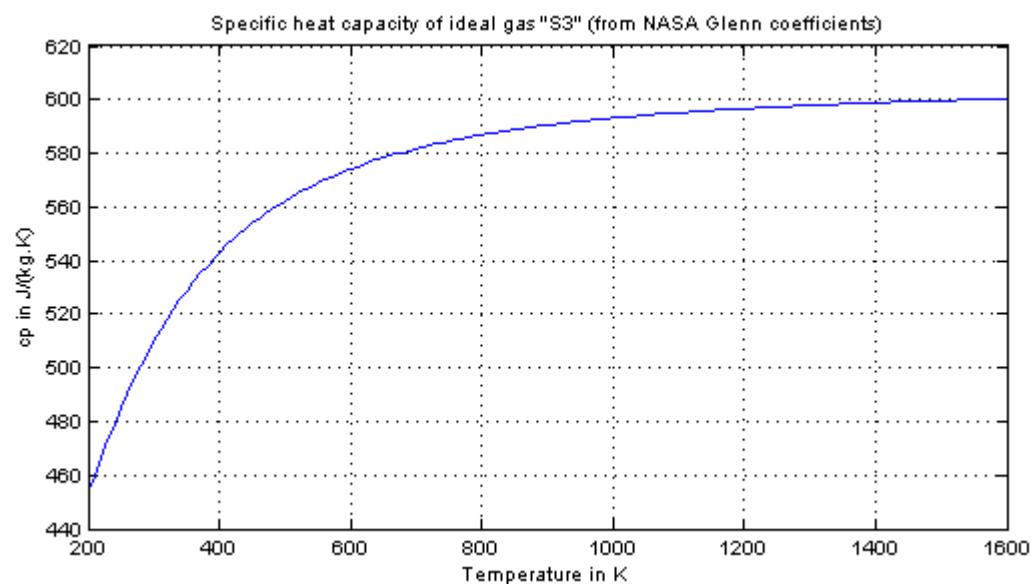


---

### Modelica.Media.IdealGases.SingleGases.S3

Ideal gas "S3" from NASA Glenn coefficients

## Information



---

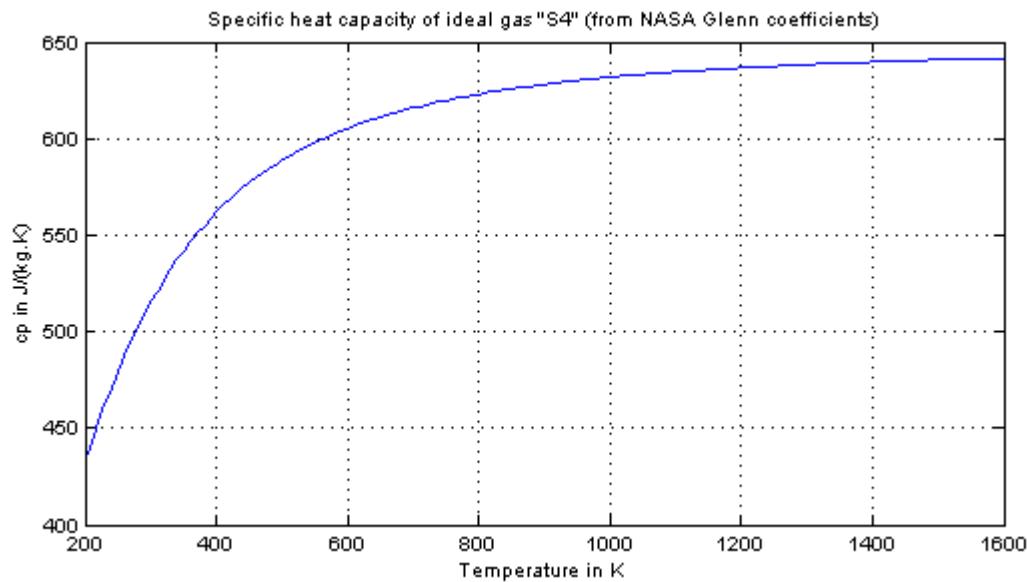
### Modelica.Media.IdealGases.SingleGases.S4

Ideal gas "S4" from NASA Glenn coefficients

## 1510 Modelica.Media.IdealGases.SingleGases.S4

---

### Information

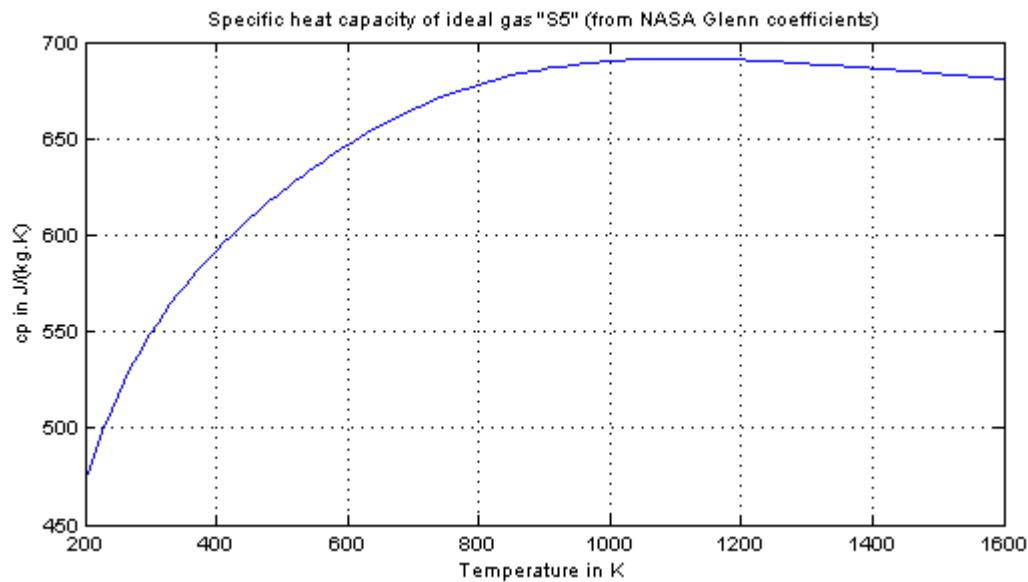


---

## Modelica.Media.IdealGases.SingleGases.S5

### Ideal gas "S5" from NASA Glenn coefficients

### Information

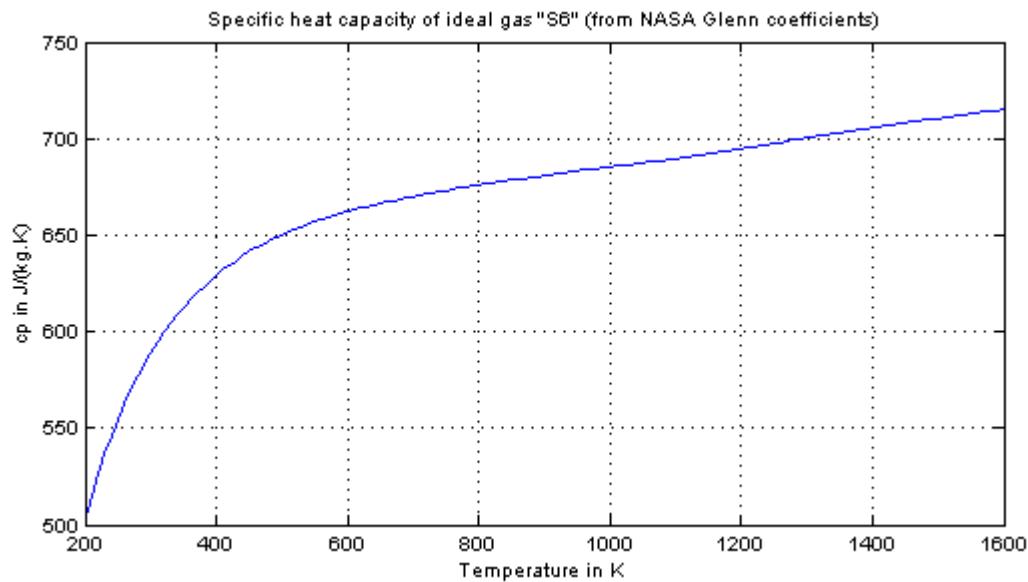


---

## Modelica.Media.IdealGases.SingleGases.S6

### Ideal gas "S6" from NASA Glenn coefficients

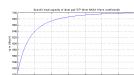
## Information



## Modelica.Media.IdealGases.SingleGases.S7

Ideal gas "S7" from NASA Glenn coefficients

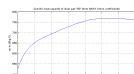
## Information



## Modelica.Media.IdealGases.SingleGases.S8

Ideal gas "S8" from NASA Glenn coefficients

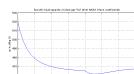
## Information



## Modelica.Media.IdealGases.SingleGases.Sc

Ideal gas "Sc" from NASA Glenn coefficients

## Information



---

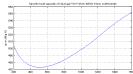
## 1512 Modelica.Media.IdealGases.SingleGases.Scplus

---

### Modelica.Media.IdealGases.SingleGases.Scplus

Ideal gas "Sc+" from NASA Glenn coefficients

#### Information



---

### Modelica.Media.IdealGases.SingleGases.Scminus

Ideal gas "Sc-" from NASA Glenn coefficients

#### Information

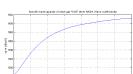


---

### Modelica.Media.IdealGases.SingleGases.ScO

Ideal gas "ScO" from NASA Glenn coefficients

#### Information

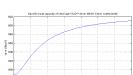


---

### Modelica.Media.IdealGases.SingleGases.ScOplus

Ideal gas "ScO+" from NASA Glenn coefficients

#### Information

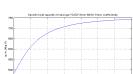


---

### Modelica.Media.IdealGases.SingleGases.ScO2

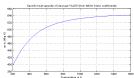
Ideal gas "ScO2" from NASA Glenn coefficients

#### Information

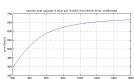


**Modelica.Media.IdealGases.SingleGases.Sc2O**

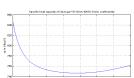
Ideal gas "Sc2O" from NASA Glenn coefficients

**Information****Modelica.Media.IdealGases.SingleGases.Sc2O2**

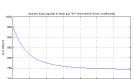
Ideal gas "Sc2O2" from NASA Glenn coefficients

**Information****Modelica.Media.IdealGases.SingleGases.Si**

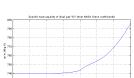
Ideal gas "Si" from NASA Glenn coefficients

**Information****Modelica.Media.IdealGases.SingleGases.Siplus**

Ideal gas "Si+" from NASA Glenn coefficients

**Information****Modelica.Media.IdealGases.SingleGases.Siminus**

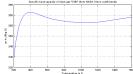
Ideal gas "Si-" from NASA Glenn coefficients

**Information**

### Modelica.Media.IdealGases.SingleGases.SiBr

Ideal gas "SiBr" from NASA Glenn coefficients

#### Information

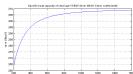


---

### Modelica.Media.IdealGases.SingleGases.SiBr2

Ideal gas "SiBr2" from NASA Glenn coefficients

#### Information

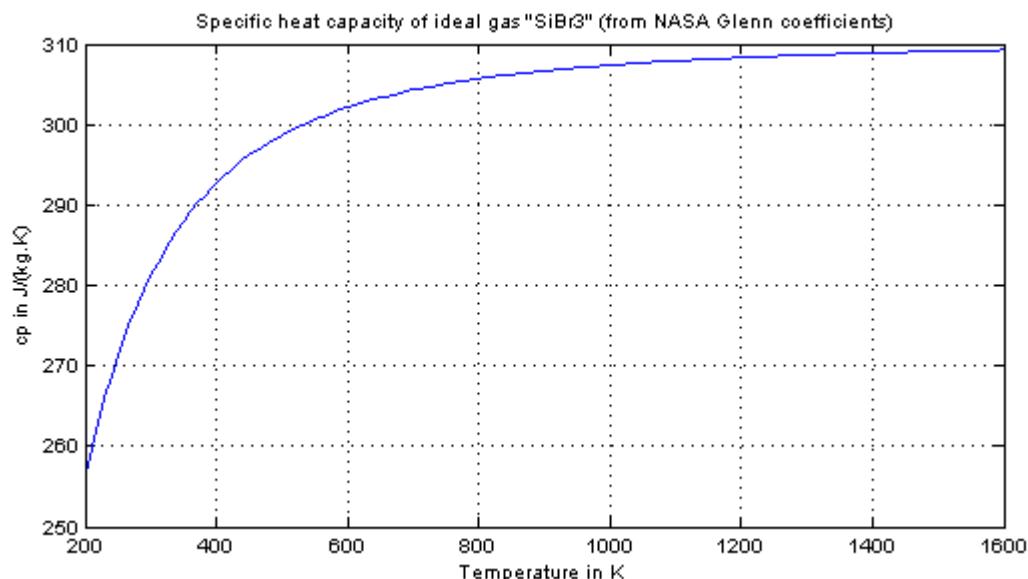


---

### Modelica.Media.IdealGases.SingleGases.SiBr3

Ideal gas "SiBr3" from NASA Glenn coefficients

#### Information

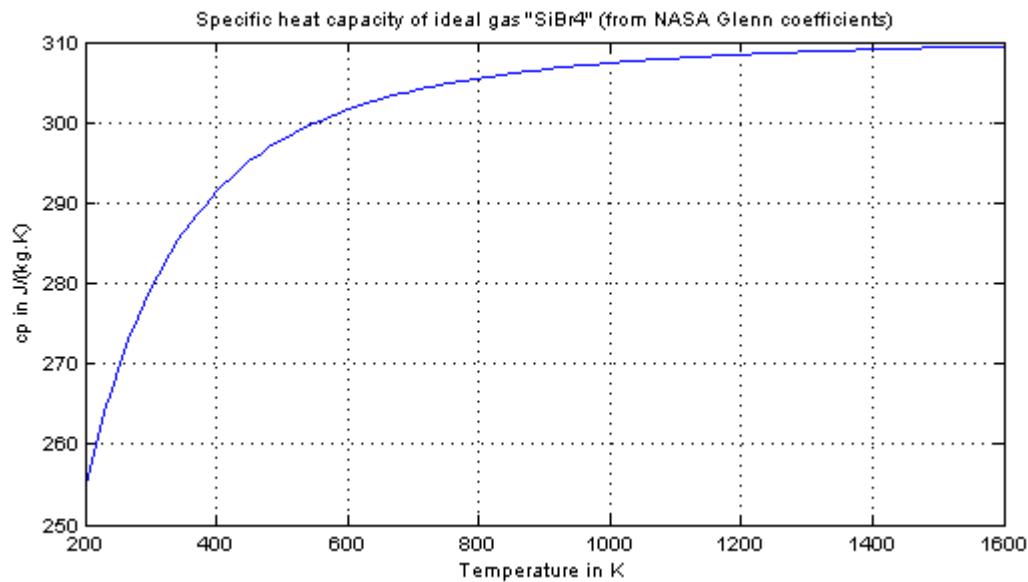


---

### Modelica.Media.IdealGases.SingleGases.SiBr4

Ideal gas "SiBr4" from NASA Glenn coefficients

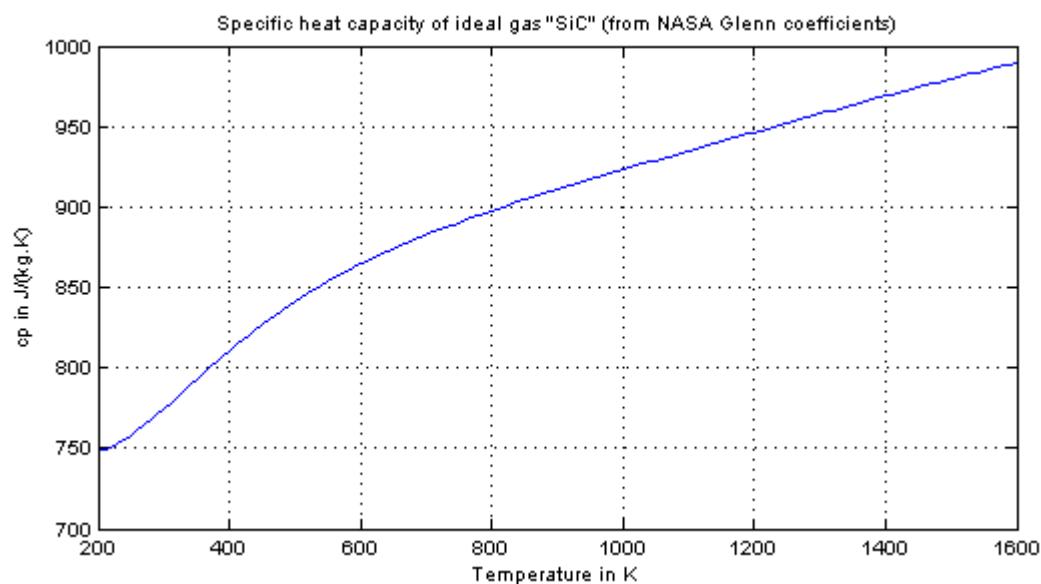
## Information



## Modelica.Media.IdealGases.SingleGases.SiC

Ideal gas "SiC" from NASA Glenn coefficients

## Information



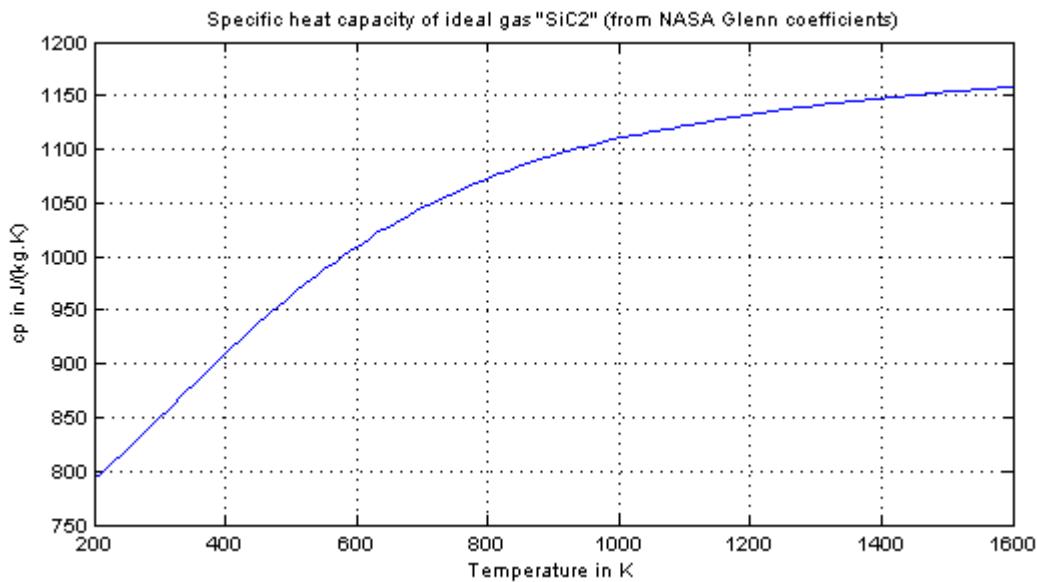
## Modelica.Media.IdealGases.SingleGases.SiC2

Ideal gas "SiC2" from NASA Glenn coefficients

## 1516 Modelica.Media.IdealGases.SingleGases.SiC2

---

### Information

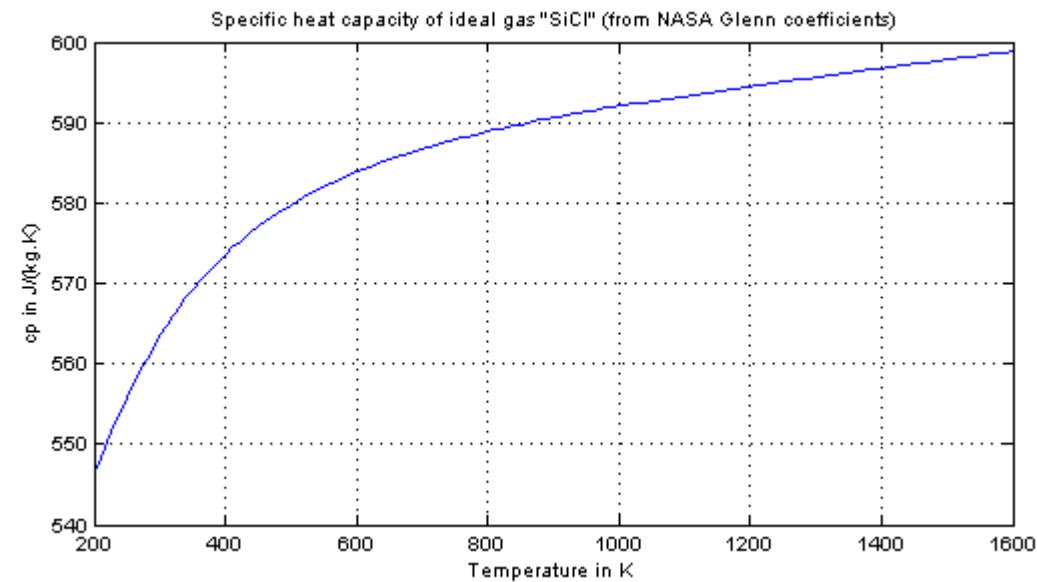


---

## Modelica.Media.IdealGases.SingleGases.SiCL

Ideal gas "SiCl" from NASA Glenn coefficients

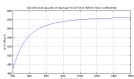
### Information



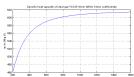
---

## Modelica.Media.IdealGases.SingleGases.SiCL2

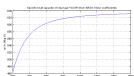
Ideal gas "SiCl2" from NASA Glenn coefficients

**Information****Modelica.Media.IdealGases.SingleGases.SiCL3**

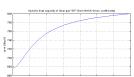
Ideal gas "SiCl3" from NASA Glenn coefficients

**Information****Modelica.Media.IdealGases.SingleGases.SiCL4**

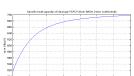
Ideal gas "SiCl4" from NASA Glenn coefficients

**Information****Modelica.Media.IdealGases.SingleGases.SiF**

Ideal gas "SiF" from NASA Glenn coefficients

**Information****Modelica.Media.IdealGases.SingleGases.SiFCL**

Ideal gas "SiFCI" from NASA Glenn coefficients

**Information****Modelica.Media.IdealGases.SingleGases.SiF2**

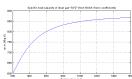
Ideal gas "SiF2" from NASA Glenn coefficients

---

## 1518 Modelica.Media.IdealGases.SingleGases.SiF2

---

### Information

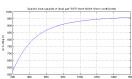


---

## Modelica.Media.IdealGases.SingleGases.SiF3

Ideal gas "SiF3" from NASA Glenn coefficients

### Information

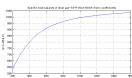


---

## Modelica.Media.IdealGases.SingleGases.SiF4

Ideal gas "SiF4" from NASA Glenn coefficients

### Information

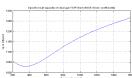


---

## Modelica.Media.IdealGases.SingleGases.SiH

Ideal gas "SiH" from NASA Glenn coefficients

### Information

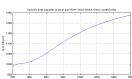


---

## Modelica.Media.IdealGases.SingleGases.SiHplus

Ideal gas "SiH+" from NASA Glenn coefficients

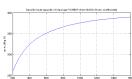
### Information



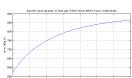
---

## Modelica.Media.IdealGases.SingleGases.SiHBr3

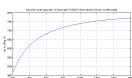
Ideal gas "SiHBr3" from NASA Glenn coefficients

**Information****Modelica.Media.IdealGases.SingleGases.SiHCl**

Ideal gas "SiHCl" from NASA Glenn coefficients

**Information****Modelica.Media.IdealGases.SingleGases.SiHCl3**

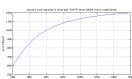
Ideal gas "SiHCl3" from NASA Glenn coefficients

**Information****Modelica.Media.IdealGases.SingleGases.SiHF**

Ideal gas "SiHF" from NASA Glenn coefficients

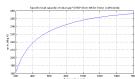
**Information****Modelica.Media.IdealGases.SingleGases.SiHF3**

Ideal gas "SiHF3" from NASA Glenn coefficients

**Information****Modelica.Media.IdealGases.SingleGases.SiHI3**

Ideal gas "SiHI3" from NASA Glenn coefficients

**Information**

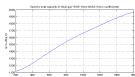


---

**Modelica.Media.IdealGases.SingleGases.SiH2**

Ideal gas "SiH2" from NASA Glenn coefficients

**Information**

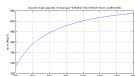


---

**Modelica.Media.IdealGases.SingleGases.SiH2Br2**

Ideal gas "SiH2Br2" from NASA Glenn coefficients

**Information**

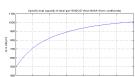


---

**Modelica.Media.IdealGases.SingleGases.SiH2Cl2**

Ideal gas "SiH2Cl2" from NASA Glenn coefficients

**Information**

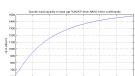


---

**Modelica.Media.IdealGases.SingleGases.SiH2F2**

Ideal gas "SiH2F2" from NASA Glenn coefficients

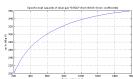
**Information**



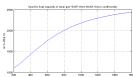
---

**Modelica.Media.IdealGases.SingleGases.SiH2I2**

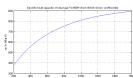
Ideal gas "SiH2I2" from NASA Glenn coefficients

**Information****Modelica.Media.IdealGases.SingleGases.SiH3**

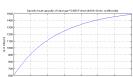
Ideal gas "SiH3" from NASA Glenn coefficients

**Information****Modelica.Media.IdealGases.SingleGases.SiH3Br**

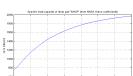
Ideal gas "SiH3Br" from NASA Glenn coefficients

**Information****Modelica.Media.IdealGases.SingleGases.SiH3Cl**

Ideal gas "SiH3Cl" from NASA Glenn coefficients

**Information****Modelica.Media.IdealGases.SingleGases.SiH3F**

Ideal gas "SiH3F" from NASA Glenn coefficients

**Information****Modelica.Media.IdealGases.SingleGases.SiH3I**

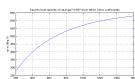
Ideal gas "SiH3I" from NASA Glenn coefficients

---

## 1522 Modelica.Media.IdealGases.SingleGases.SiH3I

---

### Information

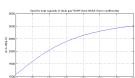


---

## Modelica.Media.IdealGases.SingleGases.SiH4

Ideal gas "SiH4" from NASA Glenn coefficients

### Information

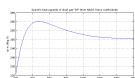


---

## Modelica.Media.IdealGases.SingleGases.SiI

Ideal gas "SiI" from NASA Glenn coefficients

### Information



---

## Modelica.Media.IdealGases.SingleGases.SiI2

Ideal gas "SiI2" from NASA Glenn coefficients

### Information



---

## Modelica.Media.IdealGases.SingleGases.SiN

Ideal gas "SiN" from NASA Glenn coefficients

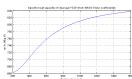
### Information



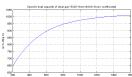
---

## Modelica.Media.IdealGases.SingleGases.SiO

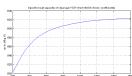
Ideal gas "SiO" from NASA Glenn coefficients

**Information****Modelica.Media.IdealGases.SingleGases.SiO2**

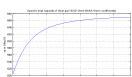
Ideal gas "SiO2" from NASA Glenn coefficients

**Information****Modelica.Media.IdealGases.SingleGases.SiS**

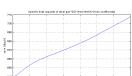
Ideal gas "SiS" from NASA Glenn coefficients

**Information****Modelica.Media.IdealGases.SingleGases.SiS2**

Ideal gas "SiS2" from NASA Glenn coefficients

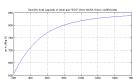
**Information****Modelica.Media.IdealGases.SingleGases.Si2**

Ideal gas "Si2" from NASA Glenn coefficients

**Information****Modelica.Media.IdealGases.SingleGases.Si2C**

Ideal gas "Si2C" from NASA Glenn coefficients

**Information**

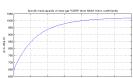


---

**Modelica.Media.IdealGases.SingleGases.Si2F6**

Ideal gas "Si2F6" from NASA Glenn coefficients

**Information**

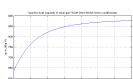


---

**Modelica.Media.IdealGases.SingleGases.Si2N**

Ideal gas "Si2N" from NASA Glenn coefficients

**Information**

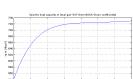


---

**Modelica.Media.IdealGases.SingleGases.Si3**

Ideal gas "Si3" from NASA Glenn coefficients

**Information**

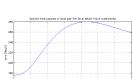


---

**Modelica.Media.IdealGases.SingleGases.Sn**

Ideal gas "Sn" from NASA Glenn coefficients

**Information**

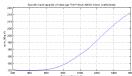


---

**Modelica.Media.IdealGases.SingleGases.Snplus**

Ideal gas "Sn+" from NASA Glenn coefficients

## Information

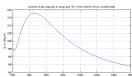


---

## Modelica.Media.IdealGases.SingleGases.Snminus

Ideal gas "Sn-" from NASA Glenn coefficients

## Information

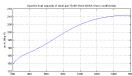


---

## Modelica.Media.IdealGases.SingleGases.SnBr

Ideal gas "SnBr" from NASA Glenn coefficients

## Information

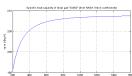


---

## Modelica.Media.IdealGases.SingleGases.SnBr2

Ideal gas "SnBr2" from NASA Glenn coefficients

## Information

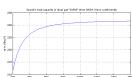


---

## Modelica.Media.IdealGases.SingleGases.SnBr3

Ideal gas "SnBr3" from NASA Glenn coefficients

## Information



---

## Modelica.Media.IdealGases.SingleGases.SnBr4

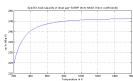
Ideal gas "SnBr4" from NASA Glenn coefficients

---

## 1526 Modelica.Media.IdealGases.SingleGases.SnBr4

---

### Information

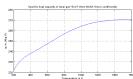


---

## Modelica.Media.IdealGases.SingleGases.SnCL

Ideal gas "SnCl" from NASA Glenn coefficients

### Information

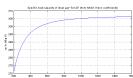


---

## Modelica.Media.IdealGases.SingleGases.SnCL2

Ideal gas "SnCl2" from NASA Glenn coefficients

### Information

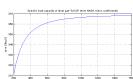


---

## Modelica.Media.IdealGases.SingleGases.SnCL3

Ideal gas "SnCl3" from NASA Glenn coefficients

### Information



---

## Modelica.Media.IdealGases.SingleGases.SnCL4

Ideal gas "SnCl4" from NASA Glenn coefficients

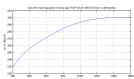
### Information



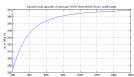
---

## Modelica.Media.IdealGases.SingleGases.SnF

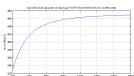
Ideal gas "SnF" from NASA Glenn coefficients

**Information****Modelica.Media.IdealGases.SingleGases.SnF2**

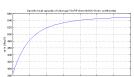
Ideal gas "SnF2" from NASA Glenn coefficients

**Information****Modelica.Media.IdealGases.SingleGases.SnF3**

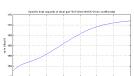
Ideal gas "SnF3" from NASA Glenn coefficients

**Information****Modelica.Media.IdealGases.SingleGases.SnF4**

Ideal gas "SnF4" from NASA Glenn coefficients

**Information****Modelica.Media.IdealGases.SingleGases.SnI**

Ideal gas "SnI" from NASA Glenn coefficients

**Information****Modelica.Media.IdealGases.SingleGases.SnI2**

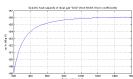
Ideal gas "SnI2" from NASA Glenn coefficients

---

## 1528 Modelica.Media.IdealGases.SingleGases.SnI2

---

### Information

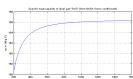


---

## Modelica.Media.IdealGases.SingleGases.SnI3

Ideal gas "SnI3" from NASA Glenn coefficients

### Information

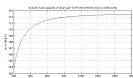


---

## Modelica.Media.IdealGases.SingleGases.SnI4

Ideal gas "SnI4" from NASA Glenn coefficients

### Information

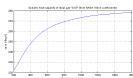


---

## Modelica.Media.IdealGases.SingleGases.SnO

Ideal gas "SnO" from NASA Glenn coefficients

### Information

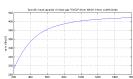


---

## Modelica.Media.IdealGases.SingleGases.SnO2

Ideal gas "SnO2" from NASA Glenn coefficients

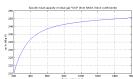
### Information



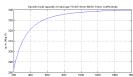
---

## Modelica.Media.IdealGases.SingleGases.SnS

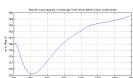
Ideal gas "SnS" from NASA Glenn coefficients

**Information****Modelica.Media.IdealGases.SingleGases.SnS2**

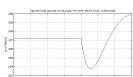
Ideal gas "SnS2" from NASA Glenn coefficients

**Information****Modelica.Media.IdealGases.SingleGases.Sn2**

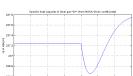
Ideal gas "Sn2" from NASA Glenn coefficients

**Information****Modelica.Media.IdealGases.SingleGases.Sr**

Ideal gas "Sr" from NASA Glenn coefficients

**Information****Modelica.Media.IdealGases.SingleGases.Srplus**

Ideal gas "Sr+" from NASA Glenn coefficients

**Information****Modelica.Media.IdealGases.SingleGases.SrBr**

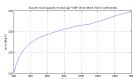
Ideal gas "SrBr" from NASA Glenn coefficients

---

## 1530 Modelica.Media.IdealGases.SingleGases.SrBr

---

### Information

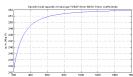


---

## Modelica.Media.IdealGases.SingleGases.SrBr2

Ideal gas "SrBr2" from NASA Glenn coefficients

### Information

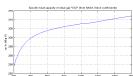


---

## Modelica.Media.IdealGases.SingleGases.SrCl

Ideal gas "SrCl" from NASA Glenn coefficients

### Information

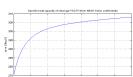


---

## Modelica.Media.IdealGases.SingleGases.SrClplus

Ideal gas "SrCl+" from NASA Glenn coefficients

### Information

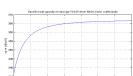


---

## Modelica.Media.IdealGases.SingleGases.SrCl2

Ideal gas "SrCl2" from NASA Glenn coefficients

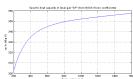
### Information



---

## Modelica.Media.IdealGases.SingleGases.SrF

Ideal gas "SrF" from NASA Glenn coefficients

**Information**

---

**Modelica.Media.IdealGases.SingleGases.SrFplus**

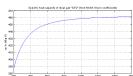
Ideal gas "SrF+" from NASA Glenn coefficients

**Information**

---

**Modelica.Media.IdealGases.SingleGases.SrF2**

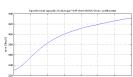
Ideal gas "SrF2" from NASA Glenn coefficients

**Information**

---

**Modelica.Media.IdealGases.SingleGases.SrH**

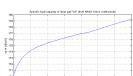
Ideal gas "SrH" from NASA Glenn coefficients

**Information**

---

**Modelica.Media.IdealGases.SingleGases.Srl**

Ideal gas "Srl" from NASA Glenn coefficients

**Information**

---

**Modelica.Media.IdealGases.SingleGases.Srl2**

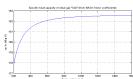
Ideal gas "Srl2" from NASA Glenn coefficients

---

## 1532 Modelica.Media.IdealGases.SingleGases.SrI2

---

### Information

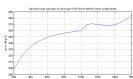


---

## Modelica.Media.IdealGases.SingleGases.SrO

Ideal gas "SrO" from NASA Glenn coefficients

### Information

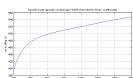


---

## Modelica.Media.IdealGases.SingleGases.SrOH

Ideal gas "SrOH" from NASA Glenn coefficients

### Information

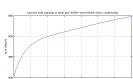


---

## Modelica.Media.IdealGases.SingleGases.SrOHplus

Ideal gas "SrOH+" from NASA Glenn coefficients

### Information

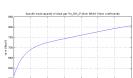


---

## Modelica.Media.IdealGases.SingleGases.Sr\_OH\_2

Ideal gas "Sr\_OH\_2" from NASA Glenn coefficients

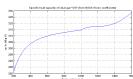
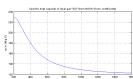
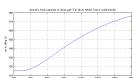
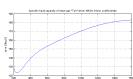
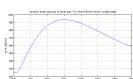
### Information



---

## Modelica.Media.IdealGases.SingleGases.SrS

Ideal gas "SrS" from NASA Glenn coefficients

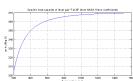
**Information****Modelica.Media.IdealGases.SingleGases.Sr2****Ideal gas "Sr2" from NASA Glenn coefficients****Information****Modelica.Media.IdealGases.SingleGases.Ta****Ideal gas "Ta" from NASA Glenn coefficients****Information****Modelica.Media.IdealGases.SingleGases.Taplus****Ideal gas "Ta+" from NASA Glenn coefficients****Information****Modelica.Media.IdealGases.SingleGases.Taminus****Ideal gas "Ta-" from NASA Glenn coefficients****Information****Modelica.Media.IdealGases.SingleGases.TaCL5****Ideal gas "TaCl5" from NASA Glenn coefficients**

---

## 1534 Modelica.Media.IdealGases.SingleGases.TaCL5

---

### Information

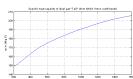


---

## Modelica.Media.IdealGases.SingleGases.TaO

Ideal gas "TaO" from NASA Glenn coefficients

### Information

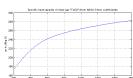


---

## Modelica.Media.IdealGases.SingleGases.TaO2

Ideal gas "TaO2" from NASA Glenn coefficients

### Information

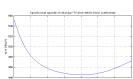


---

## Modelica.Media.IdealGases.SingleGases.Ti

Ideal gas "Ti" from NASA Glenn coefficients

### Information

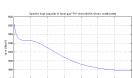


---

## Modelica.Media.IdealGases.SingleGases.Tiplus

Ideal gas "Ti+" from NASA Glenn coefficients

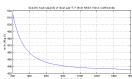
### Information



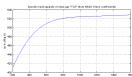
---

## Modelica.Media.IdealGases.SingleGases.Timinus

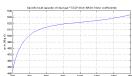
Ideal gas "Ti-" from NASA Glenn coefficients

**Information****Modelica.Media.IdealGases.SingleGases.TiCL**

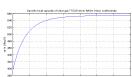
Ideal gas "TiCl" from NASA Glenn coefficients

**Information****Modelica.Media.IdealGases.SingleGases.TiCL2**

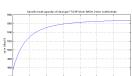
Ideal gas "TiCl2" from NASA Glenn coefficients

**Information****Modelica.Media.IdealGases.SingleGases.TiCL3**

Ideal gas "TiCl3" from NASA Glenn coefficients

**Information****Modelica.Media.IdealGases.SingleGases.TiCL4**

Ideal gas "TiCl4" from NASA Glenn coefficients

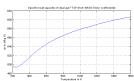
**Information****Modelica.Media.IdealGases.SingleGases.TiO**

Ideal gas "TiO" from NASA Glenn coefficients

## 1536 Modelica.Media.IdealGases.SingleGases.TiO

---

### Information

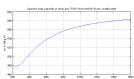


---

## Modelica.Media.IdealGases.SingleGases.TiOplus

Ideal gas "TiO+" from NASA Glenn coefficients

### Information

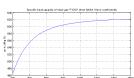


---

## Modelica.Media.IdealGases.SingleGases.TiOCL

Ideal gas "TiOCl" from NASA Glenn coefficients

### Information

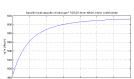


---

## Modelica.Media.IdealGases.SingleGases.TiOCL2

Ideal gas "TiOCl2" from NASA Glenn coefficients

### Information

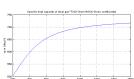


---

## Modelica.Media.IdealGases.SingleGases.TiO2

Ideal gas "TiO2" from NASA Glenn coefficients

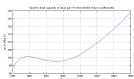
### Information



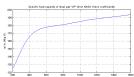
---

## Modelica.Media.IdealGases.SingleGases.U

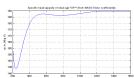
Ideal gas "U" from NASA Glenn coefficients

**Information****Modelica.Media.IdealGases.SingleGases.UF**

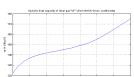
Ideal gas "UF" from NASA Glenn coefficients

**Information****Modelica.Media.IdealGases.SingleGases.UFplus**

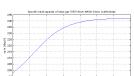
Ideal gas "UF+" from NASA Glenn coefficients

**Information****Modelica.Media.IdealGases.SingleGases.UFminus**

Ideal gas "UF-" from NASA Glenn coefficients

**Information****Modelica.Media.IdealGases.SingleGases.UF2**

Ideal gas "UF2" from NASA Glenn coefficients

**Information****Modelica.Media.IdealGases.SingleGases.UF2plus**

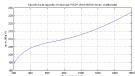
Ideal gas "UF2+" from NASA Glenn coefficients

---

## 1538 Modelica.Media.IdealGases.SingleGases.UF2plus

---

### Information

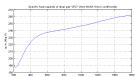


---

## Modelica.Media.IdealGases.SingleGases.UF2minus

Ideal gas "UF2-" from NASA Glenn coefficients

### Information

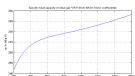


---

## Modelica.Media.IdealGases.SingleGases.UF3

Ideal gas "UF3" from NASA Glenn coefficients

### Information

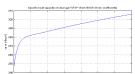


---

## Modelica.Media.IdealGases.SingleGases.UF3plus

Ideal gas "UF3+" from NASA Glenn coefficients

### Information

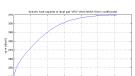


---

## Modelica.Media.IdealGases.SingleGases.UF3minus

Ideal gas "UF3-" from NASA Glenn coefficients

### Information

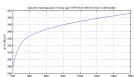


---

## Modelica.Media.IdealGases.SingleGases.UF4

Ideal gas "UF4" from NASA Glenn coefficients

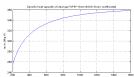
## Information



## Modelica.Media.IdealGases.SingleGases.UF4plus

Ideal gas "UF4+" from NASA Glenn coefficients

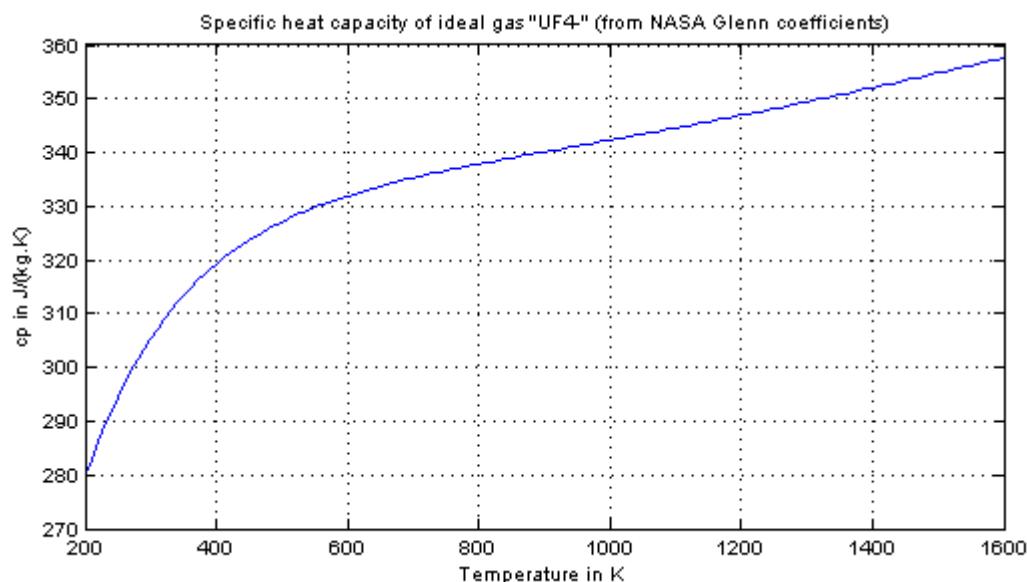
## Information



## Modelica.Media.IdealGases.SingleGases.UF4minus

Ideal gas "UF4-" from NASA Glenn coefficients

## Information



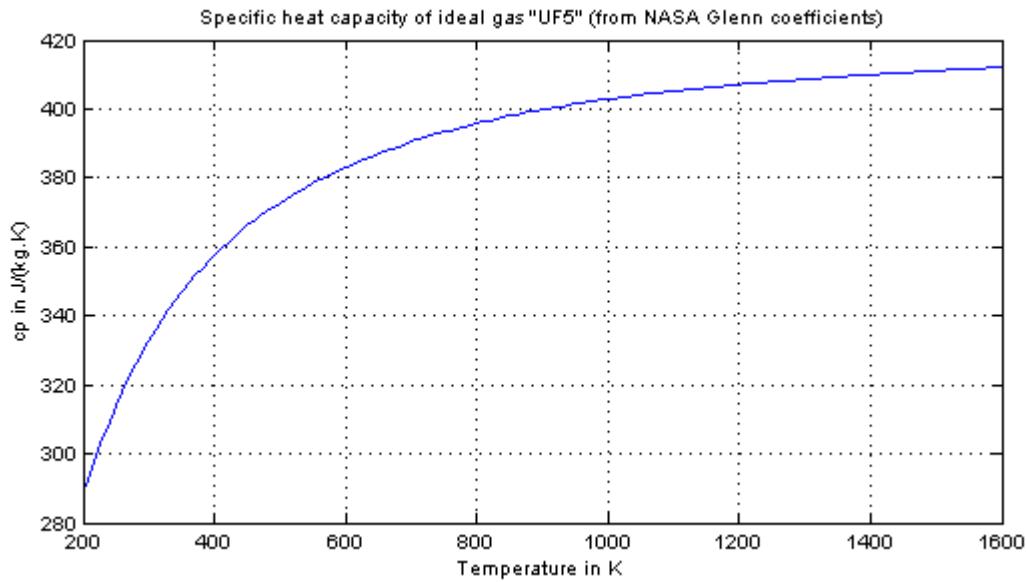
## Modelica.Media.IdealGases.SingleGases.UF5

Ideal gas "UF5" from NASA Glenn coefficients

## 1540 Modelica.Media.IdealGases.SingleGases.UF5

---

### Information

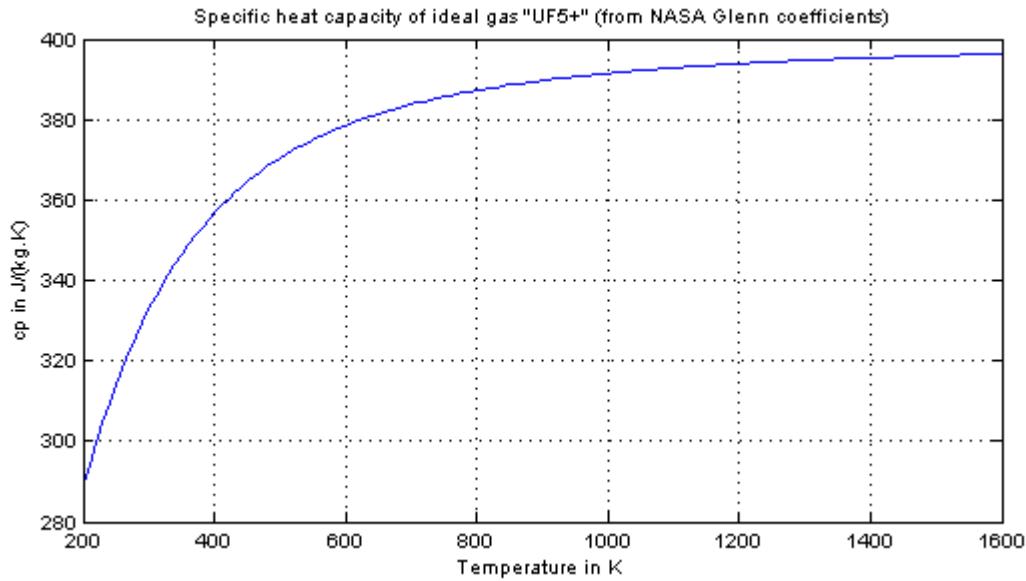


---

## Modelica.Media.IdealGases.SingleGases.UF5plus

Ideal gas "UF5+" from NASA Glenn coefficients

### Information

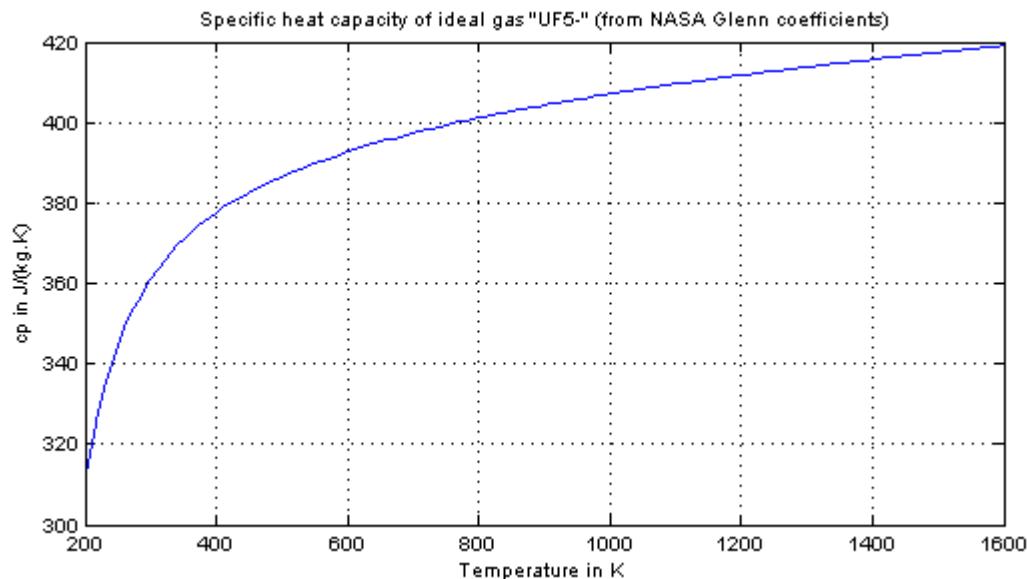


---

## Modelica.Media.IdealGases.SingleGases.UF5minus

Ideal gas "UF5-" from NASA Glenn coefficients

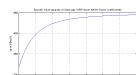
## Information



## Modelica.Media.IdealGases.SingleGases.UF6

Ideal gas "UF6" from NASA Glenn coefficients

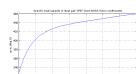
## Information



## Modelica.Media.IdealGases.SingleGases.UF6minus

Ideal gas "UF6-" from NASA Glenn coefficients

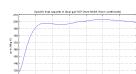
## Information



## Modelica.Media.IdealGases.SingleGases.UO

Ideal gas "UO" from NASA Glenn coefficients

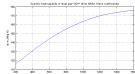
## Information



### Modelica.Media.IdealGases.SingleGases.UOplus

Ideal gas "UO+" from NASA Glenn coefficients

#### Information

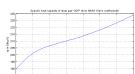


---

### Modelica.Media.IdealGases.SingleGases.UOF

Ideal gas "UOF" from NASA Glenn coefficients

#### Information

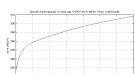


---

### Modelica.Media.IdealGases.SingleGases.UOF2

Ideal gas "UOF2" from NASA Glenn coefficients

#### Information

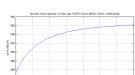


---

### Modelica.Media.IdealGases.SingleGases.UOF3

Ideal gas "UOF3" from NASA Glenn coefficients

#### Information

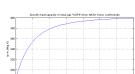


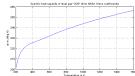
---

### Modelica.Media.IdealGases.SingleGases.UOF4

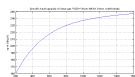
Ideal gas "UOF4" from NASA Glenn coefficients

#### Information

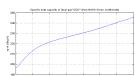


**Modelica.Media.IdealGases.SingleGases.UO2****Ideal gas "UO2" from NASA Glenn coefficients****Information**

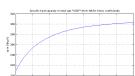
---

**Modelica.Media.IdealGases.SingleGases.UO2plus****Ideal gas "UO2+" from NASA Glenn coefficients****Information**

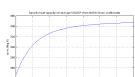
---

**Modelica.Media.IdealGases.SingleGases.UO2minus****Ideal gas "UO2-" from NASA Glenn coefficients****Information**

---

**Modelica.Media.IdealGases.SingleGases.UO2F****Ideal gas "UO2F" from NASA Glenn coefficients****Information**

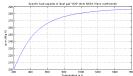
---

**Modelica.Media.IdealGases.SingleGases.UO2F2****Ideal gas "UO2F2" from NASA Glenn coefficients****Information**

**Modelica.Media.IdealGases.SingleGases.UO3**

Ideal gas "UO3" from NASA Glenn coefficients

**Information**

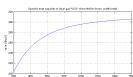


---

**Modelica.Media.IdealGases.SingleGases.UO3minus**

Ideal gas "UO3-" from NASA Glenn coefficients

**Information**

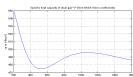


---

**Modelica.Media.IdealGases.SingleGases.V**

Ideal gas "V" from NASA Glenn coefficients

**Information**

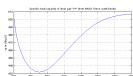


---

**Modelica.Media.IdealGases.SingleGases.Vplus**

Ideal gas "V+" from NASA Glenn coefficients

**Information**

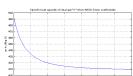


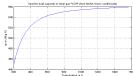
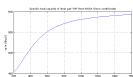
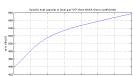
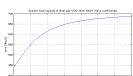
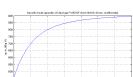
---

**Modelica.Media.IdealGases.SingleGases.Vminus**

Ideal gas "V-" from NASA Glenn coefficients

**Information**



**Modelica.Media.IdealGases.SingleGases.VCL4****Ideal gas "VCl4" from NASA Glenn coefficients****Information****Modelica.Media.IdealGases.SingleGases.VN****Ideal gas "VN" from NASA Glenn coefficients****Information****Modelica.Media.IdealGases.SingleGases.VO****Ideal gas "VO" from NASA Glenn coefficients****Information****Modelica.Media.IdealGases.SingleGases.VO2****Ideal gas "VO2" from NASA Glenn coefficients****Information****Modelica.Media.IdealGases.SingleGases.V4O10****Ideal gas "V4O10" from NASA Glenn coefficients****Information**

---

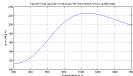
## 1546 Modelica.Media.IdealGases.SingleGases.W

---

### Modelica.Media.IdealGases.SingleGases.W

Ideal gas "W" from NASA Glenn coefficients

#### Information

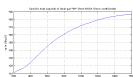


---

### Modelica.Media.IdealGases.SingleGases.Wplus

Ideal gas "W+" from NASA Glenn coefficients

#### Information

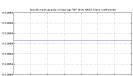


---

### Modelica.Media.IdealGases.SingleGases.Wminus

Ideal gas "W-" from NASA Glenn coefficients

#### Information

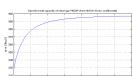


---

### Modelica.Media.IdealGases.SingleGases.WCl6

Ideal gas "WCl6" from NASA Glenn coefficients

#### Information

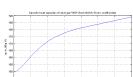


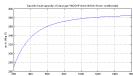
---

### Modelica.Media.IdealGases.SingleGases.WO

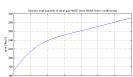
Ideal gas "WO" from NASA Glenn coefficients

#### Information

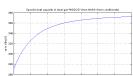


**Modelica.Media.IdealGases.SingleGases.WOCL4****Ideal gas "WOC14" from NASA Glenn coefficients****Information**

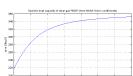
---

**Modelica.Media.IdealGases.SingleGases.WO2****Ideal gas "WO2" from NASA Glenn coefficients****Information**

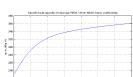
---

**Modelica.Media.IdealGases.SingleGases.WO2CL2****Ideal gas "WO2Cl2" from NASA Glenn coefficients****Information**

---

**Modelica.Media.IdealGases.SingleGases.WO3****Ideal gas "WO3" from NASA Glenn coefficients****Information**

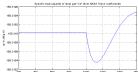
---

**Modelica.Media.IdealGases.SingleGases.WO3minus****Ideal gas "WO3-" from NASA Glenn coefficients****Information**

**Modelica.Media.IdealGases.SingleGases.Xe**

Ideal gas "Xe" from NASA Glenn coefficients

**Information**

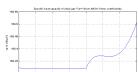


---

**Modelica.Media.IdealGases.SingleGases.Xeplus**

Ideal gas "Xe+" from NASA Glenn coefficients

**Information**

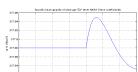


---

**Modelica.Media.IdealGases.SingleGases.Zn**

Ideal gas "Zn" from NASA Glenn coefficients

**Information**

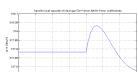


---

**Modelica.Media.IdealGases.SingleGases.Znplus**

Ideal gas "Zn+" from NASA Glenn coefficients

**Information**

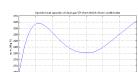


---

**Modelica.Media.IdealGases.SingleGases.Zr**

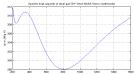
Ideal gas "Zr" from NASA Glenn coefficients

**Information**

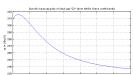


**Modelica.Media.IdealGases.SingleGases.Zrplus**

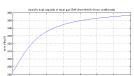
Ideal gas "Zr+" from NASA Glenn coefficients

**Information****Modelica.Media.IdealGases.SingleGases.Zrminus**

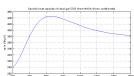
Ideal gas "Zr-" from NASA Glenn coefficients

**Information****Modelica.Media.IdealGases.SingleGases.ZrN**

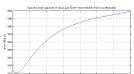
Ideal gas "ZrN" from NASA Glenn coefficients

**Information****Modelica.Media.IdealGases.SingleGases.ZrO**

Ideal gas "ZrO" from NASA Glenn coefficients

**Information****Modelica.Media.IdealGases.SingleGases.ZrOplus**

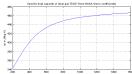
Ideal gas "ZrO+" from NASA Glenn coefficients

**Information**

## Modelica.Media.IdealGases.SingleGases.ZrO2

Ideal gas "ZrO2" from NASA Glenn coefficients

### Information



---

## Modelica.Media.Incompressible

Medium model for T-dependent properties, defined by tables or polynomials

### Information

#### Incompressible media package

This package provides a structure and examples of how to create simple medium models of incompressible fluids, meaning fluids with very little pressure influence on density. The medium properties is typically described in terms of tables, functions or polynomial coefficients.

#### Definitions

The common meaning of *incompressible* is that properties like density and enthalpy are independent of pressure. Thus properties are conveniently described as functions of temperature, e.g. as polynomials density( $T$ ) and  $cp(T)$ . However, enthalpy can not be independent of pressure since  $h = u - p/d$ . For liquids it is anyway common to neglect this dependence since for constant density the neglected term is  $(p - p_0)/d$ , which in comparison with  $cp$  is very small for most liquids. For water, the equivalent change of temperature to increasing pressure 1 bar is 0.025 Kelvin.

Two boolean flags are used to choose how enthalpy and inner energy is calculated:

- **enthalpyOfT**=true, means assuming that enthalpy is only a function of temperature, neglecting the pressure dependent term.
- **singleState**=true, means also neglect the pressure influence on inner energy, which makes all medium properties pure functions of temperature.

The default setting for both these flags is true, which enables the simulation tool to choose temperature as the only medium state and avoids non-linear equation systems, see the section about [Static state selection](#) in the Modelica.Media user's Guide.

#### Contents

Currently, the package contains the following parts:

1. Table based medium models
2. Example medium models

A few examples are given in the Examples package. The model [Examples.TestGlycol](#) shows how the medium models can be used. For more realistic examples of how to implement volume models with medium properties look in the [Medium usage section](#) of the User's guide.

#### Package Content

Name	Description
Common	Common data structures

---

 TableBased	Incompressible medium properties based on tables
 Examples	

---

## Modelica.Media.Incompressible.Common

### Common data structures

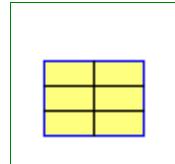
#### Package Content

Name	Description
 BaseProps_Tpoly	Fluid state record

---

## Modelica.Media.Incompressible.Common.BaseProps\_Tpoly

### Fluid state record



#### Modelica definition

```
record BaseProps_Tpoly "Fluid state record"
  extends Modelica.Media.Interfaces.PartialMedium.ThermodynamicState;
  //      SI.SpecificHeatCapacity cp "specific heat capacity";
  SI.Temperature T "temperature";
  SI.Pressure p "pressure";
  //      SI.Density d "density";
end BaseProps_Tpoly;
```

---

## Modelica.Media.Incompressible.TableBased

### Incompressible medium properties based on tables

#### Information

This is the base package for medium models of incompressible fluids based on tables. The minimal data to provide for a useful medium description is tables of density and heat capacity as functions of temperature.

#### Using the package TableBased

To implement a new medium model, create a package that **extends** TableBased and provides one or more of the constant tables:

```
tableDensity      = [T, d];
tableHeatCapacity = [T, Cp];
tableConductivity = [T, lam];
tableViscosity   = [T, eta];
tableVaporPressure = [T, pVap];
```

The table data is used to fit constant polynomials of order **npol**, the temperature data points do not need to be same for different properties. Properties like enthalpy, inner energy and entropy are calculated consistently from integrals and derivatives of d(T) and Cp(T). The minimal data for a useful medium model is thus density and heat capacity. Transport properties and vapor pressure are optional, if the data tables are empty the corresponding function calls can not be used.

## Package Content

Name	Description
enthalpyOfT=true	true if enthalpy is approximated as a function of T only, (p-dependence neglected)
densityOfT=size(tableDensity, 1) > 1	true if density is a function of temperature
T_min	Minimum temperature valid for medium model
T_max	Maximum temperature valid for medium model
T0=273.15	reference Temperature
h0=0	reference enthalpy at T0, reference_p
s0=0	reference entropy at T0, reference_p
MM_const=0.1	Molar mass
npol=2	degree of polynomial used for fitting
neta=size(tableViscosity, 1)	number of data points for viscosity
tableDensity	Table for rho(T)
tableHeatCapacity	Table for Cp(T)
tableViscosity	Table for eta(T)
tableVaporPressure	Table for pVap(T)
tableConductivity	Table for lambda(T)
TinK	true if T[K], Kelvin used for table temperatures
hasDensity=not (size(tableDensity, 1) == 0)	
hasHeatCapacity=not (size(tableHeatCapacity, 1) == 0)	
hasViscosity=not (size(tableViscosity, 1) == 0)	
hasVaporPressure=not (size(tableVaporPressure, 1) == 0)	
invTK=invertTemp(tableViscosity[:, 1], TinK)	
poly_rho;if hasDensity then Poly.fitting(tableDensity[:, 1], tableDensity[:, 2], npol) else zeros(npol + 1)	
poly_Cp;if hasHeatCapacity then Poly.fitting(tableHeatCapacity[:, 1], tableHeatCapacity[:, 2], npol) else zeros(npol + 1)	
poly_eta;if hasViscosity then Poly.fitting(invTK, Math.log(tableViscosity[:, 2]), npol) else zeros(npol + 1)	
poly_pVap;if hasVaporPressure then Poly.fitting(tableVaporPressure[:, 1], tableVaporPressure[:, 2], npol) else zeros(npol + 1)	
poly_lam;if size(tableConductivity, 1) > 0 then Poly.fitting(tableConductivity[:, 1], tableConductivity[:, 2], npol) else zeros(npol + 1)	
 invertTemp	function to invert temperatures
 BaseProperties	Base properties of T dependent medium
 setState_pTX	Returns state record, given pressure and temperature
 setState_dTX	Returns state record, given pressure and temperature

<code>setState_pT</code>	returns state record as function of p and T
<code>setState_phX</code>	Returns state record, given pressure and specific enthalpy
<code>setState_ph</code>	returns state record as function of p and h
<code>setState_psX</code>	Returns state record, given pressure and specific entropy
<code>setState_ps</code>	returns state record as function of p and s
<code>specificHeatCapacityCv</code>	Specific heat capacity at constant volume (or pressure) of medium
<code>specificHeatCapacityCp</code>	Specific heat capacity at constant volume (or pressure) of medium
<code>dynamicViscosity</code>	
<code>thermalConductivity</code>	
<code>s_T</code>	compute specific entropy
<code>specificEntropy</code>	
<code>h_T</code>	Compute specific enthalpy from temperature
<code>h_T_der</code>	Compute specific enthalpy from temperature
<code>h_pT</code>	Compute specific enthalpy from pressure and temperature
<code>density_T</code>	
<code>temperature</code>	
<code>pressure</code>	
<code>density</code>	
<code>specificEnthalpy</code>	
<code>specificInternalEnergy</code>	
<code>T_ph</code>	Compute temperature from pressure and specific enthalpy
<code>T_ps</code>	Compute temperature from pressure and specific enthalpy
<code>Polynomials_Temp</code>	Temporary Functions operating on polynomials (including polynomial fitting); only to be used in Modelica.Media.Incompressible.TableBased
<b>Inherited</b>	
<code>mediumName="unusablePartialMedium"</code>	Name of the medium
<code>substanceNames={mediumName}</code>	Names of the mixture substances. Set <code>substanceNames={mediumName}</code> if only one substance.
<code>extraPropertiesNames=fill("", 0)</code>	Names of the additional (extra) transported properties. Set <code>extraPropertiesNames=fill("",0)</code> if unused
<code>singleState</code>	= true, if u and d are not a function of pressure
<code>reducedX=true</code>	= true if medium contains the equation $\sum(X) = 1.0$ ; set <code>reducedX=true</code> if only one substance (see docu for details)
<code>fixedX=false</code>	= true if medium contains the equation $X = \text{reference\_}X$
<code>reference_p=101325</code>	Reference pressure of Medium: default 1 atmosphere
<code>reference_T=298.15</code>	Reference temperature of Medium: default 25 deg Celsius

## 1554 Modelica.Media.Incompressible.TableBased

<code>reference_X=if nX == 0 then fill(0, nX) else fill(1/nX, nX)</code>	Default mass fractions of medium
<code>p_default=101325</code>	Default value for pressure of medium (for initialization)
<code>T_default=Modelica.Slunits.Conversions.from_degC(20)</code>	Default value for temperature of medium (for initialization)
<code>h_default=specificEnthalpy_pTX(p_default, T_default, X_default)</code>	Default value for specific enthalpy of medium (for initialization)
<code>X_default=reference_X</code>	Default value for mass fractions of medium (for initialization)
<code>nS=size(substanceNames, 1)</code>	Number of substances
<code>nX=if nS == 1 then 0 else nS</code>	Number of mass fractions (= 0, if only one substance)
<code>nXi=if fixedX then 0 else if reducedX then nS - 1 else nX</code>	Number of structurally independent mass fractions (see docu for details)
<code>nC=size(extraPropertiesNames, 1)</code>	Number of extra (outside of standard mass-balance) transported properties
 <a href="#">FluidConstants</a>	critical, triple, molecular and other standard data of fluid
 <a href="#">ThermodynamicState</a>	Minimal variable set that is available as input argument to every medium function
 <a href="#">BasePropertiesRecord</a>	Variables contained in every instance of BaseProperties
 <a href="#">prandtlNumber</a>	Return the Prandtl number
 <a href="#">specificGibbsEnergy</a>	Return specific Gibbs energy
 <a href="#">specificHelmholtzEnergy</a>	Return specific Helmholtz energy
 <a href="#">heatCapacity_cp</a>	alias for deprecated name
 <a href="#">heatCapacity_cv</a>	alias for deprecated name
 <a href="#">isentropicExponent</a>	Return isentropic exponent
 <a href="#">isentropicEnthalpy</a>	Return isentropic enthalpy
 <a href="#">velocityOfSound</a>	Return velocity of sound
 <a href="#">isobaricExpansionCoefficient</a>	Return overall the isobaric expansion coefficient beta
 <a href="#">beta</a>	alias for isobaricExpansionCoefficient for user convenience
 <a href="#">isothermalCompressibility</a>	Return overall the isothermal compressibility factor
 <a href="#">kappa</a>	alias of isothermalCompressibility for user convenience
 <a href="#">density_derP_h</a>	Return density derivative wrt pressure at const specific enthalpy
 <a href="#">density_derH_p</a>	Return density derivative wrt specific enthalpy at constant pressure
 <a href="#">density_derP_T</a>	Return density derivative wrt pressure at const temperature
 <a href="#">density_derT_p</a>	Return density derivative wrt temperature at constant pressure
 <a href="#">density_derX</a>	Return density derivative wrt mass fraction
 <a href="#">molarMass</a>	Return the molar mass of the medium
 <a href="#">specificEnthalpy_pTX</a>	Return specific enthalpy from p, T, and X or Xi
 <a href="#">density_pTX</a>	Return density from p, T, and X or Xi

 <code>temperature_phX</code>	Return temperature from p, h, and X or Xi
 <code>density_phX</code>	Return density from p, h, and X or Xi
 <code>temperature_psX</code>	Return temperature from p,s, and X or Xi
 <code>density_psX</code>	Return density from p, s, and X or Xi
 <code>specificEnthalpy_psX</code>	Return specific enthalpy from p, s, and X or Xi
<code>AbsolutePressure</code>	Type for absolute pressure with medium specific attributes
<code>Density</code>	Type for density with medium specific attributes
<code>DynamicViscosity</code>	Type for dynamic viscosity with medium specific attributes
<code>EnthalpyFlowRate</code>	Type for enthalpy flow rate with medium specific attributes
<code>MassFlowRate</code>	Type for mass flow rate with medium specific attributes
<code>MassFraction</code>	Type for mass fraction with medium specific attributes
<code>MoleFraction</code>	Type for mole fraction with medium specific attributes
<code>MolarMass</code>	Type for molar mass with medium specific attributes
<code>MolarVolume</code>	Type for molar volume with medium specific attributes
<code>IsentropicExponent</code>	Type for isentropic exponent with medium specific attributes
<code>SpecificEnergy</code>	Type for specific energy with medium specific attributes
<code>SpecificInternalEnergy</code>	Type for specific internal energy with medium specific attributes
<code>SpecificEnthalpy</code>	Type for specific enthalpy with medium specific attributes
<code>SpecificEntropy</code>	Type for specific entropy with medium specific attributes
<code>SpecificHeatCapacity</code>	Type for specific heat capacity with medium specific attributes
<code>SurfaceTension</code>	Type for surface tension with medium specific attributes
<code>Temperature</code>	Type for temperature with medium specific attributes
<code>ThermalConductivity</code>	Type for thermal conductivity with medium specific attributes
<code>PrandtlNumber</code>	Type for Prandtl number with medium specific attributes
<code>VelocityOfSound</code>	Type for velocity of sound with medium specific attributes
<code>ExtraProperty</code>	Type for unspecified, mass-specific property transported by flow
<code>CumulativeExtraProperty</code>	Type for conserved integral of unspecified, mass specific property
<code>ExtraPropertyFlowRate</code>	Type for flow rate of unspecified, mass-specific property
<code>IsobaricExpansionCoefficient</code>	Type for isobaric expansion coefficient with medium specific attributes
<code>DipoleMoment</code>	Type for dipole moment with medium specific attributes
<code>DerDensityByPressure</code>	Type for partial derivative of density with respect to pressure with medium specific attributes
<code>DerDensityByEnthalpy</code>	Type for partial derivative of density with respect to enthalpy with medium specific attributes
<code>DerEnthalpyByPressure</code>	Type for partial derivative of enthalpy with respect to pressure with medium specific attributes
<code>DerDensityByTemperature</code>	Type for partial derivative of density with respect to temperature with medium specific attributes
 <code>Choices</code>	Types, constants to define menu choices

## Types and constants

```
constant Boolean enthalpyOfT=true  
"true if enthalpy is approximated as a function of T only, (p-dependence  
neglected)";  
  
constant Boolean densityOfT = size(tableDensity,1) > 1  
"true if density is a function of temperature";  
  
constant Temperature T_min "Minimum temperature valid for medium model";  
  
constant Temperature T_max "Maximum temperature valid for medium model";  
  
constant Temperature T0=273.15 "reference Temperature";  
  
constant SpecificEnthalpy h0=0 "reference enthalpy at T0, reference_p";  
  
constant SpecificEntropy s0=0 "reference entropy at T0, reference_p";  
  
constant MolarMass MM_const=0.1 "Molar mass";  
  
constant Integer npol=2 "degree of polynomial used for fitting";  
  
constant Integer neta=size(tableViscosity,1)  
"number of data points for viscosity";  
  
constant Real[::,:] tableDensity "Table for rho(T)";  
  
constant Real[::,:] tableHeatCapacity "Table for Cp(T)";  
  
constant Real[::,:] tableViscosity "Table for eta(T)";  
  
constant Real[::,:] tableVaporPressure "Table for pVap(T)";  
  
constant Real[::,:] tableConductivity "Table for lambda(T)";  
  
constant Boolean TinK "true if T[K], Kelvin used for table temperatures";  
  
constant Boolean hasDensity = not (size(tableDensity,1)==0);  
  
constant Boolean hasHeatCapacity = not (size(tableHeatCapacity,1)==0);  
  
constant Boolean hasViscosity = not (size(tableViscosity,1)==0);  
  
constant Boolean hasVaporPressure = not (size(tableVaporPressure,1)==0);  
  
final constant Real invTK[neta] = invertTemp(tableViscosity[:,1],TinK);  
  
final constant Real poly_rho[:] = if hasDensity then  
                                Poly.fitting(tableDensity[:,1],tableDensi  
ty[:,2],npol) else  
                                zeros(npol+1);
```

```

final constant Real poly_Cp[:] = if hasHeatCapacity then
                                    Poly.fitting(tableHeatCapacity[:,1],table
HeatCapacity[:,2],nopol) else
                                    zeros(nopol+1);

final constant Real poly_eta[:] = if hasViscosity then
                                    Poly.fitting(invTK,
Math.log(tableViscosity[:,2]),nopol) else
                                    zeros(nopol+1);

final constant Real poly_pVap[:] = if hasVaporPressure then
                                    Poly.fitting(tableVaporPressure[:,1],tabl
eVaporPressure[:,2],nopol) else
                                    zeros(nopol+1);

final constant Real poly_lam[:] = if size(tableConductivity,1)>0 then
                                    Poly.fitting(tableConductivity[:,1],table
Conductivity[:,2],nopol) else
                                    zeros(nopol+1);

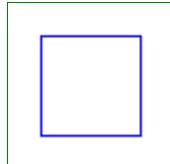
```

**Modelica.Media.Incompressible.TableBased.invertTemp****function to invert temperatures****Inputs**

Name	Default	Description
table[:]		table temperature data
Tink		flag for Celsius or Kelvin

**Outputs**

Name	Description
invTable[size(table, 1)]	inverted temperatures

**Modelica.Media.Incompressible.TableBased.BaseProperties****Base properties of T dependent medium****Information**

Note that the inner energy neglects the pressure dependence, which is only true for an incompressible medium with  $d = \text{constant}$ . The neglected term is  $p\text{-reference\_p}/\rho * (T/\rho)^*(\partial \rho / \partial T)$ . This is very small for liquids due to proportionality to  $1/d^2$ , but can be problematic for gases that are modeled incompressible.

Enthalpy is never a function of T only ( $h = h(T) + (p\text{-reference\_p})/d$ ), but the error is also small and non-linear systems can be avoided. In particular, non-linear systems are small and local as opposed to large and over all volumes.

Entropy is calculated as

$$s = s_0 + \int (C_p(T)/T) dt$$

## 1558 Modelica.Media.Incompressible.TableBased.BaseProperties

---

which is only exactly true for a fluid with constant density  $d=d_0$ .

### Parameters

Name	Default	Description
T_start	298.15	initial temperature [K]
R	Modelica.Constants.R	Gas constant (of mixture if applicable) [J/(kg.K)]
p_bar	Cv.to_bar(p)	Absolute pressure of medium in [bar] [bar]
standardOrderComponents	true	if true, last element in components is computed from 1-sum(Xi)
<b>Advanced</b>		
preferredMediumStates	false	= true if StateSelect.prefer shall be used for the independent property variables of the medium

---

## Modelica.Media.Incompressible.TableBased.setState\_pTX

Returns state record, given pressure and temperature



### Inputs

Name	Default	Description
p		Pressure [Pa]
T		Temperature [K]
X[:]	reference_X	Mass fractions [kg/kg]

### Outputs

Name	Description
state	

---

## Modelica.Media.Incompressible.TableBased.setState\_dTX

Returns state record, given pressure and temperature



### Inputs

Name	Default	Description
d		density [kg/m <sup>3</sup> ]
T		Temperature [K]
X[:]	reference_X	Mass fractions [kg/kg]

### Outputs

Name	Description
state	

**Modelica.Media.Incompressible.TableBased.setState\_pT**

returns state record as function of p and T

**Inputs**

Name	Default	Description
p		pressure [Pa]
T		temperature [K]

**Outputs**

Name	Description
state	thermodynamic state

**Modelica.Media.Incompressible.TableBased.setState\_phX**

Returns state record, given pressure and specific enthalpy

**Inputs**

Name	Default	Description
p		Pressure [Pa]
h		Specific enthalpy [J/kg]
X[:]	reference_X	Mass fractions [kg/kg]

**Outputs**

Name	Description
state	

**Modelica.Media.Incompressible.TableBased.setState\_ph**

returns state record as function of p and h

**Inputs**

Name	Default	Description
p		pressure [Pa]
h		specific enthalpy [J/kg]

**Outputs**

Name	Description
state	thermodynamic state

**Modelica.Media.Incompressible.TableBased.setState\_psX**

Returns state record, given pressure and specific entropy

---

## 1560 Modelica.Media.Incompressible.TableBased.setState\_psX

---

### Inputs

Name	Default	Description
p		Pressure [Pa]
s		Specific entropy [J/(kg.K)]
X[:]	reference_X	Mass fractions [kg/kg]

### Outputs

Name	Description
state	

---

## Modelica.Media.Incompressible.TableBased.setState\_ps

returns state record as function of p and s

### Inputs

Name	Default	Description
p		pressure [Pa]
s		specific entropy [J/(kg.K)]

### Outputs

Name	Description
state	thermodynamic state

---

## Modelica.Media.Incompressible.TableBased.specificHeatCapacityCv

Specific heat capacity at constant volume (or pressure) of medium



### Inputs

Name	Default	Description
state		

### Outputs

Name	Description
cv	Specific heat capacity at constant volume [J/(kg.K)]

---

## Modelica.Media.Incompressible.TableBased.specificHeatCapacityCp

Specific heat capacity at constant volume (or pressure) of medium



### Inputs

Name	Default	Description
state		

**Outputs**

Name	Description
cp	Specific heat capacity at constant pressure [J/(kg.K)]

**Modelica.Media.Incompressible.TableBased.dynamicViscosity****Inputs**

Name	Default	Description
state		

**Outputs**

Name	Description
eta	Dynamic viscosity [Pa.s]

**Modelica.Media.Incompressible.TableBased.thermalConductivity****Inputs**

Name	Default	Description
state		

**Outputs**

Name	Description
lambda	Thermal conductivity [W/(m.K)]

**Modelica.Media.Incompressible.TableBased.s\_T**

compute specific entropy

**Inputs**

Name	Default	Description
T		temperature [K]

**Outputs**

Name	Description
s	specific entropy [J/(kg.K)]

**Modelica.Media.Incompressible.TableBased.specificEntropy****Inputs**

Name	Default	Description
state		

## 1562 Modelica.Media.Incompressible.TableBased.specificEntropy

---

### Outputs

Name	Description
s	Specific entropy [J/(kg.K)]

---

### Modelica.Media.Incompressible.TableBased.h\_T

Compute specific enthalpy from temperature



### Inputs

Name	Default	Description
T		Temperature [K]

### Outputs

Name	Description
h	Specific enthalpy at p, T [J/kg]

---

### Modelica.Media.Incompressible.TableBased.h\_T\_der

Compute specific enthalpy from temperature



### Inputs

Name	Default	Description
T		Temperature [K]
dT		temperature derivative

### Outputs

Name	Description
dh	derivative of Specific enthalpy at T

---

### Modelica.Media.Incompressible.TableBased.h\_pT

Compute specific enthalpy from pressure and temperature



### Inputs

Name	Default	Description
p		Pressure [Pa]
T		Temperature [K]
densityOfT	false	include or neglect density derivative dependence of enthalpy

### Outputs

Name	Description
h	Specific enthalpy at p, T [J/kg]

**Modelica.Media.Incompressible.TableBased.density\_T****Inputs**

Name	Default	Description
T		temperature [K]

**Outputs**

Name	Description
d	density [kg/m3]

**Modelica.Media.Incompressible.TableBased.temperature****Inputs**

Name	Default	Description
state		

**Outputs**

Name	Description
T	Temperature [K]

**Modelica.Media.Incompressible.TableBased.pressure****Inputs**

Name	Default	Description
state		

**Outputs**

Name	Description
p	Pressure [Pa]

**Modelica.Media.Incompressible.TableBased.density****Inputs**

Name	Default	Description
state		

**Outputs**

Name	Description
d	Density [kg/m3]

---

## 1564 Modelica.Media.Incompressible.TableBased.specificEnthalpy

---

### Modelica.Media.Incompressible.TableBased.specificEnthalpy



#### Inputs

Name	Default	Description
state		

#### Outputs

Name	Description
h	Specific enthalpy [J/kg]

---

### Modelica.Media.Incompressible.TableBased.specificInternalEnergy



#### Inputs

Name	Default	Description
state		

#### Outputs

Name	Description
u	Specific internal energy [J/kg]

---

### Modelica.Media.Incompressible.TableBased.T\_ph

Compute temperature from pressure and specific enthalpy

#### Inputs

Name	Default	Description
p		pressure [Pa]
h		specific enthalpy [J/kg]

#### Outputs

Name	Description
T	temperature [K]

---

### Modelica.Media.Incompressible.TableBased.T\_ps

Compute temperature from pressure and specific entropy

#### Inputs

Name	Default	Description
p		pressure [Pa]
s		specific entropy [J/(kg.K)]

## Outputs

Name	Description
T	temperature [K]

---

## Modelica.Media.Incompressible.TableBased.Polynomials\_Temp

Temporary Functions operating on polynomials (including polynomial fitting); only to be used in Modelica.Media.Incompressible.TableBased

## Information

This package contains functions to operate on polynomials, in particular to determine the derivative and the integral of a polynomial and to use a polynomial to fit a given set of data points.

**Copyright © 2004-2007, Modelica Association and DLR.**

*This package is free software. It can be redistributed and/or modified under the terms of the Modelica license, see the license conditions and the accompanying disclaimer in the documentation of package Modelica in file "Modelica/package.mo".*

## Package Content

Name	Description
(f) evaluate	Evaluate polynomial at a given abszissa value
(f) derivative	Derivative of polynomial
(f) derivativeValue	Value of derivative of polynomial at abszissa value u
(f) secondDerivativeValue	Value of 2nd derivative of polynomial at abszissa value u
(f) integral	Indefinite integral of polynomial p(u)
(f) integralValue	Integral of polynomial p(u) from u_low to u_high
(f) fitting	Computes the coefficients of a polynomial that fits a set of data points in a least-squares sense
(f) evaluate_der	Evaluate polynomial at a given abszissa value
(f) integralValue_der	Time derivative of integral of polynomial p(u) from u_low to u_high, assuming only u_high as time-dependent (Leibnitz rule)
(f) derivativeValue_der	time derivative of derivative of polynomial

---

## Modelica.Media.Incompressible.TableBased.Polynomials\_Temp.evaluate

Evaluate polynomial at a given abszissa value



## Inputs

Name	Default	Description
p[:]		Polynomial coefficients (p[1] is coefficient of highest power)
u		Abszissa value

## Outputs

Name	Description

---

**1566 Modelica.Media.Incompressible.TableBased.Polynomials\_Temp.evaluate**

y	Value of polynomial at u
---	--------------------------

---

**Modelica.Media.Incompressible.TableBased.Polynomials\_Temp.derivative****Derivative of polynomial****Inputs**

Name	Default	Description
p1[:]		Polynomial coefficients (p1[1] is coefficient of highest power)

**Outputs**

Name	Description
p2[size(p1, 1) - 1]	Derivative of polynomial p1

---

**Modelica.Media.Incompressible.TableBased.Polynomials\_Temp.derivativeValue****Value of derivative of polynomial at abszissa value u****Inputs**

Name	Default	Description
p[:]		Polynomial coefficients (p[1] is coefficient of highest power)
u		Abszissa value

**Outputs**

Name	Description
y	Value of derivative of polynomial at u

---

**Modelica.Media.Incompressible.TableBased.Polynomials\_Temp.secondDerivativeValue****Value of 2nd derivative of polynomial at abszissa value u****Inputs**

Name	Default	Description
p[:]		Polynomial coefficients (p[1] is coefficient of highest power)
u		Abszissa value

**Outputs**

Name	Description
y	Value of 2nd derivative of polynomial at u

---

**Modelica.Media.Incompressible.TableBased.Polynomials\_Temp.integral****Indefinite integral of polynomial p(u)**

## Inputs

Name	Default	Description
p1[:]		Polynomial coefficients (p1[1] is coefficient of highest power)

## Outputs

Name	Description
p2[size(p1, 1) + 1]	Polynomial coefficients of indefinite integral of polynomial p1 (polynomial p2 + C is the indefinite integral of p1, where C is an arbitrary constant)

## Modelica.Media.Incompressible.TableBased.Polynomials\_Temp.integralValue



Integral of polynomial p(u) from u\_low to u\_high

## Inputs

Name	Default	Description
p[:]		Polynomial coefficients
u_high		High integrand value
u_low	0	Low integrand value, default 0

## Outputs

Name	Description
integral	Integral of polynomial p from u_low to u_high

## Modelica.Media.Incompressible.TableBased.Polynomials\_Temp.fitting



Computes the coefficients of a polynomial that fits a set of data points in a least-squares sense

## Information

Polynomials.fitting(u,y,n) computes the coefficients of a polynomial p(u) of degree "n" that fits the data "p(u[i]) - y[i]" in a least squares sense. The polynomial is returned as a vector p[n+1] that has the following definition:

$$p(u) = p[1]*u^n + p[2]*u^{(n-1)} + \dots + p[n]*u + p[n+1];$$

## Inputs

Name	Default	Description
u[:]		Abscissa data values
y[size(u, 1)]		Ordinate data values
n		Order of desired polynomial that fits the data points (u,y)

## Outputs

Name	Description
p[n + 1]	Polynomial coefficients of polynomial that fits the date points

---

## 1568 Modelica.Media.Incompressible.TableBased.Polynomials\_Temp.evaluate\_der

---

### Modelica.Media.Incompressible.TableBased.Polynomials\_Temp.evaluate\_der



Evaluate polynomial at a given abszissa value

#### Inputs

Name	Default	Description
p[:]		Polynomial coefficients (p[1] is coefficient of highest power)
u		Abszissa value
du		Abszissa value

#### Outputs

Name	Description
dy	Value of polynomial at u

---

### Modelica.Media.Incompressible.TableBased.Polynomials\_Temp.integralValue\_der



Time derivative of integral of polynomial p(u) from u\_low to u\_high, assuming only u\_high as time-dependent (Leibnitz rule)

#### Inputs

Name	Default	Description
p[:]		Polynomial coefficients
u_high		High integrand value
u_low	0	Low integrand value, default 0
du_high		High integrand value
du_low	0	Low integrand value, default 0

#### Outputs

Name	Description
dintegral	Integral of polynomial p from u_low to u_high

---

### Modelica.Media.Incompressible.TableBased.Polynomials\_Temp.derivativeValue\_der



time derivative of derivative of polynomial

#### Inputs

Name	Default	Description
p[:]		Polynomial coefficients (p[1] is coefficient of highest power)
u		Abszissa value
du		delta of abszissa value

#### Outputs

Name	Description
dy	time-derivative of derivative of polynomial w.r.t. input variable at u

## Modelica.Media.Incompressible.Examples

### Information

#### Examples of incompressible media

This package provides a few examples of how to construct medium models for incompressible fluids. The package contains:

- **Glycol47**, a model of 47% glycol water mixture, based on tables of density and heat capacity as functions of temperature.
- **Essotherm650**, a medium model for thermal oil, also based on tables.

### Package Content

Name	Description
<input type="checkbox"/> Glycol47	1,2-Propylene glycol, 47% mixture with water
<input type="checkbox"/> Essotherm650	Essotherm thermal oil
<input type="checkbox"/> TestGlycol	Test Glycol47 Medium model

---

## Modelica.Media.Incompressible.Examples.Glycol47

### 1,2-Propylene glycol, 47% mixture with water

### Information

---

## Modelica.Media.Incompressible.Examples.Essotherm650

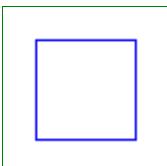
### Essotherm thermal oil

### Information

---

## Modelica.Media.Incompressible.Examples.TestGlycol

### Test Glycol47 Medium model



### Parameters

Name	Default	Description
T_start	298.15	initial temperature [K]
standardOrderComponents	true	if true, last element in components is computed from $1 - \sum(X_i)$
<b>Advanced</b>		
preferredMediumStates	false	= true if StateSelect.prefer shall be used for the independent property variables of the medium

## Modelica.Media.Water

### Medium models for water

#### Information

This package contains different medium models for water:

- **ConstantPropertyLiquidWater**  
Simple liquid water medium (incompressible, constant data).
- **IdealSteam**  
Steam water medium as ideal gas from Media.IdealGases.SingleGases.H2O
- **WaterIF97 derived models**  
High precision water model according to the IAPWS/IF97 standard (liquid, steam, two phase region). Models with different independent variables are provided as well as models valid only for particular regions. The **WaterIF97\_ph** model is valid in all regions and is the recommended one to use.

#### Overview of WaterIF97 derived water models

The WaterIF97 models calculate medium properties for water in the **liquid**, **gas** and **two phase** regions according to the IAPWS/IF97 standard, i.e., the accepted industrial standard and best compromise between accuracy and computation time. It has been part of the ThermoFluid Modelica library and been extended, reorganized and documented to become part of the Modelica Standard library.

An important feature that distinguishes this implementation of the IF97 steam property standard is that this implementation has been explicitly designed to work well in dynamic simulations. Computational performance has been of high importance. This means that there often exist several ways to get the same result from different functions if one of the functions is called often but can be optimized for that purpose.

Three variable pairs can be the independent variables of the model:

1. Pressure **p** and specific enthalpy **h** are the most natural choice for general applications. This is the recommended choice for most general purpose applications, in particular for power plants.
2. Pressure **p** and temperature **T** are the most natural choice for applications where water is always in the same phase, both for liquid water and steam.
3. Density **d** and temperature **T** are explicit variables of the Helmholtz function in the near-critical region and can be the best choice for applications with super-critical or near-critical states.

The following quantities are always computed in Medium.Baseproperties:

Variable	Unit	Description
T	K	temperature
u	J/kg	specific internal energy
d	kg/m <sup>3</sup>	density
p	Pa	pressure
h	J/kg	specific enthalpy

In some cases additional medium properties are needed. A component that needs these optional properties has to call one of the following functions:

Function call	Unit	Description
Medium.dynamicViscosity(medium.state)	Pa.s	dynamic viscosity
Medium.thermalConductivity(medium.state)	W/(m.K)	thermal conductivity
Medium.prandtlNumber(medium.state)	1	Prandtl number
Medium.specificEntropy(medium.state)	J/(kg.K)	specific entropy
Medium.heatCapacity_cp(medium.state)	J/(kg.K)	specific heat capacity at constant pressure
Medium.heatCapacity_cv(medium.state)	J/(kg.K)	specific heat capacity at constant density
Medium.isentropicExponent(medium.state)	1	isentropic exponent

Medium.isentropicEnthalpy(pressure, medium.state)	J/kg	isentropic enthalpy
Medium.velocityOfSound(medium.state)	m/s	velocity of sound
Medium.isobaricExpansionCoefficient(medium.state)	1/K	isobaric expansion coefficient
Medium.isothermalCompressibility(medium.state)	1/Pa	isothermal compressibility
Medium.density_derP_h(medium.state)	kg/(m <sup>3</sup> .Pa)	derivative of density by pressure at constant enthalpy
Medium.density_derH_p(medium.state)	kg <sup>2</sup> /(m <sup>3</sup> .J)	derivative of density by enthalpy at constant pressure
Medium.density_derP_T(medium.state)	kg/(m <sup>3</sup> .Pa)	derivative of density by pressure at constant temperature
Medium.density_derT_p(medium.state)	kg/(m <sup>3</sup> .K)	derivative of density by temperature at constant pressure
Medium.density_derX(medium.state)	kg/m <sup>3</sup>	derivative of density by mass fraction
Medium.molarMass(medium.state)	kg/mol	molar mass

More details are given in [Modelica.Media.UsersGuide.MediumUsage.OptionalProperties](#). Many additional optional functions are defined to compute properties of saturated media, either liquid (bubble point) or vapour (dew point). The argument to such functions is a SaturationProperties record, which can be set starting from either the saturation pressure or the saturation temperature. With reference to a model defining a pressure p, a temperature T, and a SaturationProperties record sat, the following functions are provided:

Function call	Unit	Description
Medium.saturationPressure(T)	Pa	Saturation pressure at temperature T
Medium.saturationTemperature(p)	K	Saturation temperature at pressure p
Medium.saturationTemperature_derP(p)	K/Pa	Derivative of saturation temperature with respect to pressure
Medium.bubbleEnthalpy(sat)	J/kg	Specific enthalpy at bubble point
Medium.dewEnthalpy(sat)	J/kg	Specific enthalpy at dew point
Medium.bubbleEntropy(sat)	J/(kg.K)	Specific entropy at bubble point
Medium.dewEntropy(sat)	J/(kg.K)	Specific entropy at dew point
Medium.bubbleDensity(sat)	kg/m <sup>3</sup>	Density at bubble point
Medium.dewDensity(sat)	kg/m <sup>3</sup>	Density at dew point
Medium.dBubbleDensity_dPressure(sat)	kg/(m <sup>3</sup> .Pa)	Derivative of density at bubble point with respect to pressure
Medium.dDewDensity_dPressure(sat)	kg/(m <sup>3</sup> .Pa)	Derivative of density at dew point with respect to pressure
Medium.dBubbleEnthalpy_dPressure(sat)	J/(kg.Pa)	Derivative of specific enthalpy at bubble point with respect to pressure
Medium.dDewEnthalpy_dPressure(sat)	J/(kg.Pa)	Derivative of specific enthalpy at dew point with respect to pressure
Medium.surfaceTension(sat)	N/m	Surface tension between liquid and vapour phase

Details on usage and some examples are given in: [Modelica.Media.UsersGuide.MediumUsage.TwoPhase](#).

Many further properties can be computed. Using the well-known Bridgman's Tables, all first partial derivatives of the standard thermodynamic variables can be computed easily.

The documentation of the IAPWS/IF97 steam properties can be freely distributed with computer implementations and are included here (in directory Modelica\help\Documentation\IF97documentation):

- [IF97.pdf](#) The standards document for the main part of the IF97.
- [Back3.pdf](#) The backwards equations for region 3.
- [crits.pdf](#) The critical point data.

- [meltsub.pdf](#) The melting- and sublimation line formulation (not implemented)
- [surf.pdf](#) The surface tension standard definition
- [thcond.pdf](#) The thermal conductivity standard definition
- [visc.pdf](#) The viscosity standard definition

## Package Content

Name	Description
waterConstants	
simpleWaterConstants	
ConstantPropertyLiquidWater	Water: Simple liquid water medium (incompressible, constant data)
IdealSteam	Water: Steam as ideal gas from NASA source
WaterIF97OnePhase_ph	Water using the IF97 standard, explicit in p and h, and only valid outside the two-phase dome
WaterIF97_ph	Water using the IF97 standard, explicit in p and h
WaterIF97_base	Water: Steam properties as defined by IAPWS/IF97 standard
WaterIF97_fixedregion	Water: Steam properties as defined by IAPWS/IF97 standard
WaterIF97_R1ph	region 1 (liquid) water according to IF97 standard
WaterIF97_R2ph	region 2 (steam) water according to IF97 standard
WaterIF97_R3ph	region 3 water according to IF97 standard
WaterIF97_R4ph	region 4 water according to IF97 standard
WaterIF97_R5ph	region 5 water according to IF97 standard
WaterIF97_R1pT	region 1 (liquid) water according to IF97 standard
WaterIF97_R2pT	region 2 (steam) water according to IF97 standard
IF97_Utils	Low level and utility computation for high accuracy water properties according to the IAPWS/IF97 standard

## Types and constants

```

constant Interfaces.PartialTwoPhaseMedium.FluidConstants[1] waterConstants(
  each chemicalFormula = "H2O",
  each structureFormula="H2O",
  each casRegistryNumber="7732-18-5",
  each iupacName="oxidane",
  each molarMass=0.018015268,
  each criticalTemperature=647.096,
  each criticalPressure=22064.0e3,
  each criticalMolarVolume=1/322.0*0.018015268,
  each normalBoilingPoint=373.124,
  each meltingPoint=273.15,
  each triplePointTemperature=273.16,
  each triplePointPressure=611.657,
  each acentricFactor = 0.344,
  each dipoleMoment = 1.8,
  each hasCriticalData=true);

constant Interfaces.PartialMedium.FluidConstants[1] simpleWaterConstants(
  each chemicalFormula = "H2O",
  each structureFormula="H2O",
  each casRegistryNumber="7732-18-5",

```

---

```

each iupacName="oxidane",
each molarMass=0.018015268);

package StandardWater = WaterIF97_ph
  "Water using the IF97 standard, explicit in p and h. Recommended for most
  applications";

package StandardWaterOnePhase = WaterIF97_pT
  "Water using the IF97 standard, explicit in p and T. Recommended for one-phase
  applications";

package WaterIF97_pT
  "Water using the IF97 standard, explicit in p and T"
  extends WaterIF97_base(
    final ph_explicit=false,
    final dT_explicit=false,
    final pT_explicit=true,
    final smoothModel=true,
    final onePhase=true);
end WaterIF97_pT;

```

---

**Modelica.Media.Water.ConstantPropertyLiquidWater****Water:** Simple liquid water medium (incompressible, constant data)**Information****Modelica.Media.Water.IdealSteam****Water:** Steam as ideal gas from NASA source**Information****Modelica.Media.Water.WaterIF97OnePhase\_ph****Water using the IF97 standard, explicit in p and h, and only valid outside the two-phase dome****Information****Modelica.Media.Water.WaterIF97\_ph****Water using the IF97 standard, explicit in p and h****Information**

## Modelica.Media.Water.WaterIF97\_base

Water: Steam properties as defined by IAPWS/IF97 standard

### Information

This model calculates medium properties for water in the **liquid**, **gas** and **two phase** regions according to the IAPWS/IF97 standard, i.e., the accepted industrial standard and best compromise between accuracy and computation time. For more details see [Modelica.Media.Water.IF97\\_Utilsities](#). Three variable pairs can be the independent variables of the model:

1. Pressure **p** and specific enthalpy **h** are the most natural choice for general applications. This is the recommended choice for most general purpose applications, in particular for power plants.
2. Pressure **p** and temperature **T** are the most natural choice for applications where water is always in the same phase, both for liquid water and steam.
3. Density **d** and temperature **T** are explicit variables of the Helmholtz function in the near-critical region and can be the best choice for applications with super-critical or near-critical states.

The following quantities are always computed:

Variable	Unit	Description
T	K	temperature
u	J/kg	specific internal energy
d	kg/m <sup>3</sup>	density
p	Pa	pressure
h	J/kg	specific enthalpy

In some cases additional medium properties are needed. A component that needs these optional properties has to call one of the functions listed in [Modelica.Media.UsersGuide.MediumUsage.OptionalProperties](#) and in [Modelica.Media.UsersGuide.MediumUsage.TwoPhase](#).

Many further properties can be computed. Using the well-known Bridgman's Tables, all first partial derivatives of the standard thermodynamic variables can be computed easily.

### Package Content

Name	Description
 ThermodynamicState	thermodynamic state
ph_explicit	true if explicit in pressure and specific enthalpy
dT_explicit	true if explicit in density and temperature
pT_explicit	true if explicit in pressure and temperature
 BaseProperties	Base properties of water
 density_ph	Computes density as a function of pressure and specific enthalpy
 temperature_ph	Computes temperature as a function of pressure and specific enthalpy
 temperature_ps	Compute temperature from pressure and specific enthalpy
 density_ps	Computes density as a function of pressure and specific enthalpy
 pressure_dt	Computes pressure as a function of density and temperature
 specificEnthalpy_dt	Computes specific enthalpy as a function of density and temperature
 specificEnthalpy_pt	Computes specific enthalpy as a function of pressure and temperature

(f) specificEnthalpy_ps	Computes specific enthalpy as a function of pressure and temperature
(f) density_pT	Computes density as a function of pressure and temperature
(f) setDewState	set the thermodynamic state on the dew line
(f) setBubbleState	set the thermodynamic state on the bubble line
(f) dynamicViscosity	Dynamic viscosity of water
(f) thermalConductivity	Thermal conductivity of water
(f) surfaceTension	Surface tension in two phase region of water
(f) pressure	return pressure of ideal gas
(f) temperature	return temperature of ideal gas
(f) density	return density of ideal gas
(f) specificEnthalpy	Return specific enthalpy
(f) specificInternalEnergy	Return specific internal energy
(f) specificGibbsEnergy	Return specific Gibbs energy
(f) specificHelmholtzEnergy	Return specific Helmholtz energy
(f) specificEntropy	specific entropy of water
(f) specificHeatCapacityCp	specific heat capacity at constant pressure of water
(f) specificHeatCapacityCv	specific heat capacity at constant volume of water
(f) isentropicExponent	Return isentropic exponent
(f) isothermalCompressibility	Isothermal compressibility of water
(f) isobaricExpansionCoefficient	isobaric expansion coefficient of water
(f) velocityOfSound	
(f) isentropicEnthalpy	compute h(p,s)
(f) density_derh_p	density derivative by specific enthalpy
(f) density_derh_h	density derivative by pressure
(f) bubbleEnthalpy	boiling curve specific enthalpy of water
(f) dewEnthalpy	dew curve specific enthalpy of water
(f) bubbleEntropy	boiling curve specific entropy of water
(f) dewEntropy	dew curve specific entropy of water
(f) bubbleDensity	boiling curve specific density of water
(f) dewDensity	dew curve specific density of water
(f) saturationTemperature	saturation temperature of water
(f) saturationTemperature_derh	derivative of saturation temperature w.r.t. pressure
(f) saturationPressure	saturation pressure of water
(f) dBubbleDensity_dPressure	bubble point density derivative
(f) dDewDensity_dPressure	dew point density derivative
(f) dBubbleEnthalpy_dPressure	bubble point specific enthalpy derivative

## 1576 Modelica.Media.Water.WaterIF97\_base

(f) <a href="#">dDewEnthalpy_dPressure</a>	dew point specific enthalpy derivative
(f) <a href="#">setState_dTX</a>	
(f) <a href="#">setState_phX</a>	
(f) <a href="#">setState_psX</a>	
(f) <a href="#">setState_pTX</a>	
<b>Inherited</b>	
smoothModel	true if the (derived) model should not generate state events
onePhase	true if the (derived) model should never be called with two-phase inputs
FluidLimits	validity limits for fluid model
FluidConstants	extended fluid constants
fluidConstants	constant data for the fluid
SaturationProperties	Saturation properties of two phase medium
FixedPhase	phase of the fluid: 1 for 1-phase, 2 for two-phase, 0 for not known, e.g. interactive use
(f) <a href="#">setSat_T</a>	Return saturation property record from temperature
(f) <a href="#">setSat_p</a>	Return saturation property record from pressure
(f) <a href="#">saturationPressure_sat</a>	Return saturation temperature
(f) <a href="#">saturationTemperature_sat</a>	Return saturation temperature
(f) <a href="#">saturationTemperature_derP_sat</a>	Return derivative of saturation temperature w.r.t. pressure
(f) <a href="#">molarMass</a>	Return the molar mass of the medium
(f) <a href="#">specificEnthalpy_pTX</a>	Return specific enthalpy from pressure, temperature and mass fraction
(f) <a href="#">temperature_phX</a>	Return temperature from p, h, and X or Xi
(f) <a href="#">density_phX</a>	Return density from p, h, and X or Xi
(f) <a href="#">temperature_psX</a>	Return temperature from p, s, and X or Xi
(f) <a href="#">density_psX</a>	Return density from p, s, and X or Xi
(f) <a href="#">specificEnthalpy_psX</a>	Return specific enthalpy from p, s, and X or Xi
(f) <a href="#">setState_pT</a>	Return thermodynamic state from p and T
(f) <a href="#">setState_ph</a>	Return thermodynamic state from p and h
(f) <a href="#">setState_ps</a>	Return thermodynamic state from p and s
(f) <a href="#">setState_dT</a>	Return thermodynamic state from d and T
(f) <a href="#">setState_px</a>	Return thermodynamic state from pressure and vapour quality
(f) <a href="#">setState_Tx</a>	Return thermodynamic state from temperature and vapour quality
(f) <a href="#">vapourQuality</a>	Return vapour quality
mediumName="unusablePartialMedium"	Name of the medium
substanceNames={mediumName}	Names of the mixture substances. Set substanceNames={mediumName} if only one substance.
extraPropertiesNames=fill("", 0)	Names of the additional (extra) transported properties. Set

	extraPropertiesNames=fill("",0) if unused
singleState	= true, if u and d are not a function of pressure
reducedX=true	= true if medium contains the equation sum(X) = 1.0; set reducedX=true if only one substance (see docu for details)
fixedX=false	= true if medium contains the equation X = reference_X
reference_p=101325	Reference pressure of Medium: default 1 atmosphere
reference_T=298.15	Reference temperature of Medium: default 25 deg Celsius
reference_X=if nX == 0 then fill(0, nX) else fill(1/nX, nX)	Default mass fractions of medium
p_default=101325	Default value for pressure of medium (for initialization)
T_default=Modelica.SIunits.Conversions.from_degC(20)	Default value for temperature of medium (for initialization)
h_default=specificEnthalpy_pTX(p_default, T_default, X_default)	Default value for specific enthalpy of medium (for initialization)
X_default=reference_X	Default value for mass fractions of medium (for initialization)
nS=size(substanceNames, 1)	Number of substances
nX=if nS == 1 then 0 else nS	Number of mass fractions (= 0, if only one substance)
nXi=if fixedX then 0 else if reducedX then nS - 1 else nX	Number of structurally independent mass fractions (see docu for details)
nC=size(extraPropertiesNames, 1)	Number of extra (outside of standard mass-balance) transported properties
 BasePropertiesRecord	Variables contained in every instance of BaseProperties
 prandtlNumber	Return the Prandtl number
 heatCapacity_cp	alias for deprecated name
 heatCapacity_cv	alias for deprecated name
 beta	alias for isobaricExpansionCoefficient for user convenience
 kappa	alias of isothermalCompressibility for user convenience
 density_derP_T	Return density derivative wrt pressure at const temperature
 density_derT_p	Return density derivative wrt temperature at constant pressure
 density_derX	Return density derivative wrt mass fraction
 density_pTX	Return density from p, T, and X or Xi
AbsolutePressure	Type for absolute pressure with medium specific attributes
Density	Type for density with medium specific attributes
DynamicViscosity	Type for dynamic viscosity with medium specific attributes
EnthalpyFlowRate	Type for enthalpy flow rate with medium specific attributes
MassFlowRate	Type for mass flow rate with medium specific attributes
MassFraction	Type for mass fraction with medium specific attributes
MoleFraction	Type for mole fraction with medium specific attributes
MolarMass	Type for molar mass with medium specific attributes
MolarVolume	Type for molar volume with medium specific attributes
IsentropicExponent	Type for isentropic exponent with medium specific attributes
SpecificEnergy	Type for specific energy with medium specific attributes
SpecificInternalEnergy	Type for specific internal energy with medium specific attributes

## 1578 Modelica.Media.Water.WaterIF97\_base

SpecificEnthalpy	Type for specific enthalpy with medium specific attributes
SpecificEntropy	Type for specific entropy with medium specific attributes
SpecificHeatCapacity	Type for specific heat capacity with medium specific attributes
SurfaceTension	Type for surface tension with medium specific attributes
Temperature	Type for temperature with medium specific attributes
ThermalConductivity	Type for thermal conductivity with medium specific attributes
PrandtlNumber	Type for Prandtl number with medium specific attributes
VelocityOfSound	Type for velocity of sound with medium specific attributes
ExtraProperty	Type for unspecified, mass-specific property transported by flow
CumulativeExtraProperty	Type for conserved integral of unspecified, mass specific property
ExtraPropertyFlowRate	Type for flow rate of unspecified, mass-specific property
IsobaricExpansionCoefficient	Type for isobaric expansion coefficient with medium specific attributes
DipoleMoment	Type for dipole moment with medium specific attributes
DerDensityByPressure	Type for partial derivative of density with respect to pressure with medium specific attributes
DerDensityByEnthalpy	Type for partial derivative of density with respect to enthalpy with medium specific attributes
DerEnthalpyByPressure	Type for partial derivative of enthalpy with respect to pressure with medium specific attributes
DerDensityByTemperature	Type for partial derivative of density with respect to temperature with medium specific attributes
<input checked="" type="checkbox"/> Choices	Types, constants to define menu choices

### Types and constants

```

constant Boolean ph_explicit
"true if explicit in pressure and specific enthalpy";

constant Boolean dT_explicit "true if explicit in density and temperature";

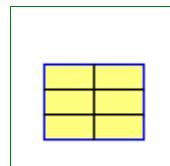
constant Boolean pT_explicit "true if explicit in pressure and temperature";

```

---

## Modelica.Media.Water.WaterIF97\_base.ThermodynamicState

thermodynamic state

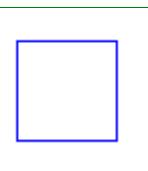


### Modelica definition

```

redeclare record extends ThermodynamicState "thermodynamic state"
  SpecificEnthalpy h "specific enthalpy";
  Density d "density";
  Temperature T "temperature";
  AbsolutePressure p "pressure";
end ThermodynamicState;

```

**Modelica.Media.Water.WaterIF97\_base.BaseProperties**

Base properties of water

**Parameters**

Name	Default	Description
standardOrderComponents	true	if true, last element in components is computed from 1-sum(Xi)
<b>Advanced</b>		
preferredMediumStates	false	= true if StateSelect.prefer shall be used for the independent property variables of the medium

**Modelica.Media.Water.WaterIF97\_base.density\_ph**

Computes density as a function of pressure and specific enthalpy

**Inputs**

Name	Default	Description
p		Pressure [Pa]
h		Specific enthalpy [J/kg]
phase	0	2 for two-phase, 1 for one-phase, 0 if not known

**Outputs**

Name	Description
d	Density [kg/m3]

**Modelica.Media.Water.WaterIF97\_base.temperature\_ph**

Computes temperature as a function of pressure and specific enthalpy

**Inputs**

Name	Default	Description
p		Pressure [Pa]
h		Specific enthalpy [J/kg]
phase	0	2 for two-phase, 1 for one-phase, 0 if not known

**Outputs**

Name	Description
T	Temperature [K]

**Modelica.Media.Water.WaterIF97\_base.temperature\_ps**

Compute temperature from pressure and specific enthalpy

---

## 1580 Modelica.Media.Water.WaterIF97\_base.temperature\_ps

---

### Inputs

Name	Default	Description
p		Pressure [Pa]
s		Specific entropy [J/(kg.K)]
phase	0	2 for two-phase, 1 for one-phase, 0 if not known

### Outputs

Name	Description
T	Temperature [K]

---

## Modelica.Media.Water.WaterIF97\_base.density\_ps

Computes density as a function of pressure and specific enthalpy



### Inputs

Name	Default	Description
p		Pressure [Pa]
s		Specific entropy [J/(kg.K)]
phase	0	2 for two-phase, 1 for one-phase, 0 if not known

### Outputs

Name	Description
d	density [kg/m3]

---

## Modelica.Media.Water.WaterIF97\_base.pressure\_dT

Computes pressure as a function of density and temperature



### Inputs

Name	Default	Description
d		Density [kg/m3]
T		Temperature [K]
phase	0	2 for two-phase, 1 for one-phase, 0 if not known

### Outputs

Name	Description
p	Pressure [Pa]

---

## Modelica.Media.Water.WaterIF97\_base.specificEnthalpy\_dT

Computes specific enthalpy as a function of density and temperature



## Inputs

Name	Default	Description
d		Density [kg/m3]
T		Temperature [K]
phase	0	2 for two-phase, 1 for one-phase, 0 if not known

## Outputs

Name	Description
h	specific enthalpy [J/kg]

## Modelica.Media.Water.WaterIF97\_base.specificEnthalpy\_pT

Computes specific enthalpy as a function of pressure and temperature



## Inputs

Name	Default	Description
p		Pressure [Pa]
T		Temperature [K]
phase	0	2 for two-phase, 1 for one-phase, 0 if not known

## Outputs

Name	Description
h	specific enthalpy [J/kg]

## Modelica.Media.Water.WaterIF97\_base.specificEnthalpy\_ps

Computes specific enthalpy as a function of pressure and temperature



## Inputs

Name	Default	Description
p		Pressure [Pa]
s		Specific entropy [J/(kg.K)]
phase	0	2 for two-phase, 1 for one-phase, 0 if not known

## Outputs

Name	Description
h	specific enthalpy [J/kg]

## Modelica.Media.Water.WaterIF97\_base.density\_pT

Computes density as a function of pressure and temperature



---

## 1582 Modelica.Media.Water.WaterIF97\_base.density\_pT

---

### Inputs

Name	Default	Description
p		Pressure [Pa]
T		Temperature [K]
phase	0	2 for two-phase, 1 for one-phase, 0 if not known

### Outputs

Name	Description
d	Density [kg/m3]

---

## Modelica.Media.Water.WaterIF97\_base.setDewState

set the thermodynamic state on the dew line



### Inputs

Name	Default	Description
sat		saturation point
phase	1	phase: default is one phase

### Outputs

Name	Description
state	complete thermodynamic state info

---

## Modelica.Media.Water.WaterIF97\_base.setBubbleState

set the thermodynamic state on the bubble line



### Inputs

Name	Default	Description
sat		saturation point
phase	1	phase: default is one phase

### Outputs

Name	Description
state	complete thermodynamic state info

---

## Modelica.Media.Water.WaterIF97\_base.dynamicViscosity

Dynamic viscosity of water



### Inputs

Name	Default	Description
state		

**Outputs**

Name	Description
eta	Dynamic viscosity [Pa.s]

**Modelica.Media.Water.WaterIF97\_base.thermalConductivity**

Thermal conductivity of water

**Inputs**

Name	Default	Description
state		

**Outputs**

Name	Description
lambda	Thermal conductivity [W/(m.K)]

**Modelica.Media.Water.WaterIF97\_base.surfaceTension**

Surface tension in two phase region of water

**Inputs**

Name	Default	Description
sat		saturation property record

**Outputs**

Name	Description
sigma	Surface tension sigma in the two phase region [N/m]

**Modelica.Media.Water.WaterIF97\_base.pressure**

return pressure of ideal gas

**Inputs**

Name	Default	Description
state		

**Outputs**

Name	Description
p	Pressure [Pa]

**Modelica.Media.Water.WaterIF97\_base.temperature**

return temperature of ideal gas



---

## 1584 Modelica.Media.Water.WaterIF97\_base.temperature

---

### Inputs

Name	Default	Description
state		

### Outputs

Name	Description
T	Temperature [K]

---

## Modelica.Media.Water.WaterIF97\_base.density



return density of ideal gas

### Inputs

Name	Default	Description
state		

### Outputs

Name	Description
d	Density [kg/m3]

---

## Modelica.Media.Water.WaterIF97\_base.specificEnthalpy



Return specific enthalpy

### Inputs

Name	Default	Description
state		

### Outputs

Name	Description
h	Specific enthalpy [J/kg]

---

## Modelica.Media.Water.WaterIF97\_base.specificInternalEnergy



Return specific internal energy

### Inputs

Name	Default	Description
state		

### Outputs

Name	Description
u	Specific internal energy [J/kg]

**Modelica.Media.Water.WaterIF97\_base.specificGibbsEnergy**

Return specific Gibbs energy

**Inputs**

Name	Default	Description
state		

**Outputs**

Name	Description
g	Specific Gibbs energy [J/kg]

**Modelica.Media.Water.WaterIF97\_base.specificHelmholtzEnergy**

Return specific Helmholtz energy

**Inputs**

Name	Default	Description
state		

**Outputs**

Name	Description
f	Specific Helmholtz energy [J/kg]

**Modelica.Media.Water.WaterIF97\_base.specificEntropy**

specific entropy of water

**Inputs**

Name	Default	Description
state		

**Outputs**

Name	Description
s	Specific entropy [J/(kg.K)]

**Modelica.Media.Water.WaterIF97\_base.specificHeatCapacityCp**

specific heat capacity at constant pressure of water

**Information**

In the two phase region this function returns the interpolated heat capacity between the liquid and vapour state heat capacities.

---

**1586 Modelica.Media.Water.WaterIF97\_base.specificHeatCapacityCp**

---

Error:Found no end-tag in HTML-documentation

**Inputs**

Name	Default	Description
state		

**Outputs**

Name	Description
cp	Specific heat capacity at constant pressure [J/(kg.K)]

**Modelica.Media.Water.WaterIF97\_base.specificHeatCapacityCv**

specific heat capacity at constant volume of water

**Inputs**

Name	Default	Description
state		

**Outputs**

Name	Description
cv	Specific heat capacity at constant volume [J/(kg.K)]

**Modelica.Media.Water.WaterIF97\_base.isentropicExponent**

Return isentropic exponent

**Inputs**

Name	Default	Description
state		

**Outputs**

Name	Description
gamma	Isentropic exponent [1]

**Modelica.Media.Water.WaterIF97\_base.isothalCompressibility**

Isothermal compressibility of water

**Inputs**

Name	Default	Description
state		

**Outputs**

Name	Description
kappa	Isothermal compressibility [1/Pa]

**Modelica.Media.Water.WaterIF97\_base.isobaricExpansionCoefficient**

isobaric expansion coefficient of water

**Inputs**

Name	Default	Description
state		

**Outputs**

Name	Description
beta	Isobaric expansion coefficient [1/K]

**Modelica.Media.Water.WaterIF97\_base.velocityOfSound****Inputs**

Name	Default	Description
state		

**Outputs**

Name	Description
a	Velocity of sound [m/s]

**Modelica.Media.Water.WaterIF97\_base.isentropicEnthalpy**

compute h(p,s)

**Inputs**

Name	Default	Description
p_downstream		downstream pressure [Pa]
refState		reference state for entropy

**Outputs**

Name	Description
h_is	Isentropic enthalpy [J/kg]

**Modelica.Media.Water.WaterIF97\_base.density\_d erh\_p**

density derivative by specific enthalpy

---

**1588 Modelica.Media.Water.WaterIF97\_base.density\_derh\_p**

---

**Inputs**

Name	Default	Description
state		

**Outputs**

Name	Description
ddhp	Density derivative wrt specific enthalpy [kg.s <sup>2</sup> /m <sup>5</sup> ]

---

**Modelica.Media.Water.WaterIF97\_base.density\_derh\_p****density derivative by pressure****Inputs**

Name	Default	Description
state		

**Outputs**

Name	Description
ddph	Density derivative wrt pressure [s <sup>2</sup> /m <sup>2</sup> ]

---

**Modelica.Media.Water.WaterIF97\_base.bubbleEnthalpy****boiling curve specific enthalpy of water****Inputs**

Name	Default	Description
sat		saturation property record

**Outputs**

Name	Description
hl	boiling curve specific enthalpy [J/kg]

---

**Modelica.Media.Water.WaterIF97\_base.dewEnthalpy****dew curve specific enthalpy of water****Inputs**

Name	Default	Description
sat		saturation property record

**Outputs**

Name	Description
hv	dew curve specific enthalpy [J/kg]

**Modelica.Media.Water.WaterIF97\_base.bubbleEntropy**

boiling curve specific entropy of water

**Inputs**

Name	Default	Description
sat		saturation property record

**Outputs**

Name	Description
sl	boiling curve specific entropy [J/(kg.K)]

**Modelica.Media.Water.WaterIF97\_base.dewEntropy**

dew curve specific entropy of water

**Inputs**

Name	Default	Description
sat		saturation property record

**Outputs**

Name	Description
sv	dew curve specific entropy [J/(kg.K)]

**Modelica.Media.Water.WaterIF97\_base.bubbleDensity**

boiling curve specific density of water

**Inputs**

Name	Default	Description
sat		saturation property record

**Outputs**

Name	Description
dl	boiling curve density [kg/m3]

**Modelica.Media.Water.WaterIF97\_base.dewDensity**

dew curve specific density of water

**Inputs**

Name	Default	Description
sat		saturation property record

## 1590 Modelica.Media.Water.WaterIF97\_base.dewDensity

---

### Outputs

Name	Description
dv	dew curve density [kg/m3]

---

### Modelica.Media.Water.WaterIF97\_base.saturationTemperature

saturation temperature of water



### Inputs

Name	Default	Description
p		pressure [Pa]

### Outputs

Name	Description
T	saturation temperature [K]

---

### Modelica.Media.Water.WaterIF97\_base.saturationTemperature\_derp

derivative of saturation temperature w.r.t. pressure



### Inputs

Name	Default	Description
p		pressure [Pa]

### Outputs

Name	Description
dTp	derivative of saturation temperature w.r.t. pressure

---

### Modelica.Media.Water.WaterIF97\_base.saturationPressure

saturation pressure of water



### Inputs

Name	Default	Description
T		temperature [K]

### Outputs

Name	Description
p	saturation pressure [Pa]

---

### Modelica.Media.Water.WaterIF97\_base.dBubbleDensity\_dPressure

bubble point density derivative



**Inputs**

Name	Default	Description
sat		saturation property record

**Outputs**

Name	Description
ddldp	boiling curve density derivative [s <sup>2</sup> /m <sup>2</sup> ]

**Modelica.Media.Water.WaterIF97\_base.dDewDensity\_dPressure**

dew point density derivative

**Inputs**

Name	Default	Description
sat		saturation property record

**Outputs**

Name	Description
ddvdp	saturated steam density derivative [s <sup>2</sup> /m <sup>2</sup> ]

**Modelica.Media.Water.WaterIF97\_base.dBubbleEnthalpy\_dPressure**

bubble point specific enthalpy derivative

**Inputs**

Name	Default	Description
sat		saturation property record

**Outputs**

Name	Description
dhldp	boiling curve specific enthalpy derivative [J.m.s <sup>2</sup> /kg <sup>2</sup> ]

**Modelica.Media.Water.WaterIF97\_base.dDewEnthalpy\_dPressure**

dew point specific enthalpy derivative

**Inputs**

Name	Default	Description
sat		saturation property record

**Outputs**

Name	Description
dhvdp	saturated steam specific enthalpy derivative [J.m.s <sup>2</sup> /kg <sup>2</sup> ]

**Modelica.Media.Water.WaterIF97\_base.setState\_dTX****Inputs**

Name	Default	Description
phase	0	2 for two-phase, 1 for one-phase, 0 if not known
d		density [kg/m <sup>3</sup> ]
T		Temperature [K]
X[:]	reference_X	Mass fractions [kg/kg]

**Outputs**

Name	Description
state	

**Modelica.Media.Water.WaterIF97\_base.setState\_phX****Inputs**

Name	Default	Description
phase	0	2 for two-phase, 1 for one-phase, 0 if not known
p		Pressure [Pa]
h		Specific enthalpy [J/kg]
X[:]	reference_X	Mass fractions [kg/kg]

**Outputs**

Name	Description
state	

**Modelica.Media.Water.WaterIF97\_base.setState\_psX****Inputs**

Name	Default	Description
phase	0	2 for two-phase, 1 for one-phase, 0 if not known
p		Pressure [Pa]
s		Specific entropy [J/(kg.K)]
X[:]	reference_X	Mass fractions [kg/kg]

**Outputs**

Name	Description
state	

**Modelica.Media.Water.WaterIF97\_base.setState\_pTX****Inputs**

Name	Default	Description
phase	0	2 for two-phase, 1 for one-phase, 0 if not known
p		Pressure [Pa]
T		Temperature [K]
X[:]	reference_X	Mass fractions [kg/kg]

**Outputs**

Name	Description
state	

**Modelica.Media.Water.WaterIF97\_fixedregion**

Water: Steam properties as defined by IAPWS/IF97 standard

**Information**

This model calculates medium properties for water in the **liquid**, **gas** and **two phase** regions according to the IAPWS/IF97 standard, i.e., the accepted industrial standard and best compromise between accuracy and computation time. For more details see [Modelica.Media.Water.IF97\\_Utils](#). Three variable pairs can be the independent variables of the model:

1. Pressure **p** and specific enthalpy **h** are the most natural choice for general applications. This is the recommended choice for most general purpose applications, in particular for power plants.
2. Pressure **p** and temperature **T** are the most natural choice for applications where water is always in the same phase, both for liquid water and steam.
3. Density **d** and temperature **T** are explicit variables of the Helmholtz function in the near-critical region and can be the best choice for applications with super-critical or near-critical states.

The following quantities are always computed:

Variable	Unit	Description
T	K	temperature
u	J/kg	specific internal energy
d	kg/m <sup>3</sup>	density
p	Pa	pressure
h	J/kg	specific enthalpy

In some cases additional medium properties are needed. A component that needs these optional properties has to call one of the functions listed in [Modelica.Media.UsersGuide.MediumUsage.OptionalProperties](#) and in [Modelica.Media.UsersGuide.MediumUsage.TwoPhase](#).

Many further properties can be computed. Using the well-known Bridgman's Tables, all first partial derivatives of the standard thermodynamic variables can be computed easily.

**Package Content**

Name	Description
 ThermodynamicState	thermodynamic state
Region	region of IF97, if known
ph_explicit	true if explicit in pressure and specific enthalpy

---

**1594 Modelica.Media.Water.WaterIF97\_fixedregion**


---

<code>dT_explicit</code>	true if explicit in density and temperature
<code>pT_explicit</code>	true if explicit in pressure and temperature
<input checked="" type="checkbox"/> <code>BaseProperties</code>	Base properties of water
(f) <code>density_ph</code>	Computes density as a function of pressure and specific enthalpy
(f) <code>temperature_ph</code>	Computes temperature as a function of pressure and specific enthalpy
(f) <code>temperature_ps</code>	Compute temperature from pressure and specific enthalpy
(f) <code>density_ps</code>	Computes density as a function of pressure and specific enthalpy
(f) <code>pressure_dT</code>	Computes pressure as a function of density and temperature
(f) <code>specificEnthalpy_dT</code>	Computes specific enthalpy as a function of density and temperature
(f) <code>specificEnthalpy_pT</code>	Computes specific enthalpy as a function of pressure and temperature
(f) <code>specificEnthalpy_ps</code>	Computes specific enthalpy as a function of pressure and temperature
(f) <code>density_pT</code>	Computes density as a function of pressure and temperature
(f) <code>setDewState</code>	set the thermodynamic state on the dew line
(f) <code>setBubbleState</code>	set the thermodynamic state on the bubble line
(f) <code>dynamicViscosity</code>	Dynamic viscosity of water
(f) <code>thermalConductivity</code>	Thermal conductivity of water
(f) <code>surfaceTension</code>	Surface tension in two phase region of water
(f) <code>pressure</code>	return pressure of ideal gas
(f) <code>temperature</code>	return temperature of ideal gas
(f) <code>density</code>	return density of ideal gas
(f) <code>specificEnthalpy</code>	Return specific enthalpy
(f) <code>specificInternalEnergy</code>	Return specific internal energy
(f) <code>specificGibbsEnergy</code>	Return specific Gibbs energy
(f) <code>specificHelmholtzEnergy</code>	Return specific Helmholtz energy
(f) <code>specificEntropy</code>	specific entropy of water
(f) <code>specificHeatCapacityCp</code>	specific heat capacity at constant pressure of water
(f) <code>specificHeatCapacityCv</code>	specific heat capacity at constant volume of water
(f) <code>isentropicExponent</code>	Return isentropic exponent
(f) <code>isothermalCompressibility</code>	Isothermal compressibility of water
(f) <code>isobaricExpansionCoefficient</code>	isobaric expansion coefficient of water
(f) <code>velocityOfSound</code>	
(f) <code>isentropicEnthalpy</code>	compute h(s,p)
(f) <code>density_derh_p</code>	density derivative by specific enthalpy
(f) <code>density_derh_h</code>	density derivative by pressure

(f) bubbleEnthalpy	boiling curve specific enthalpy of water
(f) dewEnthalpy	dew curve specific enthalpy of water
(f) bubbleEntropy	boiling curve specific entropy of water
(f) dewEntropy	dew curve specific entropy of water
(f) bubbleDensity	boiling curve specific density of water
(f) dewDensity	dew curve specific density of water
(f) saturationTemperature	saturation temperature of water
(f) saturationTemperature_derP	derivative of saturation temperature w.r.t. pressure
(f) saturationPressure	saturation pressure of water
(f) dBubbleDensity_dPressure	bubble point density derivative
(f) dDewDensity_dPressure	dew point density derivative
(f) dBubbleEnthalpy_dPressure	bubble point specific enthalpy derivative
(f) dDewEnthalpy_dPressure	dew point specific enthalpy derivative
(f) setState_dTX	
(f) setState_phX	
(f) setState_psX	
(f) setState_pTX	

**Inherited**

smoothModel	true if the (derived) model should not generate state events
onePhase	true if the (derived) model should never be called with two-phase inputs
FluidLimits	validity limits for fluid model
FluidConstants	extended fluid constants
fluidConstants	constant data for the fluid
SaturationProperties	Saturation properties of two phase medium
FixedPhase	phase of the fluid: 1 for 1-phase, 2 for two-phase, 0 for not known, e.g. interactive use
(f) setSat_T	Return saturation property record from temperature
(f) setSat_p	Return saturation property record from pressure
(f) saturationPressure_sat	Return saturation temperature
(f) saturationTemperature_sat	Return saturation temperature
(f) saturationTemperature_derP_sat	Return derivative of saturation temperature w.r.t. pressure
(f) molarMass	Return the molar mass of the medium
(f) specificEnthalpy_pTX	Return specific enthalpy from pressure, temperature and mass fraction
(f) temperature_phX	Return temperature from p, h, and X or Xi
(f) density_phX	Return density from p, h, and X or Xi
(f) temperature_psX	Return temperature from p, s, and X or Xi
(f) density_psX	Return density from p, s, and X or Xi

## 1596 Modelica.Media.Water.WaterIF97\_fixedregion

 <a href="#">specificEnthalpy_psX</a>	Return specific enthalpy from p, s, and X or Xi
 <a href="#">setState_pT</a>	Return thermodynamic state from p and T
 <a href="#">setState_ph</a>	Return thermodynamic state from p and h
 <a href="#">setState_ps</a>	Return thermodynamic state from p and s
 <a href="#">setState_dT</a>	Return thermodynamic state from d and T
 <a href="#">setState_px</a>	Return thermodynamic state from pressure and vapour quality
 <a href="#">setState_Tx</a>	Return thermodynamic state from temperature and vapour quality
 <a href="#">vapourQuality</a>	Return vapour quality
mediumName="unusablePartialMedium"	Name of the medium
substanceNames={mediumName}	Names of the mixture substances. Set substanceNames={mediumName} if only one substance.
extraPropertiesNames=fill("", 0)	Names of the additional (extra) transported properties. Set extraPropertiesNames=fill("",0) if unused
singleState	= true, if u and d are not a function of pressure
reducedX=true	= true if medium contains the equation sum(X) = 1.0; set reducedX=true if only one substance (see docu for details)
fixedX=false	= true if medium contains the equation X = reference_X
reference_p=101325	Reference pressure of Medium: default 1 atmosphere
reference_T=298.15	Reference temperature of Medium: default 25 deg Celsius
reference_X=if nX == 0 then fill(0, nX) else fill(1/nX, nX)	Default mass fractions of medium
p_default=101325	Default value for pressure of medium (for initialization)
T_default=Modelica.Slunits.Conversions.from_degC(20)	Default value for temperature of medium (for initialization)
h_default=specificEnthalpy_pTX(p_default, T_default, X_default)	Default value for specific enthalpy of medium (for initialization)
X_default=reference_X	Default value for mass fractions of medium (for initialization)
nS=size(substanceNames, 1)	Number of substances
nX=if nS == 1 then 0 else nS	Number of mass fractions (= 0, if only one substance)
nXi=if fixedX then 0 else if reducedX then nS - 1 else nX	Number of structurally independent mass fractions (see docu for details)
nC=size(extraPropertiesNames, 1)	Number of extra (outside of standard mass-balance) transported properties
 <a href="#">BasePropertiesRecord</a>	Variables contained in every instance of BaseProperties
 <a href="#">prandtlNumber</a>	Return the Prandtl number
 <a href="#">heatCapacity_cp</a>	alias for deprecated name
 <a href="#">heatCapacity_cv</a>	alias for deprecated name
 <a href="#">beta</a>	alias for isobaricExpansionCoefficient for user convenience
 <a href="#">kappa</a>	alias of isothermalCompressibility for user convenience
 <a href="#">density_derp_T</a>	Return density derivative wrt pressure at const temperature
 <a href="#">density_derT_p</a>	Return density derivative wrt temperature at constant pressure
 <a href="#">density_derX</a>	Return density derivative wrt mass fraction

 density_pTX	Return density from p, T, and X or Xi
AbsolutePressure	Type for absolute pressure with medium specific attributes
Density	Type for density with medium specific attributes
DynamicViscosity	Type for dynamic viscosity with medium specific attributes
EnthalpyFlowRate	Type for enthalpy flow rate with medium specific attributes
MassFlowRate	Type for mass flow rate with medium specific attributes
MassFraction	Type for mass fraction with medium specific attributes
MoleFraction	Type for mole fraction with medium specific attributes
MolarMass	Type for molar mass with medium specific attributes
MolarVolume	Type for molar volume with medium specific attributes
IsentropicExponent	Type for isentropic exponent with medium specific attributes
SpecificEnergy	Type for specific energy with medium specific attributes
SpecificInternalEnergy	Type for specific internal energy with medium specific attributes
SpecificEnthalpy	Type for specific enthalpy with medium specific attributes
SpecificEntropy	Type for specific entropy with medium specific attributes
SpecificHeatCapacity	Type for specific heat capacity with medium specific attributes
SurfaceTension	Type for surface tension with medium specific attributes
Temperature	Type for temperature with medium specific attributes
ThermalConductivity	Type for thermal conductivity with medium specific attributes
PrandtlNumber	Type for Prandtl number with medium specific attributes
VelocityOfSound	Type for velocity of sound with medium specific attributes
ExtraProperty	Type for unspecified, mass-specific property transported by flow
CumulativeExtraProperty	Type for conserved integral of unspecified, mass specific property
ExtraPropertyFlowRate	Type for flow rate of unspecified, mass-specific property
IsobaricExpansionCoefficient	Type for isobaric expansion coefficient with medium specific attributes
DipoleMoment	Type for dipole moment with medium specific attributes
DerDensityByPressure	Type for partial derivative of density with respect to pressure with medium specific attributes
DerDensityByEnthalpy	Type for partial derivative of density with respect to enthalpy with medium specific attributes
DerEnthalpyByPressure	Type for partial derivative of enthalpy with respect to pressure with medium specific attributes
DerDensityByTemperature	Type for partial derivative of density with respect to temperature with medium specific attributes
 Choices	Types, constants to define menu choices

## Types and constants

```

constant Integer Region "region of IF97, if known";

constant Boolean ph_explicit
"true if explicit in pressure and specific enthalpy";

constant Boolean dT_explicit "true if explicit in density and temperature";

```

## 1598 Modelica.Media.Water.WaterIF97\_fixedregion

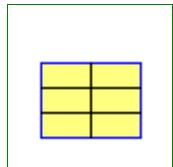
---

```
constant Boolean pT_explicit "true if explicit in pressure and temperature";
```

---

### Modelica.Media.Water.WaterIF97\_fixedregion.ThermodynamicState

thermodynamic state



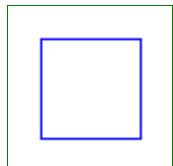
#### Modelica definition

```
redeclare record extends ThermodynamicState "thermodynamic state"
  SpecificEnthalpy h "specific enthalpy";
  Density d "density";
  Temperature T "temperature";
  AbsolutePressure p "pressure";
end ThermodynamicState;
```

---

### Modelica.Media.Water.WaterIF97\_fixedregion.BaseProperties

Base properties of water



#### Parameters

Name	Default	Description
standardOrderComponents	true	if true, last element in components is computed from 1-sum(Xi)
<b>Advanced</b>		
preferredMediumStates	false	= true if StateSelect.prefer shall be used for the independent property variables of the medium

---

### Modelica.Media.Water.WaterIF97\_fixedregion.density\_ph

Computes density as a function of pressure and specific enthalpy



#### Inputs

Name	Default	Description
p		Pressure [Pa]
h		Specific enthalpy [J/kg]
phase	0	2 for two-phase, 1 for one-phase, 0 if not known
region	0	if 0, region is unknown, otherwise known and this input

#### Outputs

Name	Description
d	Density [kg/m3]

---

### Modelica.Media.Water.WaterIF97\_fixedregion.temperature\_ph

Computes temperature as a function of pressure and specific enthalpy



## Inputs

Name	Default	Description
p		Pressure [Pa]
h		Specific enthalpy [J/kg]
phase	0	2 for two-phase, 1 for one-phase, 0 if not known
region	0	if 0, region is unknown, otherwise known and this input

## Outputs

Name	Description
T	Temperature [K]

## Modelica.Media.Water.WaterIF97\_fixedregion.temperature\_ps

Compute temperature from pressure and specific enthalpy



## Inputs

Name	Default	Description
p		Pressure [Pa]
s		Specific entropy [J/(kg.K)]
phase	0	2 for two-phase, 1 for one-phase, 0 if not known
region	0	if 0, region is unknown, otherwise known and this input

## Outputs

Name	Description
T	Temperature [K]

## Modelica.Media.Water.WaterIF97\_fixedregion.density\_ps

Computes density as a function of pressure and specific enthalpy



## Inputs

Name	Default	Description
p		Pressure [Pa]
s		Specific entropy [J/(kg.K)]
phase	0	2 for two-phase, 1 for one-phase, 0 if not known
region	0	if 0, region is unknown, otherwise known and this input

## Outputs

Name	Description
d	density [kg/m3]

## Modelica.Media.Water.WaterIF97\_fixedregion.pressure\_dT

Computes pressure as a function of density and temperature



---

## 1600 Modelica.Media.Water.WaterIF97\_fixedregion.pressure\_dT

---

### Inputs

Name	Default	Description
d		Density [kg/m3]
T		Temperature [K]
phase	0	2 for two-phase, 1 for one-phase, 0 if not known
region	0	if 0, region is unknown, otherwise known and this input

### Outputs

Name	Description
p	Pressure [Pa]

---

## Modelica.Media.Water.WaterIF97\_fixedregion.specificEnthalpy\_dT

Computes specific enthalpy as a function of density and temperature



### Inputs

Name	Default	Description
d		Density [kg/m3]
T		Temperature [K]
phase	0	2 for two-phase, 1 for one-phase, 0 if not known
region	0	if 0, region is unknown, otherwise known and this input

### Outputs

Name	Description
h	specific enthalpy [J/kg]

---

## Modelica.Media.Water.WaterIF97\_fixedregion.specificEnthalpy\_pT

Computes specific enthalpy as a function of pressure and temperature



### Inputs

Name	Default	Description
p		Pressure [Pa]
T		Temperature [K]
phase	0	2 for two-phase, 1 for one-phase, 0 if not known
region	0	if 0, region is unknown, otherwise known and this input

### Outputs

Name	Description
h	specific enthalpy [J/kg]

---

## Modelica.Media.Water.WaterIF97\_fixedregion.specificEnthalpy\_ps

Computes specific enthalpy as a function of pressure and temperature



## Inputs

Name	Default	Description
p		Pressure [Pa]
s		Specific entropy [J/(kg.K)]
phase	0	2 for two-phase, 1 for one-phase, 0 if not known
region	0	if 0, region is unknown, otherwise known and this input

## Outputs

Name	Description
h	specific enthalpy [J/kg]

## Modelica.Media.Water.WaterIF97\_fixedregion.density\_pT

Computes density as a function of pressure and temperature



## Inputs

Name	Default	Description
p		Pressure [Pa]
T		Temperature [K]
phase	0	2 for two-phase, 1 for one-phase, 0 if not known
region	0	if 0, region is unknown, otherwise known and this input

## Outputs

Name	Description
d	Density [kg/m3]

## Modelica.Media.Water.WaterIF97\_fixedregion.setDewState

set the thermodynamic state on the dew line



## Inputs

Name	Default	Description
sat		saturation point
phase	1	phase: default is one phase

## Outputs

Name	Description
state	complete thermodynamic state info

## Modelica.Media.Water.WaterIF97\_fixedregion.setBubbleState

set the thermodynamic state on the bubble line



---

**1602 Modelica.Media.Water.WaterIF97\_fixedregion.setBubbleState**

---

**Inputs**

Name	Default	Description
sat		saturation point
phase	1	phase: default is one phase

**Outputs**

Name	Description
state	complete thermodynamic state info

---

**Modelica.Media.Water.WaterIF97\_fixedregion.dynamicViscosity****Dynamic viscosity of water****Inputs**

Name	Default	Description
state		

**Outputs**

Name	Description
eta	Dynamic viscosity [Pa.s]

---

**Modelica.Media.Water.WaterIF97\_fixedregion.thermalConductivity****Thermal conductivity of water****Inputs**

Name	Default	Description
state		

**Outputs**

Name	Description
lambda	Thermal conductivity [W/(m.K)]

---

**Modelica.Media.Water.WaterIF97\_fixedregion.surfaceTension****Surface tension in two phase region of water****Inputs**

Name	Default	Description
sat		saturation property record

**Outputs**

Name	Description

sigma	Surface tension sigma in the two phase region [N/m]
-------	---

---

**Modelica.Media.Water.WaterIF97\_fixedregion.pressure**

return pressure of ideal gas

**Inputs**

Name	Default	Description
state		

**Outputs**

Name	Description
p	Pressure [Pa]

---

**Modelica.Media.Water.WaterIF97\_fixedregion.temperature**

return temperature of ideal gas

**Inputs**

Name	Default	Description
state		

**Outputs**

Name	Description
T	Temperature [K]

---

**Modelica.Media.Water.WaterIF97\_fixedregion.density**

return density of ideal gas

**Inputs**

Name	Default	Description
state		

**Outputs**

Name	Description
d	Density [kg/m3]

---

**Modelica.Media.Water.WaterIF97\_fixedregion.specificEnthalpy**

Return specific enthalpy



---

## 1604 Modelica.Media.Water.WaterIF97\_fixedregion.specificEnthalpy

---

### Inputs

Name	Default	Description
state		

### Outputs

Name	Description
h	Specific enthalpy [J/kg]

---

## Modelica.Media.Water.WaterIF97\_fixedregion.specificInternalEnergy



Return specific internal energy

### Inputs

Name	Default	Description
state		

### Outputs

Name	Description
u	Specific internal energy [J/kg]

---

## Modelica.Media.Water.WaterIF97\_fixedregion.specificGibbsEnergy



Return specific Gibbs energy

### Inputs

Name	Default	Description
state		

### Outputs

Name	Description
g	Specific Gibbs energy [J/kg]

---

## Modelica.Media.Water.WaterIF97\_fixedregion.specificHelmholtzEnergy



Return specific Helmholtz energy

### Inputs

Name	Default	Description
state		

### Outputs

Name	Description
f	Specific Helmholtz energy [J/kg]

**Modelica.Media.Water.WaterIF97\_fixedregion.specificEntropy**

specific entropy of water

**Inputs**

Name	Default	Description
state		

**Outputs**

Name	Description
s	Specific entropy [J/(kg.K)]

**Modelica.Media.Water.WaterIF97\_fixedregion.specificHeatCapacityCp**

specific heat capacity at constant pressure of water

**Inputs**

Name	Default	Description
state		

**Outputs**

Name	Description
cp	Specific heat capacity at constant pressure [J/(kg.K)]

**Modelica.Media.Water.WaterIF97\_fixedregion.specificHeatCapacityCv**

specific heat capacity at constant volume of water

**Inputs**

Name	Default	Description
state		

**Outputs**

Name	Description
cv	Specific heat capacity at constant volume [J/(kg.K)]

**Modelica.Media.Water.WaterIF97\_fixedregion.isentropicExponent**

Return isentropic exponent

**Inputs**

Name	Default	Description
state		

## 1606 Modelica.Media.Water.WaterIF97\_fixedregion.isentropicExponent

---

### Outputs

Name	Description
gamma	Isentropic exponent [1]

---

## Modelica.Media.Water.WaterIF97\_fixedregion.isoCompressibility

Isothermal compressibility of water



### Inputs

Name	Default	Description
state		

### Outputs

Name	Description
kappa	Isothermal compressibility [1/Pa]

---

## Modelica.Media.Water.WaterIF97\_fixedregion.isoExpansionCoefficient

isobaric expansion coefficient of water



### Inputs

Name	Default	Description
state		

### Outputs

Name	Description
beta	Isobaric expansion coefficient [1/K]

---

## Modelica.Media.Water.WaterIF97\_fixedregion.velocityOfSound



### Inputs

Name	Default	Description
state		

### Outputs

Name	Description
a	Velocity of sound [m/s]

---

## Modelica.Media.Water.WaterIF97\_fixedregion.isentropicEnthalpy

compute h(s,p)



**Inputs**

Name	Default	Description
p_downstream		downstream pressure [Pa]
refState		reference state for entropy

**Outputs**

Name	Description
h_is	Isentropic enthalpy [J/kg]

**Modelica.Media.Water.WaterIF97\_fixedregion.density\_derh\_p****density derivative by specific enthalpy****Inputs**

Name	Default	Description
state		

**Outputs**

Name	Description
ddhp	Density derivative wrt specific enthalpy [kg.s <sup>2</sup> /m <sup>5</sup> ]

**Modelica.Media.Water.WaterIF97\_fixedregion.density\_derh\_p****density derivative by pressure****Inputs**

Name	Default	Description
state		

**Outputs**

Name	Description
ddph	Density derivative wrt pressure [s <sup>2</sup> /m <sup>2</sup> ]

**Modelica.Media.Water.WaterIF97\_fixedregion.bubbleEnthalpy****boiling curve specific enthalpy of water****Inputs**

Name	Default	Description
sat		saturation property record

**Outputs**

Name	Description

---

## 1608 Modelica.Media.Water.WaterIF97\_fixedregion.bubbleEnthalpy

---

hl	boiling curve specific enthalpy [J/kg]
----	--

---

### Modelica.Media.Water.WaterIF97\_fixedregion.dewEnthalpy

dew curve specific enthalpy of water



#### Inputs

Name	Default	Description
sat		saturation property record

#### Outputs

Name	Description
hv	dew curve specific enthalpy [J/kg]

---

### Modelica.Media.Water.WaterIF97\_fixedregion.bubbleEntropy

boiling curve specific entropy of water



#### Inputs

Name	Default	Description
sat		saturation property record

#### Outputs

Name	Description
sl	boiling curve specific entropy [J/(kg.K)]

---

### Modelica.Media.Water.WaterIF97\_fixedregion.dewEntropy

dew curve specific entropy of water



#### Inputs

Name	Default	Description
sat		saturation property record

#### Outputs

Name	Description
sv	dew curve specific entropy [J/(kg.K)]

---

### Modelica.Media.Water.WaterIF97\_fixedregion.bubbleDensity

boiling curve specific density of water



**Inputs**

Name	Default	Description
sat		saturation property record

**Outputs**

Name	Description
dl	boiling curve density [kg/m3]

**Modelica.Media.Water.WaterIF97\_fixedregion.dewDensity**

dew curve specific density of water

**Inputs**

Name	Default	Description
sat		saturation property record

**Outputs**

Name	Description
dv	dew curve density [kg/m3]

**Modelica.Media.Water.WaterIF97\_fixedregion.saturationTemperature**

saturation temperature of water

**Inputs**

Name	Default	Description
p		pressure [Pa]

**Outputs**

Name	Description
T	saturation temperature [K]

**Modelica.Media.Water.WaterIF97\_fixedregion.saturationTemperature\_derp**

derivative of saturation temperature w.r.t. pressure

**Inputs**

Name	Default	Description
p		pressure [Pa]

**Outputs**

Name	Description
dTp	derivative of saturation temperature w.r.t. pressure

**Modelica.Media.Water.WaterIF97\_fixedregion.saturationPressure****saturation pressure of water****Inputs**

Name	Default	Description
T		temperature [K]

**Outputs**

Name	Description
p	saturation pressure [Pa]

---

**Modelica.Media.Water.WaterIF97\_fixedregion.dBubbleDensity\_dPressure****bubble point density derivative****Inputs**

Name	Default	Description
sat		saturation property record

**Outputs**

Name	Description
ddldp	boiling curve density derivative [s <sup>2</sup> /m <sup>2</sup> ]

---

**Modelica.Media.Water.WaterIF97\_fixedregion.dDewDensity\_dPressure****dew point density derivative****Inputs**

Name	Default	Description
sat		saturation property record

**Outputs**

Name	Description
ddvdp	saturated steam density derivative [s <sup>2</sup> /m <sup>2</sup> ]

---

**Modelica.Media.Water.WaterIF97\_fixedregion.dBubbleEnthalpy\_dPressure****bubble point specific enthalpy derivative****Inputs**

Name	Default	Description
sat		saturation property record

## Outputs

Name	Description
dhldp	boiling curve specific enthalpy derivative [J.m.s2/kg2]

---

## Modelica.Media.Water.WaterIF97\_fixedregion.dDewEnthalpy\_dPressure

dew point specific enthalpy derivative



## Inputs

Name	Default	Description
sat		saturation property record

## Outputs

Name	Description
dhvdp	saturated steam specific enthalpy derivative [J.m.s2/kg2]

---

## Modelica.Media.Water.WaterIF97\_fixedregion.setState\_dTX



## Inputs

Name	Default	Description
region	0	if 0, region is unknown, otherwise known and this input
phase	0	2 for two-phase, 1 for one-phase, 0 if not known
d		density [kg/m3]
T		Temperature [K]
X[:]	reference_X	Mass fractions [kg/kg]

## Outputs

Name	Description
state	

---

## Modelica.Media.Water.WaterIF97\_fixedregion.setState\_phX



## Inputs

Name	Default	Description
region	0	if 0, region is unknown, otherwise known and this input
phase	0	2 for two-phase, 1 for one-phase, 0 if not known
p		Pressure [Pa]
h		Specific enthalpy [J/kg]
X[:]	reference_X	Mass fractions [kg/kg]

## Outputs

Name	Description

## 1612 Modelica.Media.Water.WaterIF97\_fixedregion.setState\_phX

---

state	
-------	--

---

## Modelica.Media.Water.WaterIF97\_fixedregion.setState\_psX



### Inputs

Name	Default	Description
region	0	if 0, region is unknown, otherwise known and this input
phase	0	2 for two-phase, 1 for one-phase, 0 if not known
p		Pressure [Pa]
s		Specific entropy [J/(kg.K)]
X[:]	reference_X	Mass fractions [kg/kg]

### Outputs

Name	Description
state	

---

## Modelica.Media.Water.WaterIF97\_fixedregion.setState\_pTX



### Inputs

Name	Default	Description
region	0	if 0, region is unknown, otherwise known and this input
phase	0	2 for two-phase, 1 for one-phase, 0 if not known
p		Pressure [Pa]
T		Temperature [K]
X[:]	reference_X	Mass fractions [kg/kg]

### Outputs

Name	Description
state	

---

## Modelica.Media.Water.WaterIF97\_R1ph

region 1 (liquid) water according to IF97 standard

### Information

---

## Modelica.Media.Water.WaterIF97\_R2ph

region 2 (steam) water according to IF97 standard

### Information

**Modelica.Media.Water.WaterIF97\_R3ph**

region 3 water according to IF97 standard

**Information**

---

**Modelica.Media.Water.WaterIF97\_R4ph**

region 4 water according to IF97 standard

**Information**

---

**Modelica.Media.Water.WaterIF97\_R5ph**

region 5 water according to IF97 standard

**Information**

---

**Modelica.Media.Water.WaterIF97\_R1pT**

region 1 (liquid) water according to IF97 standard

**Information**

---

**Modelica.Media.Water.WaterIF97\_R2pT**

region 2 (steam) water according to IF97 standard

**Information**

---

**Modelica.Media.Water.IF97\_Utilsities**

Low level and utility computation for high accuracy water properties according to the IAPWS/IF97 standard

**Information**

**Package description:**

This package provides high accuracy physical properties for water according to the IAPWS/IF97 standard. It has been part of the ThermoFluid Modelica library and been extended, reorganized and documented to become part of the Modelica Standard library.

An important feature that distinguishes this implementation of the IF97 steam property standard is that this implementation has been explicitly designed to work well in dynamic simulations. Computational performance has been of high importance. This means that there often exist several ways to get the same result from different functions if one of the functions is called often but can be optimized for that purpose.

The original documentation of the IAPWS/IF97 steam properties can freely be distributed with computer implementations, so for curious minds the complete standard documentation is provided with the Modelica properties library. The following documents are included (in directory Modelica\help\Documentation\IF97documentation):

- [IF97.pdf](#) The standards document for the main part of the IF97.
- [Back3.pdf](#) The backwards equations for region 3.
- [crits.pdf](#) The critical point data.
- [meltsub.pdf](#) The melting- and sublimation line formulation (in IF97\_Utilsies.BaselIF97.IceBoundaries)
- [surf.pdf](#) The surface tension standard definition
- [thcond.pdf](#) The thermal conductivity standard definition
- [visc.pdf](#) The viscosity standard definition

### Package contents

- Package **BaselF97** contains the implementation of the IAPWS-IF97 as described in [IF97.pdf](#). The explicit backwards equations for region 3 from [Back3.pdf](#) are implemented as initial values for an inverse iteration of the exact function in IF97 for the input pairs (p,h) and (p,s). The low-level functions in BaselF97 are not needed for standard simulation usage, but can be useful for experts and some special purposes.
- Function **water\_ph** returns all properties needed for a dynamic control volume model and properties of general interest using pressure p and specific entropy enthalpy h as dynamic states in the record ThermoProperties\_ph.
- Function **water\_ps** returns all properties needed for a dynamic control volume model and properties of general interest using pressure p and specific entropy s as dynamic states in the record ThermoProperties\_ps.
- Function **water\_dT** returns all properties needed for a dynamic control volume model and properties of general interest using density d and temperature T as dynamic states in the record ThermoProperties\_dT.
- Function **water\_pT** returns all properties needed for a dynamic control volume model and properties of general interest using pressure p and temperature T as dynamic states in the record ThermoProperties\_pT. Due to the coupling of pressure and temperature in the two-phase region, this model can obviously only be used for one-phase models or models treating both phases independently.
- Function **hl\_p** computes the liquid specific enthalpy as a function of pressure. For overcritical pressures, the critical specific enthalpy is returned
- Function **hv\_p** computes the vapour specific enthalpy as a function of pressure. For overcritical pressures, the critical specific enthalpy is returned
- Function **sl\_p** computes the liquid specific entropy as a function of pressure. For overcritical pressures, the critical specific entropy is returned
- Function **sv\_p** computes the vapour specific entropy as a function of pressure. For overcritical pressures, the critical specific entropy is returned
- Function **rhol\_T** computes the liquid density as a function of temperature. For overcritical temperatures, the critical density is returned
- Function **rhov\_T** computes the vapour density as a function of temperature. For overcritical temperatures, the critical density is returned
- Function **dynamicViscosity** computes the dynamic viscosity as a function of density and temperature.
- Function **thermalConductivity** computes the thermal conductivity as a function of density, temperature and pressure. **Important note:** Obviously only two of the three inputs are really needed, but using three inputs speeds up the computation and the three variables are known in most models anyways. The inputs d,T and p have to be consistent.
- Function **surfaceTension** computes the surface tension between vapour and liquid water as a function of temperature.
- Function **isentropicEnthalpy** computes the specific enthalpy h(p,s,phase) in all regions. The phase input is needed due to discontinuous derivatives at the phase boundary.
- Function **dynamicIsentropicEnthalpy** computes the specific enthalpy h(p,s,,dguess,Tguess,phase) in all regions. The phase input is needed due to discontinuous derivatives at the phase boundary.

Tguess and dguess are initial guess values for the density and temperature consistent with p and s. This function should be preferred in dynamic simulations where good guesses are often available.

### Version Info and Revision history

- First implemented: *July, 2000* by [Hubertus Tummescheit](#) for the ThermoFluid Library with help from Jonas Eborn and Falko Jens Wagner
- Code reorganization, enhanced documentation, additional functions: *December, 2002* by [Hubertus Tummescheit](#) and moved to Modelica properties library.

*Author: Hubertus Tummescheit,*

*Modelon AB*

*Ideon Science Park*

*SE-22370 Lund, Sweden*

*email: hubertus@modelon.se*

### Package Content

Name	Description
 <a href="#">BaselF97</a>	Modelica Physical Property Model: the new industrial formulation IAPWS-IF97
 <a href="#">iter</a>	
 <a href="#">waterBaseProp_ph</a>	intermediate property record for water
 <a href="#">waterBaseProp_ps</a>	intermediate property record for water
 <a href="#">rho_props_ps</a>	density as function of pressure and specific entropy
 <a href="#">rho_ps</a>	density as function of pressure and specific entropy
 <a href="#">T_props_ps</a>	temperature as function of pressure and specific entropy
 <a href="#">T_ps</a>	temperature as function of pressure and specific entropy
 <a href="#">h_props_ps</a>	specific enthalpy as function of pressure and temperature
 <a href="#">h_ps</a>	specific enthalpy as function of pressure and temperature
 <a href="#">phase_ps</a>	phase as a function of pressure and specific entropy
 <a href="#">phase_ph</a>	phase as a function of pressure and specific enthalpy
 <a href="#">phase_dT</a>	phase as a function of pressure and temperature
 <a href="#">rho_props_ph</a>	density as function of pressure and specific enthalpy
 <a href="#">rho_ph</a>	density as function of pressure and specific enthalpy
 <a href="#">rho_ph_der</a>	derivative function of rho_ph
 <a href="#">T_props_ph</a>	temperature as function of pressure and specific enthalpy
 <a href="#">T_ph</a>	temperature as function of pressure and specific enthalpy
 <a href="#">T_ph_der</a>	derivative function of T_ph
 <a href="#">s_props_ph</a>	specific entropy as function of pressure and specific enthalpy
 <a href="#">s_ph</a>	specific entropy as function of pressure and specific enthalpy
 <a href="#">s_ph_der</a>	specific entropy as function of pressure and specific enthalpy
 <a href="#">cv_props_ph</a>	specific heat capacity at constant volume as function of pressure and specific enthalpy

## 1616 Modelica.Media.Water.IF97\_Utilities

(f) <code>cv_ph</code>	specific heat capacity at constant volume as function of pressure and specific enthalpy
(f) <code>regionAssertReal</code>	assert function for inlining
(f) <code>cp_props_ph</code>	specific heat capacity at constant pressure as function of pressure and specific enthalpy
(f) <code>cp_ph</code>	specific heat capacity at constant pressure as function of pressure and specific enthalpy
(f) <code>beta_props_ph</code>	isobaric expansion coefficient as function of pressure and specific enthalpy
(f) <code>beta_ph</code>	isobaric expansion coefficient as function of pressure and specific enthalpy
(f) <code>kappa_props_ph</code>	isothermal compressibility factor as function of pressure and specific enthalpy
(f) <code>kappa_ph</code>	isothermal compressibility factor as function of pressure and specific enthalpy
(f) <code>velocityOfSound_props_ph</code>	speed of sound as function of pressure and specific enthalpy
(f) <code>velocityOfSound_ph</code>	
(f) <code>isentropicExponent_props_ph</code>	isentropic exponent as function of pressure and specific enthalpy
(f) <code>isentropicExponent_ph</code>	isentropic exponent as function of pressure and specific enthalpy
(f) <code>ddph_props</code>	density derivative by pressure
(f) <code>ddph</code>	density derivative by pressure
(f) <code>ddhp_props</code>	density derivative by specific enthalpy
(f) <code>ddhp</code>	density derivative by specific enthalpy
(f) <code>waterBaseProp_pT</code>	intermediate property record for water (p and T prefered states)
(f) <code>rho_props_pT</code>	density as function or pressure and temperature
(f) <code>rho_pT</code>	density as function or pressure and temperature
(f) <code>h_props_pT</code>	specific enthalpy as function or pressure and temperature
(f) <code>h_pT</code>	specific enthalpy as function or pressure and temperature
(f) <code>h_pT_der</code>	derivative function of <code>h_pT</code>
(f) <code>rho_pT_der</code>	derivative function of <code>rho_pT</code>
(f) <code>s_props_pT</code>	specific entropy as function of pressure and temperature
(f) <code>s_pT</code>	temperature as function of pressure and temperature
(f) <code>cv_props_pT</code>	specific heat capacity at constant volume as function of pressure and temperature
(f) <code>cv_pT</code>	specific heat capacity at constant volume as function of pressure and temperature
(f) <code>cp_props_pT</code>	specific heat capacity at constant pressure as function of pressure and temperature
(f) <code>cp_pT</code>	specific heat capacity at constant pressure as function of pressure and temperature
(f) <code>beta_props_pT</code>	isobaric expansion coefficient as function of pressure and temperature
(f) <code>beta_pT</code>	isobaric expansion coefficient as function of pressure and temperature

(f) <code>kappa_props_pT</code>	isothermal compressibility factor as function of pressure and temperature
(f) <code>kappa_pT</code>	isothermal compressibility factor as function of pressure and temperature
(f) <code>velocityOfSound_props_pT</code>	speed of sound as function of pressure and temperature
(f) <code>velocityOfSound_pT</code>	speed of sound as function of pressure and temperature
(f) <code>isentropicExponent_props_pT</code>	isentropic exponent as function of pressure and temperature
(f) <code>isentropicExponent_pT</code>	isentropic exponent as function of pressure and temperature
(f) <code>waterBaseProp_dT</code>	intermediate property record for water (d and T prefered states)
(f) <code>h_props_dT</code>	specific enthalpy as function of density and temperature
(f) <code>h_dT</code>	specific enthalpy as function of density and temperature
(f) <code>h_dT_der</code>	derivative function of <code>h_dT</code>
(f) <code>p_props_dT</code>	pressure as function of density and temperature
(f) <code>p_dT</code>	pressure as function of density and temperature
(f) <code>p_dT_der</code>	derivative function of <code>p_dT</code>
(f) <code>s_props_dT</code>	specific entropy as function of density and temperature
(f) <code>s_dT</code>	temperature as function of density and temperature
(f) <code>cv_props_dT</code>	specific heat capacity at constant volume as function of density and temperature
(f) <code>cv_dT</code>	specific heat capacity at constant volume as function of density and temperature
(f) <code>cp_props_dT</code>	specific heat capacity at constant pressure as function of density and temperature
(f) <code>cp_dT</code>	specific heat capacity at constant pressure as function of density and temperature
(f) <code>beta_props_dT</code>	isobaric expansion coefficient as function of density and temperature
(f) <code>beta_dT</code>	isobaric expansion coefficient as function of density and temperature
(f) <code>kappa_props_dT</code>	isothermal compressibility factor as function of density and temperature
(f) <code>kappa_dT</code>	isothermal compressibility factor as function of density and temperature
(f) <code>velocityOfSound_props_dT</code>	speed of sound as function of density and temperature
(f) <code>velocityOfSound_dT</code>	speed of sound as function of density and temperature
(f) <code>isentropicExponent_props_dT</code>	isentropic exponent as function of density and temperature
(f) <code>isentropicExponent_dT</code>	isentropic exponent as function of density and temperature
(f) <code>hl_p</code>	compute the saturated liquid specific $h(p)$
(f) <code>hv_p</code>	compute the saturated vapour specific $h(p)$
(f) <code>sl_p</code>	compute the saturated liquid specific $s(p)$
(f) <code>sv_p</code>	compute the saturated vapour specific $s(p)$
(f) <code>rhol_T</code>	compute the saturated liquid $d(T)$
(f) <code>rhov_T</code>	compute the saturated vapour $d(T)$
(f) <code>rhol_p</code>	compute the saturated liquid $d(p)$

## 1618 Modelica.Media.Water.IF97\_Utilities

(f) <code>rholv_p</code>	compute the saturated vapour d(p)
(f) <code>dynamicViscosity</code>	compute eta(d,T) in the one-phase region
(f) <code>thermalConductivity</code>	compute lambda(d,T,p) in the one-phase region
(f) <code>surfaceTension</code>	compute sigma(T) at saturation T
(f) <code>isentropicEnthalpy</code>	isentropic specific enthalpy from p,s (preferably use <code>dynamicIsentropicEnthalpy</code> in dynamic simulation!)
(f) <code>isentropicEnthalpy_props</code>	
(f) <code>isentropicEnthalpy_der</code>	derivative of isentropic specific enthalpy from p,s
(f) <code>dynamicIsentropicEnthalpy</code>	isentropic specific enthalpy from p,s and good guesses of d and T

## Modelica.Media.Water.IF97\_Utilities.BaselF97

### Modelica Physical Property Model: the new industrial formulation IAPWS-IF97

#### Information

##### Version Info and Revision history

- First implemented: *July, 2000* by [Hubertus Tummescheit](#) for the ThermoFluid Library with help from Jonas Eborn and Falko Jens Wagner
- Code reorganization, enhanced documentation, additional functions: *December, 2002* by [Hubertus Tummescheit](#) and moved to Modelica properties library.

*Author:* Hubertus Tummescheit,  
Modelon AB  
Ideon Science Park  
SE-22370 Lund, Sweden  
*email:* hubertus@modelon.se

In September 1997, the International Association for the Properties of Water and Steam ([IAPWS](#)) adopted a new formulation for the thermodynamic properties of water and steam for industrial use. This new industrial standard is called "IAPWS Industrial Formulation for the Thermodynamic Properties of Water and Steam" (IAPWS-IF97). The formulation IAPWS-IF97 replaces the previous industrial standard IFC-67.

Based on this new formulation, a new steam table, titled "[Properties of Water and Steam](#)" by W. Wagner and A. Kruse, was published by the Springer-Verlag, Berlin - New-York - Tokyo in April 1998. This steam table, ref. [1] is bilingual (English / German) and contains a complete description of the equations of IAPWS-IF97. This reference is the authoritative source of information for this implementation. A mostly identical version has been published by the International Association for the Properties of Water and Steam ([IAPWS](#)) with permission granted to re-publish the information if credit is given to IAPWS. This document is distributed with this library as [IF97.pdf](#). In addition, the equations published by IAPWS for the transport properties dynamic viscosity (standards document: [visc.pdf](#)) and thermal conductivity (standards document: [thcond.pdf](#)) and equations for the surface tension (standards document: [surf.pdf](#)) are also implemented in this library and included for reference.

The functions in BaselF97.mo are low level functions which should only be used in those exceptions when the standard user level functions in Water.mo do not contain the wanted properties.

Based on IAPWS-IF97, Modelica functions are available for calculating the most common thermophysical properties (thermodynamic and transport properties). The implementation requires part of the common medium property infrastructure of the Modelica.Thermal.Properties library in the file Common.mo. There are a few extensions from the version of IF97 as documented in [IF97.pdf](#) in order to improve performance for dynamic simulations. Input variables for calculating the properties are only implemented for a limited number of variable pairs which make sense as dynamic states: (p,h), (p,T), (p,s) and (d,T).

## 1. Structure and Regions of IAPWS-IF97

The IAPWS Industrial Formulation 1997 consists of a set of equations for different regions which cover the following range of validity:

$$273,15 \text{ K} < T < 1073,15 \text{ K} \quad p < 100 \text{ MPa}$$

$$1073,15 \text{ K} < T < 2273,15 \text{ K} \quad p < 10 \text{ MPa}$$

Figure 1 shows the 5 regions into which the entire range of validity of IAPWS-IF97 is divided. The boundaries of the regions can be directly taken from Fig. 1 except for the boundary between regions 2 and 3; this boundary, which corresponds approximately to the isentropic line  $s = 5.047 \text{ kJ kg}^{-1} \text{ K}^{-1}$ , is defined by a corresponding auxiliary equation. Both regions 1 and 2 are individually covered by a fundamental equation for the specific Gibbs free energy  $g(p, T)$ , region 3 by a fundamental equation for the specific Helmholtz free energy  $f(p, T)$ , and the saturation curve, corresponding to region 4, by a saturation-pressure equation  $p_s(T)$ . The high-temperature region 5 is also covered by a  $g(p, T)$  equation. These 5 equations, shown in rectangular boxes in Fig. 1, form the so-called *basic equations*.



Figure 1: Regions and equations of IAPWS-IF97

In addition to these basic equations, so-called *backward equations* are provided for regions 1, 2, and 4 in form of  $T(p, h)$  and  $T(p, s)$  for regions 1 and 2, and  $T_s(p)$  for region 4. These backward equations, marked in grey in Fig. 1, were developed in such a way that they are numerically very consistent with the corresponding basic equation. Thus, properties as functions of  $p, h$  and of  $p, s$  for regions 1 and 2, and of  $p$  for region 4 can be calculated without any iteration. As a result of this special concept for the development of the new industrial standard IAPWS-IF97, the most important properties can be calculated extremely quickly. All modelica functions are optimized with regard to short computing times.

The complete description of the individual equations of the new industrial formulation IAPWS-IF97 is given in [IF97.pdf](#). Comprehensive information on IAPWS-IF97 (requirements, concept, accuracy, consistency along region boundaries, and the increase of computing speed in comparison with IFC-67, etc.) can be taken from [IF97.pdf](#) or [2].

[1] Wagner, W., Kruse, A. Properties of Water and Steam / Zustandsgrößen von Wasser und Wasserdampf / IAPWS-IF97. Springer-Verlag, Berlin, 1998.

[2] Wagner, W., Cooper, J. R., Dittmann, A., Kijima, J., Kretzschmar, H.-J., Kruse, A., Marei, R., Oguchi, K., Sato, H., Stöcker, I., Trifner, O., Takaishi, Y., Tanishita, I., Trübenbach, J., and Willkommen, Th. The IAPWS Industrial Formulation 1997 for the Thermodynamic Properties of Water and Steam. ASME Journal of Engineering for Gas Turbines and Power 122 (2000), 150 - 182.

## 2. Calculable Properties

	Common name	Abbreviation	Unit
1	Pressure	p	Pa
2	Temperature	T	K
3	Density	d	kg/m <sup>3</sup>
4	Specific volume	v	m <sup>3</sup> /kg
5	Specific enthalpy	h	J/kg
6	Specific entropy	s	J/(kg K)
7	Specific internal energy	u	J/kg

8	Specific isobaric heat capacity	$c_p$	J/(kg K)
9	Specific isochoric heat capacity	$c_v$	J/(kg K)
10	Isentropic exponent, $\kappa = -(v/p) (dp/dv)_s$	$\kappa$	1
11	Speed of sound	$a$	m/s
12	Dryness fraction	$x$	kg/kg
13	Specific Helmholtz free energy, $f = u - Ts$	$f$	J/kg
14	Specific Gibbs free energy, $g = h - Ts$	$g$	J/kg
15	Isenthalpic exponent, $\theta = -(v/p)(dp/dv)_h$	$\theta$	1
16	Isobaric volume expansion coefficient, $\alpha = v^{-1} (dv/dT)_p$	$\alpha$	1/K
17	Isochoric pressure coefficient, $\beta = p^{-1} (dp/dT)_v$	$\beta$	1/K
18	Isothermal compressibility, $\gamma = -v^{-1}(dv/dp)_T$	$\gamma$	1/Pa
19	Dynamic viscosity	$\eta$	Pa s
20	Kinematic viscosity	$\nu$	m <sup>2</sup> /s
21	Thermal conductivity	$\lambda$	W/(m K)
22	Surface tension	$\sigma$	N/m

The properties 1-11 are calculated by default with the functions for dynamic simulation, 2 of these variables are the dynamic states and are the inputs to calculate all other properties. In addition to these properties of general interest, the entries to the thermodynamic Jacobian matrix which render the mass- and energy balances explicit in the input variables to the property calculation are also calculated. For an explanatory example using pressure and specific enthalpy as states, see the Examples sub-package.

The high-level calls to steam properties are grouped into records comprising both the properties of general interest and the entries to the thermodynamic Jacobian. If additional properties are needed the low level functions in BaselF97 provide more choice.

## Additional functions

- Function **boundaryvals\_p** computes the temperature and the specific enthalpy and entropy on both phase boundaries as a function of p
- Function **boundaryderivs\_p** is the Modelica derivative function of **boundaryvals\_p**
- Function **extraDerivs\_ph** computes all entries to Bridgmans tables for all one-phase regions of IF97 using inputs (p,h). All 336 directional derivatives of the thermodynamic surface can be computed as a ratio of two entries in the return data, see package Common for details.
- Function **extraDerivs\_pT** computes all entries to Bridgmans tables for all one-phase regions of IF97 using inputs (p,T).

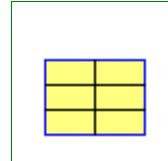
## Package Content

Name	Description
 IterationData	constants for iterations internal to some functions
 data	constant IF97 data and region limits
 getTstar	get normalization temperature for region 1, 2 or 5
 getpstar	get normalization pressure for region 1, 2 or 5
 critical	critical point data
 triple	triple point data

 Regions	functions to find the current region for given pairs of input variables
 Basic	Base functions as described in IAWPS/IF97
 IceBoundaries	the melting line and sublimation line curves from IAPWS
 Transport	transport properties for water according to IAPWS/IF97
 Isentropic	functions for calculating the isentropic enthalpy from pressure p and specific entropy s
 Inverses	efficient inverses for selected pairs of variables
 ByRegion	simple explicit functions for one region only
 TwoPhase	steam properties in the two-phase rgion and on the phase boundaries
 extraDerivs_ph	function to calculate some extra thermophysical properties in regions 1, 2, 3 and 5 as f(p,h)
 extraDerivs_pT	function to calculate some extra thermophysical properties in regions 1, 2, 3 and 5 as f(p,T)

## Modelica.Media.Water.IF97\_Utils.BaselIF97.IterationData

constants for iterations internal to some functions



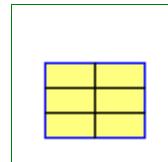
### Modelica definition

```
record IterationData
  "constants for iterations internal to some functions"

  extends Modelica.Icons.Record;
  constant Integer IMAX=50 "maximum number of iterations for inverse functions";
  constant Real DELP=1.0e-6 "maximum iteration error in pressure, Pa";
  constant Real DELS=1.0e-8
    "maximum iteration error in specific entropy, J/{kg.K}";
  constant Real DELH=1.0e-8
    "maximum iteration error in specific entthalpy, J/kg";
  constant Real DELD=1.0e-8 "maximum iteration error in density, kg/m^3";
end IterationData;
```

## Modelica.Media.Water.IF97\_Utils.BaselIF97.data

constant IF97 data and region limits



### Information

#### Record description

Constants needed in the international steam properties IF97. SCRIT and HCRIT are calculated from Helmholtz function for region 3.

#### Version Info and Revision history

- First implemented: July, 2000 by Hubertus Tummescheit

Author: Hubertus Tummescheit,  
Lund University

## 1622 Modelica.Media.Water.IF97\_Utilities.BaselF97.data

---

Department of Automatic Control  
Box 118, 22100 Lund, Sweden  
email: hubertus@control.lth.se

- Initial version: July 2000
  - Documentation added: December 2002
- 

### Modelica.Media.Water.IF97\_Utilities.BaselF97.getTstar

get normalization temperature for region 1, 2 or 5



#### Inputs

Name	Default	Description
region		IF 97 region

#### Outputs

Name	Description
Tstar	normalization temperature [K]

---

### Modelica.Media.Water.IF97\_Utilities.BaselF97.getpstar

get normalization pressure for region 1, 2 or 5



#### Inputs

Name	Default	Description
region		IF 97 region

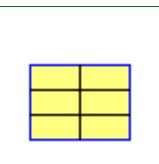
#### Outputs

Name	Description
pstar	normalization pressure [Pa]

---

### Modelica.Media.Water.IF97\_Utilities.BaselF97.critical

critical point data



#### Information

##### Record description

Critical point data for IF97 steam properties. SCRIT and HCRIT are calculated from helmholtz function for region 3

##### Version Info and Revision history

- First implemented: July, 2000 by Hubertus Tummescheit

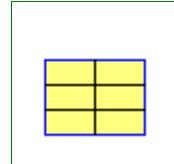
Author: Hubertus Tummescheit,  
Lund University

*Department of Automatic Control  
Box 118, 22100 Lund, Sweden  
email: hubertus@control.lth.se*

- Initial version: July 2000
- Documentation added: December 2002

## Modelica.Media.Water.IF97\_Utilities.BaseIF97.triple

triple point data



### Information

#### Record description

Vapour/liquid/ice triple point data for IF97 steam properties.

#### Version Info and Revision history

- First implemented: July, 2000 by Hubertus Tummescheit

*Author: Hubertus Tummescheit,  
Lund University*

*Department of Automatic Control  
Box 118, 22100 Lund, Sweden  
email: hubertus@control.lth.se*

- Initial version: July 2000
- Documentation added: December 2002

## Modelica.Media.Water.IF97\_Utilities.BaseIF97.Regions

functions to find the current region for given pairs of input variables

### Information

#### Package description

Package **Regions** contains a large number of auxiliary functions which are needed to compute the current region of the IAPWS/IF97 for a given pair of input variables as quickly as possible. The focus of this implementation was on computational efficiency, not on compact code. Many of the function values calculated in these functions could be obtained using the fundamental functions of IAPWS/IF97, but with considerable overhead. If the region of IAPWS/IF97 is known in advance, the input variable mode can be set to the region, then the somewhat costly region checks are omitted. The checking for the phase has to be done outside the region functions because many properties are not differentiable at the region boundary. If the input phase is 2, the output region will be set to 4 immediately.

#### Package contents

The main 4 functions in this package are the functions returning the appropriate region for two input variables.

- Function **region\_ph** compute the region of IAPWS/IF97 for input pair pressure and specific enthalpy.
- Function **region\_ps** compute the region of IAPWS/IF97 for input pair pressure and specific entropy

- Function **region\_dT** compute the region of IAPWS/IF97 for input pair density and temperature.
- Function **region\_pT** compute the region of IAPWS/IF97 for input pair pressure and temperature (only one phase region).

In addition, functions of the boiling and condensation curves compute the specific enthalpy, specific entropy, or density on these curves. The functions for the saturation pressure and temperature are included in the package **Basic** because they are part of the original [IAPWS/IF97 standards document](#). These functions are also aliased to be used directly from package **Water**.

- Function **hl\_p** computes the liquid specific enthalpy as a function of pressure. For overcritical pressures, the critical specific enthalpy is returned. An approximation is used for temperatures > 623.15 K.
- Function **hv\_p** computes the vapour specific enthalpy as a function of pressure. For overcritical pressures, the critical specific enthalpy is returned. An approximation is used for temperatures > 623.15 K.
- Function **sl\_p** computes the liquid specific entropy as a function of pressure. For overcritical pressures, the critical specific entropy is returned. An approximation is used for temperatures > 623.15 K.
- Function **sv\_p** computes the vapour specific entropy as a function of pressure. For overcritical pressures, the critical specific entropy is returned. An approximation is used for temperatures > 623.15 K.
- Function **rhol\_T** computes the liquid density as a function of temperature. For overcritical temperatures, the critical density is returned. An approximation is used for temperatures > 623.15 K.
- Function **rhol\_T** computes the vapour density as a function of temperature. For overcritical temperatures, the critical density is returned. An approximation is used for temperatures > 623.15 K.

All other functions are auxiliary functions called from the region functions to check a specific boundary.

- Function **boundary23ofT** computes the boundary pressure between regions 2 and 3 (input temperature)
- Function **boundary23ofp** computes the boundary temperature between regions 2 and 3 (input pressure)
- Function **hlowerofp5** computes the lower specific enthalpy limit of region 5 (input p, T=1073.15 K)
- Function **hupperofp5** computes the upper specific enthalpy limit of region 5 (input p, T=2273.15 K)
- Function **slowerofp5** computes the lower specific entropy limit of region 5 (input p, T=1073.15 K)
- Function **supperofp5** computes the upper specific entropy limit of region 5 (input p, T=2273.15 K)
- Function **hlowerofp1** computes the lower specific enthalpy limit of region 1 (input p, T=273.15 K)
- Function **hupperofp1** computes the upper specific enthalpy limit of region 1 (input p, T=623.15 K)
- Function **slowerofp1** computes the lower specific entropy limit of region 1 (input p, T=273.15 K)
- Function **supperofp1** computes the upper specific entropy limit of region 1 (input p, T=623.15 K)
- Function **hlowerofp2** computes the lower specific enthalpy limit of region 2 (input p, T=623.15 K)
- Function **hupperofp2** computes the upper specific enthalpy limit of region 2 (input p, T=1073.15 K)
- Function **slowerofp2** computes the lower specific entropy limit of region 2 (input p, T=623.15 K)
- Function **supperofp2** computes the upper specific entropy limit of region 2 (input p, T=1073.15 K)
- Function **d1n** computes the density in region 1 as function of pressure and temperature
- Function **d2n** computes the density in region 2 as function of pressure and temperature
- Function **dhot1ofp** computes the hot density limit of region 1 (input p, T=623.15 K)
- Function **dupper1ofT** computes the high pressure density limit of region 1 (input T, p=100MPa)
- Function **hl\_p\_R4b** computes a high accuracy approximation to the liquid enthalpy for temperatures > 623.15 K (input p)
- Function **hv\_p\_R4b** computes a high accuracy approximation to the vapour enthalpy for temperatures > 623.15 K (input p)
- Function **sl\_p\_R4b** computes a high accuracy approximation to the liquid entropy for temperatures > 623.15 K (input p)
- Function **sv\_p\_R4b** computes a high accuracy approximation to the vapour entropy for temperatures > 623.15 K (input p)
- Function **rhol\_p\_R4b** computes a high accuracy approximation to the liquid density for temperatures > 623.15 K (input p)
- Function **rhov\_p\_R4b** computes a high accuracy approximation to the vapour density for temperatures > 623.15 K (input p)

## Version Info and Revision history

- First implemented: July, 2000 by Hubertus Tummescheit

Authors: Hubertus Tummescheit, Jonas Eborn and Falko Jens Wagner

Lund University

Department of Automatic Control

Box 118, 22100 Lund, Sweden

email: hubertus@control.lth.se

- Initial version: July 2000
- Revised and extended for inclusion in Modelica.Thermal: December 2002

## Package Content

Name	Description
(f) boundary23oft	boundary function for region boundary between regions 2 and 3 (input temperature)
(f) boundary23ofp	boundary function for region boundary between regions 2 and 3 (input pressure)
(f) hlowerofp5	explicit lower specific enthalpy limit of region 5 as function of pressure
(f) hupperofp5	explicit upper specific enthalpy limit of region 5 as function of pressure
(f) slowerofp5	explicit lower specific entropy limit of region 5 as function of pressure
(f) supperofp5	explicit upper specific entropy limit of region 5 as function of pressure
(f) hlowerofp1	explicit lower specific enthalpy limit of region 1 as function of pressure
(f) hupperofp1	explicit upper specific enthalpy limit of region 1 as function of pressure (meets region 4 saturation pressure curve at 623.15 K)
(f) slowerofp1	explicit lower specific entropy limit of region 1 as function of pressure
(f) supperofp1	explicit upper specific entropy limit of region 1 as function of pressure (meets region 4 saturation pressure curve at 623.15 K)
(f) hlowerofp2	explicit lower specific enthalpy limit of region 2 as function of pressure (meets region 4 saturation pressure curve at 623.15 K)
(f) hupperofp2	explicit upper specific enthalpy limit of region 2 as function of pressure
(f) slowerofp2	explicit lower specific entropy limit of region 2 as function of pressure (meets region 4 saturation pressure curve at 623.15 K)
(f) supperofp2	explicit upper specific entropy limit of region 2 as function of pressure
(f) d1n	density in region 1 as function of p and T
(f) d2n	density in region 2 as function of p and T
(f) dhot1ofp	density at upper temperature limit of region 1
(f) dupper1ofT	density at upper pressure limit of region 1
(f) hl_p_R4b	explicit approximation of liquid specific enthalpy on the boundary between regions 4 and 3
(f) hv_p_R4b	explicit approximation of vapour specific enthalpy on the boundary between regions 4 and 3
(f) sl_p_R4b	explicit approximation of liquid specific entropy on the boundary between regions 4 and 3
(f) sv_p_R4b	explicit approximation of vapour specific entropy on the boundary between regions 4 and 3
(f) rhol_p_R4b	explicit approximation of liquid density on the boundary between regions 4 and 3

## 1626 Modelica.Media.Water.IF97\_Utils.BaselF97.Regions

<a href="#"> rhov_p_R4b</a>	explicit approximation of vapour density on the boundary between regions 4 and 2
<a href="#"> boilingcurve_p</a>	properties on the boiling curve
<a href="#"> dewcurve_p</a>	properties on the dew curve
<a href="#"> hvl_p</a>	
<a href="#"> hl_p</a>	liquid specific enthalpy on the boundary between regions 4 and 3 or 1
<a href="#"> hv_p</a>	vapour specific enthalpy on the boundary between regions 4 and 3 or 2
<a href="#"> hvl_p_der</a>	derivative function for the specific enthalpy along the phase boundary
<a href="#"> rho1_p</a>	
<a href="#"> rhol_p</a>	density of saturated water
<a href="#"> rhov_p</a>	density of saturated vapour
<a href="#"> rho1_p_der</a>	
<a href="#"> sl_p</a>	liquid specific entropy on the boundary between regions 4 and 3 or 1
<a href="#"> sv_p</a>	vapour specific entropy on the boundary between regions 4 and 3 or 2
<a href="#"> rho1_T</a>	density of saturated water
<a href="#"> rhov_T</a>	density of saturated vapour
<a href="#"> region_ph</a>	return the current region (valid values: 1,2,3,4,5) in IF97 for given pressure and specific enthalpy
<a href="#"> region_ps</a>	return the current region (valid values: 1,2,3,4,5) in IF97 for given pressure and specific entropy
<a href="#"> region_pT</a>	return the current region (valid values: 1,2,3,5) in IF97, given pressure and temperature
<a href="#"> region_dT</a>	return the current region (valid values: 1,2,3,4,5) in IF97, given density and temperature
<a href="#"> hvl_dp</a>	derivative function for the specific enthalpy along the phase boundary
<a href="#"> dhl_dp</a>	derivative of liquid specific enthalpy on the boundary between regions 4 and 3 or 1 w.r.t pressure
<a href="#"> dhv_dp</a>	derivative of vapour specific enthalpy on the boundary between regions 4 and 3 or 1 w.r.t pressure
<a href="#"> drho1_dp</a>	
<a href="#"> drhol_dp</a>	derivative of density of saturated water w.r.t. pressure
<a href="#"> drhov_dp</a>	derivative of density of saturated steam w.r.t. pressure

### Modelica.Media.Water.IF97\_Utils.BaselF97.Regions.boundary23ofT

boundary function for region boundary between regions 2 and 3 (input temperature)



#### Inputs

Name	Default	Description
t		temperature (K) [K]

#### Outputs

Name	Description
p	pressure [Pa]

**Modelica.Media.Water.IF97\_Utilsities.BaselF97.Regions.boundary23ofp**

boundary function for region boundary between regions 2 and 3 (input pressure)

**Inputs**

Name	Default	Description
p		pressure [Pa]

**Outputs**

Name	Description
t	temperature (K) [K]

**Modelica.Media.Water.IF97\_Utilsities.BaselF97.Regions.hlowerofp5**

explicit lower specific enthalpy limit of region 5 as function of pressure

**Inputs**

Name	Default	Description
p		pressure [Pa]

**Outputs**

Name	Description
h	specific enthalpy [J/kg]

**Modelica.Media.Water.IF97\_Utilsities.BaselF97.Regions.hupperofp5**

explicit upper specific enthalpy limit of region 5 as function of pressure

**Inputs**

Name	Default	Description
p		pressure [Pa]

**Outputs**

Name	Description
h	specific enthalpy [J/kg]

**Modelica.Media.Water.IF97\_Utilsities.BaselF97.Regions.slowerofp5**

explicit lower specific entropy limit of region 5 as function of pressure

**Inputs**

Name	Default	Description
p		pressure [Pa]

---

## 1628 Modelica.Media.Water.IF97\_Utilities.BaselF97.Regions.slowerofp5

---

### Outputs

Name	Description
s	specific entropy [J/(kg.K)]

---



### Modelica.Media.Water.IF97\_Utilities.BaselF97.Regions.supperofp5

explicit upper specific entropy limit of region 5 as function of pressure

### Inputs

Name	Default	Description
p		pressure [Pa]

### Outputs

Name	Description
s	specific entropy [J/(kg.K)]

---



### Modelica.Media.Water.IF97\_Utilities.BaselF97.Regions.hlowerofp1

explicit lower specific enthalpy limit of region 1 as function of pressure

### Inputs

Name	Default	Description
p		pressure [Pa]

### Outputs

Name	Description
h	specific enthalpy [J/kg]

---



### Modelica.Media.Water.IF97\_Utilities.BaselF97.Regions.hupperofp1

explicit upper specific enthalpy limit of region 1 as function of pressure (meets region 4 saturation pressure curve at 623.15 K)

### Inputs

Name	Default	Description
p		pressure [Pa]

### Outputs

Name	Description
h	specific enthalpy [J/kg]

---



### Modelica.Media.Water.IF97\_Utilities.BaselF97.Regions.slowerofp1

explicit lower specific entropy limit of region 1 as function of pressure

**Inputs**

Name	Default	Description
p		pressure [Pa]

**Outputs**

Name	Description
s	specific entropy [J/(kg.K)]

**Modelica.Media.Water.IF97\_Utilsities.BaselF97.Regions.supperofp1**

explicit upper specific entropy limit of region 1 as function of pressure (meets region 4 saturation pressure curve at 623.15 K)

**Inputs**

Name	Default	Description
p		pressure [Pa]

**Outputs**

Name	Description
s	specific entropy [J/(kg.K)]

**Modelica.Media.Water.IF97\_Utilsities.BaselF97.Regions.hlowerofp2**

explicit lower specific enthalpy limit of region 2 as function of pressure (meets region 4 saturation pressure curve at 623.15 K)

**Inputs**

Name	Default	Description
p		pressure [Pa]

**Outputs**

Name	Description
h	specific enthalpy [J/kg]

**Modelica.Media.Water.IF97\_Utilsities.BaselF97.Regions.hupperofp2**

explicit upper specific enthalpy limit of region 2 as function of pressure

**Inputs**

Name	Default	Description
p		pressure [Pa]

**Outputs**

Name	Description

---

**1630 Modelica.Media.Water.IF97\_Utilities.BaseIF97.Regions.hupperofp2**

---

h	specific enthalpy [J/kg]
---	--------------------------

---

**Modelica.Media.Water.IF97\_Utilities.BaseIF97.Regions.slowerofp2**

explicit lower specific entropy limit of region 2 as function of pressure (meets region 4 saturation pressure curve at 623.15 K)

**Inputs**

Name	Default	Description
p		pressure [Pa]

**Outputs**

Name	Description
s	specific entropy [J/(kg.K)]

---

**Modelica.Media.Water.IF97\_Utilities.BaseIF97.Regions.supperofp2**

explicit upper specific entropy limit of region 2 as function of pressure

**Inputs**

Name	Default	Description
p		pressure [Pa]

**Outputs**

Name	Description
s	specific entropy [J/(kg.K)]

---

**Modelica.Media.Water.IF97\_Utilities.BaseIF97.Regions.d1n**

density in region 1 as function of p and T

**Inputs**

Name	Default	Description
p		pressure [Pa]
T		temperature (K) [K]

**Outputs**

Name	Description
d	density [kg/m3]

---

**Modelica.Media.Water.IF97\_Utilities.BaseIF97.Regions.d2n**

density in region 2 as function of p and T



**Inputs**

Name	Default	Description
p		pressure [Pa]
T		temperature (K) [K]

**Outputs**

Name	Description
d	density [kg/m3]

**Modelica.Media.Water.IF97\_Utils.BaselF97.Regions.dhot1ofp**

density at upper temperature limit of region 1

**Inputs**

Name	Default	Description
p		pressure [Pa]

**Outputs**

Name	Description
d	density [kg/m3]

**Modelica.Media.Water.IF97\_Utils.BaselF97.Regions.dupper1oft**

density at upper pressure limit of region 1

**Inputs**

Name	Default	Description
T		temperature (K) [K]

**Outputs**

Name	Description
d	density [kg/m3]

**Modelica.Media.Water.IF97\_Utils.BaselF97.Regions.hl\_p\_R4b**

explicit approximation of liquid specific enthalpy on the boundary between regions 4 and 3

**Inputs**

Name	Default	Description
p		pressure [Pa]

**Outputs**

Name	Description

---

**1632 Modelica.Media.Water.IF97\_Utilities.BaselF97.Regions.hl\_p\_R4b**

---

h	specific enthalpy [J/kg]
---	--------------------------

---

**Modelica.Media.Water.IF97\_Utilities.BaselF97.Regions.hv\_p\_R4b**

explicit approximation of vapour specific enthalpy on the boundary between regions 4 and 3

**Inputs**

Name	Default	Description
p		pressure [Pa]

**Outputs**

Name	Description
h	specific enthalpy [J/kg]

---

**Modelica.Media.Water.IF97\_Utilities.BaselF97.Regions.sl\_p\_R4b**

explicit approximation of liquid specific entropy on the boundary between regions 4 and 3

**Inputs**

Name	Default	Description
p		pressure [Pa]

**Outputs**

Name	Description
s	specific entropy [J/(kg.K)]

---

**Modelica.Media.Water.IF97\_Utilities.BaselF97.Regions.sv\_p\_R4b**

explicit approximation of vapour specific entropy on the boundary between regions 4 and 3

**Inputs**

Name	Default	Description
p		pressure [Pa]

**Outputs**

Name	Description
s	[J/kg]

---

**Modelica.Media.Water.IF97\_Utilities.BaselF97.Regions.rhol\_p\_R4b**

explicit approximation of liquid density on the boundary between regions 4 and 3



**Inputs**

Name	Default	Description
p		pressure [Pa]

**Outputs**

Name	Description
dl	liquid density [kg/m3]

**Modelica.Media.Water.IF97\_Utilsities.BaselF97.Regions.rhov\_p\_R4b**

explicit approximation of vapour density on the boundary between regions 4 and 2

**Inputs**

Name	Default	Description
p		pressure [Pa]

**Outputs**

Name	Description
dv	vapour density [kg/m3]

**Modelica.Media.Water.IF97\_Utilsities.BaselF97.Regions.boilingcurve\_p**

properties on the boiling curve

**Inputs**

Name	Default	Description
p		pressure [Pa]

**Outputs**

Name	Description
bpro	property record

**Modelica.Media.Water.IF97\_Utilsities.BaselF97.Regions.dewcurve\_p**

properties on the dew curve

**Inputs**

Name	Default	Description
p		pressure [Pa]

**Outputs**

Name	Description
bpro	property record

**Modelica.Media.Water.IF97\_Utilsities.BaselF97.Regions.hvl\_p****Inputs**

Name	Default	Description
p		pressure [Pa]
bpro		property record

**Outputs**

Name	Description
h	specific enthalpy [J/kg]

**Modelica.Media.Water.IF97\_Utilsities.BaselF97.Regions.hl\_p**

liquid specific enthalpy on the boundary between regions 4 and 3 or 1

**Inputs**

Name	Default	Description
p		pressure [Pa]

**Outputs**

Name	Description
h	specific enthalpy [J/kg]

**Modelica.Media.Water.IF97\_Utilsities.BaselF97.Regions.hv\_p**

vapour specific enthalpy on the boundary between regions 4 and 3 or 2

**Inputs**

Name	Default	Description
p		pressure [Pa]

**Outputs**

Name	Description
h	specific enthalpy [J/kg]

**Modelica.Media.Water.IF97\_Utilsities.BaselF97.Regions.hvl\_p\_der**

derivative function for the specific enthalpy along the phase boundary

**Inputs**

Name	Default	Description
p		pressure [Pa]

bpro	property record
p_der	derivative of pressure

## Outputs

Name	Description
h_der	time derivative of specific enthalpy along the phase boundary

## Modelica.Media.Water.IF97\_Utilities.BaselF97.Regions.rhol\_p



## Inputs

Name	Default	Description
p		pressure [Pa]
bpro		property record

## Outputs

Name	Description
rho	density [kg/m3]

## Modelica.Media.Water.IF97\_Utilities.BaselF97.Regions.rhol\_p



density of saturated water

## Inputs

Name	Default	Description
p		saturation pressure [Pa]

## Outputs

Name	Description
rho	density of steam at the condensation point [kg/m3]

## Modelica.Media.Water.IF97\_Utilities.BaselF97.Regions.rhov\_p



density of saturated vapour

## Inputs

Name	Default	Description
p		saturation pressure [Pa]

## Outputs

Name	Description
rho	density of steam at the condensation point [kg/m3]

---

**1636 Modelica.Media.Water.IF97\_Utilsities.BaselF97.Regions.rho1\_p\_der**

---

**Modelica.Media.Water.IF97\_Utilsities.BaselF97.Regions.rho1\_p\_der****Inputs**

Name	Default	Description
p		saturation pressure [Pa]
bpro		property record
p_der		derivative of pressure

**Outputs**

Name	Description
d_der	time derivative of density along the phase boundary

---

**Modelica.Media.Water.IF97\_Utilsities.BaselF97.Regions.sl\_p****liquid specific entropy on the boundary between regions 4 and 3 or 1****Inputs**

Name	Default	Description
p		pressure [Pa]

**Outputs**

Name	Description
s	specific entropy [J/(kg.K)]

---

**Modelica.Media.Water.IF97\_Utilsities.BaselF97.Regions.sv\_p****vapour specific entropy on the boundary between regions 4 and 3 or 2****Inputs**

Name	Default	Description
p		pressure [Pa]

**Outputs**

Name	Description
s	specific entropy [J/(kg.K)]

---

**Modelica.Media.Water.IF97\_Utilsities.BaselF97.Regions.rho1\_T****density of saturated water****Inputs**

Name	Default	Description
T		temperature [K]

## Outputs

Name	Description
d	density of water at the boiling point [kg/m3]

---

## Modelica.Media.Water.IF97\_Utilsities.BaselF97.Regions.rhov\_T

density of saturated vapour



## Inputs

Name	Default	Description
T		temperature [K]

## Outputs

Name	Description
d	density of steam at the condensation point [kg/m3]

---

## Modelica.Media.Water.IF97\_Utilsities.BaselF97.Regions.region\_ph

return the current region (valid values: 1,2,3,4,5) in IF97 for given pressure and specific enthalpy



## Inputs

Name	Default	Description
p		pressure [Pa]
h		specific enthalpy [J/kg]
phase	0	phase: 2 for two-phase, 1 for one phase, 0 if not known
mode	0	mode: 0 means check, otherwise assume region=mode

## Outputs

Name	Description
region	region (valid values: 1,2,3,4,5) in IF97

---

## Modelica.Media.Water.IF97\_Utilsities.BaselF97.Regions.region\_ps

return the current region (valid values: 1,2,3,4,5) in IF97 for given pressure and specific entropy



## Inputs

Name	Default	Description
p		pressure [Pa]
s		specific entropy [J/(kg.K)]
phase	0	phase: 2 for two-phase, 1 for one phase, 0 if unknown
mode	0	mode: 0 means check, otherwise assume region=mode

---

## 1638 Modelica.Media.Water.IF97\_Utils.BaselF97.Regions.region\_ps

---

### Outputs

Name	Description
region	region (valid values: 1,2,3,4,5) in IF97

---

### Modelica.Media.Water.IF97\_Utils.BaselF97.Regions.region\_pT

return the current region (valid values: 1,2,3,5) in IF97, given pressure and temperature



### Inputs

Name	Default	Description
p		pressure [Pa]
T		temperature (K) [K]
mode	0	mode: 0 means check, otherwise assume region=mode

### Outputs

Name	Description
region	region (valid values: 1,2,3,5) in IF97, region 4 is impossible!

---

### Modelica.Media.Water.IF97\_Utils.BaselF97.Regions.region\_dT

return the current region (valid values: 1,2,3,4,5) in IF97, given density and temperature



### Inputs

Name	Default	Description
d		density [kg/m3]
T		temperature (K) [K]
phase	0	phase: 2 for two-phase, 1 for one phase, 0 if not known
mode	0	mode: 0 means check, otherwise assume region=mode

### Outputs

Name	Description
region	(valid values: 1,2,3,4,5) in IF97

---

### Modelica.Media.Water.IF97\_Utils.BaselF97.Regions.hvl\_dp

derivative function for the specific enthalpy along the phase boundary



### Inputs

Name	Default	Description
p		pressure [Pa]
bpro		property record

## Outputs

Name	Description
dh_dp	derivative of specific enthalpy along the phase boundary

## Modelica.Media.Water.IF97\_Utils.BaselF97.Regions.dhl\_dp

derivative of liquid specific enthalpy on the boundary between regions 4 and 3 or 1 w.r.t pressure



## Inputs

Name	Default	Description
p		pressure [Pa]

## Outputs

Name	Description
dh_dp	specific enthalpy derivative w.r.t. pressure [J.m.s2/kg2]

## Modelica.Media.Water.IF97\_Utils.BaselF97.Regions.dhv\_dp

derivative of vapour specific enthalpy on the boundary between regions 4 and 3 or 1 w.r.t pressure



## Inputs

Name	Default	Description
p		pressure [Pa]

## Outputs

Name	Description
dh_dp	specific enthalpy derivative w.r.t. pressure [J.m.s2/kg2]

## Modelica.Media.Water.IF97\_Utils.BaselF97.Regions.drhovl\_dp



## Inputs

Name	Default	Description
p		saturation pressure [Pa]
bpro		property record

## Outputs

Name	Description
dd_dp	derivative of density along the phase boundary [kg/(m3.Pa)]

**Modelica.Media.Water.IF97\_Utilities.BaselF97.Regions.drhol\_dp**

derivative of density of saturated water w.r.t. pressure

**Inputs**

Name	Default	Description
p		saturation pressure [Pa]

**Outputs**

Name	Description
dd_dp	derivative of density of water at the boiling point [s <sup>2</sup> /m <sup>2</sup> ]

---

**Modelica.Media.Water.IF97\_Utilities.BaselF97.Regions.drhov\_dp**

derivative of density of saturated steam w.r.t. pressure

**Inputs**

Name	Default	Description
p		saturation pressure [Pa]

**Outputs**

Name	Description
dd_dp	derivative of density of water at the boiling point [s <sup>2</sup> /m <sup>2</sup> ]

---

**Modelica.Media.Water.IF97\_Utilities.BaselF97.Basic**

Base functions as described in IAWPS/IF97

**Information****Package description**

Package BaselF97/Basic computes the fundamental functions for the 5 regions of the steam tables as described in the standards document [IF97.pdf](#). The code of these functions has been generated using **Mathematica** and the add-on packages "Format" and "Optimize" to generate highly efficient, expression-optimized C-code from a symbolic representation of the thermodynamic functions. The C-code has then been transformed into Modelica code. An important feature of this optimization was to simultaneously optimize the functions and the directional derivatives because they share many common subexpressions.

**Package contents**

- Function **g1** computes the dimensionless Gibbs function for region 1 and all derivatives up to order 2 w.r.t pi and tau. Inputs: p and T.
- Function **g2** computes the dimensionless Gibbs function for region 2 and all derivatives up to order 2 w.r.t pi and tau. Inputs: p and T.
- Function **g2metastable** computes the dimensionless Gibbs function for metastable vapour (adjacent to region 2 but 2-phase at equilibrium) and all derivatives up to order 2 w.r.t pi and tau. Inputs: p and T.
- Function **f3** computes the dimensionless Helmholtz function for region 3 and all derivatives up to

- order 2 w.r.t delta and tau. Inputs: d and T.
- Function **g5** computes the dimensionless Gibbs function for region 5 and all derivatives up to order 2 w.r.t pi and tau. Inputs: p and T.
- Function **tph1** computes the inverse function T(p,h) in region 1.
- Function **tph2** computes the inverse function T(p,h) in region 2.
- Function **tps2a** computes the inverse function T(p,s) in region 2a.
- Function **tps2b** computes the inverse function T(p,s) in region 2b.
- Function **tps2c** computes the inverse function T(p,s) in region 2c.
- Function **tps2** computes the inverse function T(p,s) in region 2.
- Function **tsat** computes the saturation temperature as a function of pressure.
- Function **dtsatofp** computes the derivative of the saturation temperature w.r.t. pressure as a function of pressure.
- Function **tsat\_der** computes the Modelica derivative function of tsat.
- Function **psat** computes the saturation pressure as a function of temperature.
- Function **dptofT** computes the derivative of the saturation pressure w.r.t. temperature as a function of temperature.
- Function **psat\_der** computes the Modelica derivative function of psat.

### Version Info and Revision history

- First implemented: July, 2000 by Hubertus Tummescheit

*Author: Hubertus Tummescheit,  
Lund University  
Department of Automatic Control  
Box 118, 22100 Lund, Sweden  
email: hubertus@control.lth.se*

- Initial version: July 2000
- Documentation added: December 2002

### Package Content

Name	Description
(f) <b>g1</b>	Gibbs function for region 1: g(p,T)
(f) <b>g2</b>	Gibbs function for region 2: g(p,T)
(f) <b>g2metastable</b>	Gibbs function for metastable part of region 2: g(p,T)
(f) <b>f3</b>	Helmholtz function for region 3: f(d,T)
(f) <b>g5</b>	base function for region 5: g(p,T)
(f) <b>gibbs</b>	Gibbs function for region 1, 2 or 5: g(p,T,region)
(f) <b>g1pitau</b>	derivative of g wrt pi and tau
(f) <b>g2pitau</b>	derivative of g wrt pi and tau
(f) <b>g5pitau</b>	derivative of g wrt pi and tau
(f) <b>f3deltatau</b>	1st derivatives of f wrt delta and tau
(f) <b>tph1</b>	inverse function for region 1: T(p,h)
(f) <b>tps1</b>	inverse function for region 1: T(p,s)
(f) <b>tph2</b>	reverse function for region 2: T(p,h)

## 1642 Modelica.Media.Water.IF97\_Utilsities.BaselIF97.Basic

(f) tps2a	reverse function for region 2a: T(p,s)
(f) tps2b	reverse function for region 2b: T(p,s)
(f) tps2c	reverse function for region 2c: T(p,s)
(f) tps2	reverse function for region 2: T(p,s)
(f) tsat	region 4 saturation temperature as a function of pressure
(f) dtsatofp	derivative of saturation temperature w.r.t. pressure
(f) tsat_der	derivative function for tsat
(f) psat	region 4 saturation pressure as a functionx of temperature
(f) dptofT	derivative of pressure wrt temperature along the saturation pressure curve
(f) psat_der	derivative function for psat
(f) p1_hs	pressure as a function of ehtnaly and entropy in region 1
(f) h2ab_s	boundary between regions 2a and 2b
(f) p2a_hs	pressure as a function of enthalpy and entropy in subregion 2a
(f) p2b_hs	pressure as a function of enthalpy and entropy in subregion 2a
(f) p2c_hs	pressure as a function of enthalpy and entropy in subregion 2c
(f) h3ab_p	ergion 3 a b boundary for pressure/enthalpy
(f) T3a_ph	Region 3 a: inverse function T(p,h)
(f) T3b_ph	Region 3 b: inverse function T(p,h)
(f) v3a_ph	Region 3 a: inverse function v(p,h)
(f) v3b_ph	Region 3 b: inverse function v(p,h)
(f) T3a_ps	Region 3 a: inverse function T(p,s)
(f) T3b_ps	Region 3 b: inverse function T(p,s)
(f) v3a_ps	Region 3 a: inverse function v(p,s)
(f) v3b_ps	Region 3 b: inverse function v(p,s)

## Modelica.Media.Water.IF97\_Utilsities.BaselIF97.Basic.g1

Gibbs function for region 1: g(p,T)



### Inputs

Name	Default	Description
p		pressure [Pa]
T		temperature (K) [K]

### Outputs

Name	Description
g	dimensionless Gibbs funcion and dervatives wrt pi and tau

**Modelica.Media.Water.IF97\_Utilsities.BaselF97.Basic.g2**

Gibbs function for region 2:  $g(p,T)$

**Inputs**

Name	Default	Description
p		pressure [Pa]
T		temperature (K) [K]

**Outputs**

Name	Description
g	dimensionless Gibbs function and derivatives wrt pi and tau

**Modelica.Media.Water.IF97\_Utilsities.BaselF97.Basic.g2metastable**

Gibbs function for metastable part of region 2:  $g(p,T)$

**Inputs**

Name	Default	Description
p		pressure [Pa]
T		temperature (K) [K]

**Outputs**

Name	Description
g	dimensionless Gibbs function and derivatives wrt pi and tau

**Modelica.Media.Water.IF97\_Utilsities.BaselF97.Basic.f3**

Helmholtz function for region 3:  $f(d,T)$

**Inputs**

Name	Default	Description
d		density [kg/m <sup>3</sup> ]
T		temperature (K) [K]

**Outputs**

Name	Description
f	dimensionless Helmholtz function and derivatives wrt delta and tau

**Modelica.Media.Water.IF97\_Utilsities.BaselF97.Basic.g5**

base function for region 5:  $g(p,T)$

---

## 1644 Modelica.Media.Water.IF97\_Utils.BaselF97.Basic.g5

---

### Inputs

Name	Default	Description
p		pressure [Pa]
T		temperature (K) [K]

### Outputs

Name	Description
g	dimensionless Gibbs function and derivatives wrt pi and tau

---

## Modelica.Media.Water.IF97\_Utils.BaselF97.Basic.gibbs



Gibbs function for region 1, 2 or 5:  $g(p, T, \text{region})$

### Inputs

Name	Default	Description
p		pressure [Pa]
T		temperature (K) [K]
region		IF97 region, 1, 2 or 5

### Outputs

Name	Description
g	dimensionless Gibbs function

---

## Modelica.Media.Water.IF97\_Utils.BaselF97.Basic.g1pitau



derivative of  $g$  wrt  $\pi$  and  $\tau$

### Inputs

Name	Default	Description
p		pressure [Pa]
T		temperature (K) [K]

### Outputs

Name	Description
pi	dimensionless pressure
tau	dimensionless temperature
gpi	dimensionless derivative of Gibbs function wrt pi
gtau	dimensionless derivative of Gibbs function wrt tau

---

## Modelica.Media.Water.IF97\_Utils.BaselF97.Basic.g2pitau



derivative of  $g$  wrt  $\pi$  and  $\tau$

## Inputs

Name	Default	Description
p		pressure [Pa]
T		temperature (K) [K]

## Outputs

Name	Description
pi	dimensionless pressure
tau	dimensionless temperature
gpi	dimensionless derivative of Gibbs function wrt pi
gtau	dimensionless derivative of Gibbs function wrt tau

## Modelica.Media.Water.IF97\_Utilsities.BaselF97.Basic.g5pitau



derivative of g wrt pi and tau

## Inputs

Name	Default	Description
p		pressure [Pa]
T		temperature (K) [K]

## Outputs

Name	Description
pi	dimensionless pressure
tau	dimensionless temperature
gpi	dimensionless derivative of Gibbs function wrt pi
gtau	dimensionless derivative of Gibbs function wrt tau

## Modelica.Media.Water.IF97\_Utilsities.BaselF97.Basic.f3deltatau



1st derivatives of f wrt delta and tau

## Inputs

Name	Default	Description
d		density [kg/m3]
T		temperature (K) [K]

## Outputs

Name	Description
delta	dimensionless density
tau	dimensionless temperature
fdelta	dimensionless derivative of Helmholtz function wrt delta
ftau	dimensionless derivative of Helmholtz function wrt tau

---

## 1646 Modelica.Media.Water.IF97\_Utilities.BaselF97.Basic.tph1

---

### Modelica.Media.Water.IF97\_Utilities.BaselF97.Basic.tph1

inverse function for region 1:  $T(p,h)$

#### Inputs

Name	Default	Description
p		pressure [Pa]
h		specific enthalpy [J/kg]

#### Outputs

Name	Description
T	temperature (K) [K]



---

## Modelica.Media.Water.IF97\_Utilities.BaselF97.Basic.tps1

---

inverse function for region 1:  $T(p,s)$

#### Inputs

Name	Default	Description
p		pressure [Pa]
s		specific entropy [J/(kg.K)]

#### Outputs

Name	Description
T	temperature (K) [K]



---

## Modelica.Media.Water.IF97\_Utilities.BaselF97.Basic.tph2

---

reverse function for region 2:  $T(p,h)$

#### Inputs

Name	Default	Description
p		pressure [Pa]
h		specific enthalpy [J/kg]

#### Outputs

Name	Description
T	temperature (K) [K]



---

## Modelica.Media.Water.IF97\_Utilities.BaselF97.Basic.tps2a

---

reverse function for region 2a:  $T(p,s)$



## Inputs

Name	Default	Description
p		pressure [Pa]
s		specific entropy [J/(kg.K)]

## Outputs

Name	Description
T	temperature (K) [K]

## Modelica.Media.Water.IF97\_Utilsities.BaselF97.Basic.tps2b



reverse function for region 2b: T(p,s)

## Inputs

Name	Default	Description
p		pressure [Pa]
s		specific entropy [J/(kg.K)]

## Outputs

Name	Description
T	temperature (K) [K]

## Modelica.Media.Water.IF97\_Utilsities.BaselF97.Basic.tps2c



reverse function for region 2c: T(p,s)

## Inputs

Name	Default	Description
p		pressure [Pa]
s		specific entropy [J/(kg.K)]

## Outputs

Name	Description
T	temperature (K) [K]

## Modelica.Media.Water.IF97\_Utilsities.BaselF97.Basic.tps2



reverse function for region 2: T(p,s)

## Inputs

Name	Default	Description
p		pressure [Pa]
s		specific entropy [J/(kg.K)]

## Outputs

Name	Description
T	temperature (K) [K]

---

## Modelica.Media.Water.IF97\_Utilsities.BaselF97.Basic.tsat

region 4 saturation temperature as a function of pressure



## Inputs

Name	Default	Description
p		pressure [Pa]

---

## Outputs

Name	Description
t_sat	temperature [K]

---

## Modelica.Media.Water.IF97\_Utilsities.BaselF97.Basic.dtsatofp

derivative of saturation temperature w.r.t. pressure



## Inputs

Name	Default	Description
p		pressure [Pa]

---

## Outputs

Name	Description
dtsat	derivative of T w.r.t. p

---

## Modelica.Media.Water.IF97\_Utilsities.BaselF97.Basic.tsat\_der

derivative function for tsat



## Inputs

Name	Default	Description
p		pressure [Pa]
der_p		pressure derivative

---

## Outputs

Name	Description
der_tsat	temperature derivative

---

**Modelica.Media.Water.IF97\_Utilsities.BaselF97.Basic.psat**

region 4 saturation pressure as a function of temperature

**Inputs**

Name	Default	Description
T		temperature (K) [K]

**Outputs**

Name	Description
p_sat	pressure [Pa]

**Modelica.Media.Water.IF97\_Utilsities.BaselF97.Basic.dptofT**

derivative of pressure wrt temperature along the saturation pressure curve

**Inputs**

Name	Default	Description
T		temperature (K) [K]

**Outputs**

Name	Description
dpt	temperature derivative of pressure

**Modelica.Media.Water.IF97\_Utilsities.BaselF97.Basic.psat\_der**

derivative function for psat

**Inputs**

Name	Default	Description
T		temperature (K) [K]
der_T		temperature derivative

**Outputs**

Name	Description
der_psat	pressure

**Modelica.Media.Water.IF97\_Utilsities.BaselF97.Basic.p1\_hs**

pressure as a function of enthalpy and entropy in region 1

**Information**

Equation number 1 from:

The International Association for the Properties of Water and Steam  
Gaithersburg, Maryland, USA

## **1650 Modelica.Media.Water.IF97\_Utilsities.BaselF97.Basic.p1\_hs**

---

September 2001

Supplementary Release on Backward Equations for Pressure as a Function of Enthalpy and Entropy p(h,s) to the IAPWS Industrial Formulation 1997 for the Thermodynamic Properties of Water and Steam

### **Inputs**

Name	Default	Description
h		specific enthalpy [J/kg]
s		specific entropy [J/(kg.K)]

### **Outputs**

Name	Description
p	Pressure [Pa]

---

## **Modelica.Media.Water.IF97\_Utilsities.BaselF97.Basic.h2ab\_s**

**boundary between regions 2a and 2b**



### **Information**

Equation number 2 from:

The International Association for the Properties of Water and Steam

Gaithersburg, Maryland, USA

September 2001

Supplementary Release on Backward Equations for Pressure as a Function of Enthalpy and Entropy p(h,s) to the IAPWS Industrial Formulation 1997 for the Thermodynamic Properties of Water and Steam

### **Inputs**

Name	Default	Description
s		Entropy [J/(kg.K)]

### **Outputs**

Name	Description
h	Enthalpy [J/kg]

---

## **Modelica.Media.Water.IF97\_Utilsities.BaselF97.Basic.p2a\_hs**

**pressure as a function of enthalpy and entropy in subregion 2a**



### **Information**

Equation number 3 from:

The International Association for the Properties of Water and Steam

Gaithersburg, Maryland, USA

September 2001

Supplementary Release on Backward Equations for Pressure as a Function of Enthalpy and Entropy p(h,s)

to the IAPWS Industrial Formulation 1997 for the Thermodynamic Properties of Water and Steam

## Inputs

Name	Default	Description
h		specific enthalpy [J/kg]
s		specific entropy [J/(kg.K)]

## Outputs

Name	Description
p	Pressure [Pa]

---

### **Modelica.Media.Water.IF97\_Utilsities.BaselF97.Basic.p2b\_hs**

**pressure as a function of enthalpy and entropy in subregion 2a**



## Information

Equation number 4 from:

The International Association for the Properties of Water and Steam  
Gaithersburg, Maryland, USA

September 2001

Supplementary Release on Backward Equations for Pressure as a Function of Enthalpy and Entropy p(h,s)  
to the IAPWS Industrial Formulation 1997 for the Thermodynamic Properties of Water and Steam

## Inputs

Name	Default	Description
h		specific enthalpy [J/kg]
s		specific entropy [J/(kg.K)]

## Outputs

Name	Description
p	Pressure [Pa]

---

### **Modelica.Media.Water.IF97\_Utilsities.BaselF97.Basic.p2c\_hs**

**pressure as a function of enthalpy and entropy in subregion 2c**



## Information

Equation number 5 from:

The International Association for the Properties of Water and Steam  
Gaithersburg, Maryland, USA

September 2001

Supplementary Release on Backward Equations for Pressure as a Function of Enthalpy and Entropy p(h,s)  
to the IAPWS Industrial Formulation 1997 for the Thermodynamic Properties of Water and Steam

---

## 1652 Modelica.Media.Water.IF97\_Utilsities.BaselF97.Basic.p2c\_hs

---

### Inputs

Name	Default	Description
h		specific enthalpy [J/kg]
s		specific entropy [J/(kg.K)]

### Outputs

Name	Description
p	Pressure [Pa]

---

## Modelica.Media.Water.IF97\_Utilsities.BaselF97.Basic.h3ab\_p

Region 3 a b boundary for pressure/enthalpy



### Information

Equation number 1 from:

[1] The international Association for the Properties of Water and Steam  
Vejle, Denmark  
August 2003

Supplementary Release on Backward Equations for the Fucnctions T(p,h), v(p,h) and T(p,s),  
v(p,s) for Region 3 of the IAPWS Industrial Formulation 1997 for the Thermodynamic Properties of  
Water and Steam

### Inputs

Name	Default	Description
p		Pressure [Pa]

### Outputs

Name	Description
h	Enthalpy [J/kg]

---

## Modelica.Media.Water.IF97\_Utilsities.BaselF97.Basic.T3a\_ph

Region 3 a: inverse function T(p,h)



### Information

Equation number 2 from:

[1] The international Association for the Properties of Water and Steam  
Vejle, Denmark  
August 2003

Supplementary Release on Backward Equations for the Fucnctions T(p,h), v(p,h) and T(p,s),  
v(p,s) for Region 3 of the IAPWS Industrial Formulation 1997 for the Thermodynamic Properties of  
Water and Steam

### Inputs

Name	Default	Description
p		Pressure [Pa]

h	specific enthalpy [J/kg]
---	--------------------------

## Outputs

Name	Description
T	Temperature [K]

## Modelica.Media.Water.IF97\_Utilsities.BaselF97.Basic.T3b\_ph

Region 3 b: inverse function T(p,h)



## Information

Equation number 3 from:

[1] The international Association for the Properties of Water and Steam  
Vejle, Denmark  
August 2003

Supplementary Release on Backward Equations for the Fucnctions T(p,h), v(p,h) and T(p,s),  
v(p,s) for Region 3 of the IAPWS Industrial Formulation 1997 for the Thermodynamic Properties of  
Water and Steam

## Inputs

Name	Default	Description
p		Pressure [Pa]
h		specific enthalpy [J/kg]

## Outputs

Name	Description
T	Temperature [K]

## Modelica.Media.Water.IF97\_Utilsities.BaselF97.Basic.v3a\_ph

Region 3 a: inverse function v(p,h)



## Information

Equation number 4 from:

[1] The international Association for the Properties of Water and Steam  
Vejle, Denmark  
August 2003

Supplementary Release on Backward Equations for the Fucnctions T(p,h), v(p,h) and T(p,s),  
v(p,s) for Region 3 of the IAPWS Industrial Formulation 1997 for the Thermodynamic Properties of  
Water and Steam

## Inputs

Name	Default	Description
p		Pressure [Pa]
h		specific enthalpy [J/kg]

## 1654 Modelica.Media.Water.IF97\_Utilsities.BaselF97.Basic.v3a\_ph

---

### Outputs

Name	Description
v	specific volume [m3]

---

## Modelica.Media.Water.IF97\_Utilsities.BaselF97.Basic.v3b\_ph

Region 3 b: inverse function v(p,h)



### Information

Equation number 5 from:

[1] The international Association for the Properties of Water and Steam  
Vejle, Denmark  
August 2003

Supplementary Release on Backward Equations for the Fucnctions T(p,h), v(p,h) and T(p,s),  
v(p,s) for Region 3 of the IAPWS Industrial Formulation 1997 for the Thermodynamic Properties of  
Water and Steam

### Inputs

Name	Default	Description
p		Pressure [Pa]
h		specific enthalpy [J/kg]

### Outputs

Name	Description
v	specific volume [m3]

---

## Modelica.Media.Water.IF97\_Utilsities.BaselF97.Basic.T3a\_ps

Region 3 a: inverse function T(p,s)



### Information

Equation number 6 from:

[1] The international Association for the Properties of Water and Steam  
Vejle, Denmark  
August 2003

Supplementary Release on Backward Equations for the Fucnctions T(p,h), v(p,h) and T(p,s),  
v(p,s) for Region 3 of the IAPWS Industrial Formulation 1997 for the Thermodynamic Properties of  
Water and Steam

### Inputs

Name	Default	Description
p		Pressure [Pa]
s		specific entropy [J/(kg.K)]

### Outputs

Name	Description

---

T	Temperature [K]
---	-----------------

---

**Modelica.Media.Water.IF97\_Utilsities.BaseIF97.Basic.T3b\_ps****Region 3 b: inverse function T(p,s)****Information**

Equation number 7 from:

[1] The international Association for the Properties of Water and Steam  
Vejle, Denmark  
August 2003

Supplementary Release on Backward Equations for the Fucnctions  $T(p,h)$ ,  $v(p,h)$  and  $T(p,s)$ ,  
 $v(p,s)$  for Region 3 of the IAPWS Industrial Formulation 1997 for the Thermodynamic Properties of  
Water and Steam

**Inputs**

Name	Default	Description
p		Pressure [Pa]
s		specific entropy [J/(kg.K)]

**Outputs**

Name	Description
T	Temperature [K]

---

**Modelica.Media.Water.IF97\_Utilsities.BaseIF97.Basic.v3a\_ps****Region 3 a: inverse function v(p,s)****Information**

Equation number 8 from:

[1] The international Association for the Properties of Water and Steam  
Vejle, Denmark  
August 2003

Supplementary Release on Backward Equations for the Fucnctions  $T(p,h)$ ,  $v(p,h)$  and  $T(p,s)$ ,  
 $v(p,s)$  for Region 3 of the IAPWS Industrial Formulation 1997 for the Thermodynamic Properties of  
Water and Steam

**Inputs**

Name	Default	Description
p		Pressure [Pa]
s		specific entropy [J/(kg.K)]

**Outputs**

Name	Description
v	specific volume [m <sup>3</sup> ]

## 1656 Modelica.Media.Water.IF97\_Utilsities.BaselF97.Basic.v3b\_ps

---

### Modelica.Media.Water.IF97\_Utilsities.BaselF97.Basic.v3b\_ps



Region 3 b: inverse function v(p,s)

#### Information

Equation number 9 from:

[1] The international Association for the Properties of Water and Steam  
Vejle, Denmark  
August 2003

Supplementary Release on Backward Equations for the Fucnctions T(p,h), v(p,h) and T(p,s),  
v(p,s) for Region 3 of the IAPWS Industrial Formulation 1997 for the Thermodynamic Properties of  
Water and Steam

#### Inputs

Name	Default	Description
p		Pressure [Pa]
s		specific entropy [J/(kg.K)]

#### Outputs

Name	Description
v	specific volume [m <sup>3</sup> ]

---

### Modelica.Media.Water.IF97\_Utilsities.BaselF97.IceBoundaries

the melting line and sublimation line curves from IAPWS

#### Information

The International Association for the Properties of Water and Steam

Milan, Italy

September 1993

Release on the Pressure along the Melting and the Sublimation Curves of Ordinary Water Substance

#### Package Content

Name	Description
(f) pmlcel_T	Melting pressure of ice I (temperature range from 273.16 to 251.165 K)
(f) pmlcelll_T	Melting pressure of ice III (temperature range from 251.165 to 256.164 K)
(f) pmlceV_T	Melting pressure of ice V (temperature range from 256.164 to 273.31 K)
(f) sublimationPressure_T	Sublimation pressure, valid from 190 to 273.16 K

---

### Modelica.Media.Water.IF97\_Utilsities.BaselF97.IceBoundaries.pmlcel\_T



Melting pressure of ice I (temperature range from 273.16 to 251.165 K)

#### Information

Equation 1 from:

The International Association for the Properties of Water and Steam

Milan, Italy  
September 1993  
Release on the Pressure along the Melting and the Sublimation Curves of Ordinary Water Substance

### Inputs

Name	Default	Description
T		Temperature [K]

### Outputs

Name	Description
pm	Melting pressure of icel(for T from 273.16 to 251.165 K) [Pa]

---

### Modelica.Media.Water.IF97\_Utilsities.BaselF97.IceBoundaries.pmlcellIII\_T

Melting pressure of ice III (temperature range from 251.165 to 256.164 K)



### Information

Equation 2 from:  
The International Association for the Properties of Water and Steam  
Milan, Italy  
September 1993  
Release on the Pressure along the Melting and the Sublimation Curves of Ordinary Water Substance

### Inputs

Name	Default	Description
T		Temperature [K]

### Outputs

Name	Description
pm	Melting pressure of icellIII(for T from 251.165 to 256.164 K) [Pa]

---

### Modelica.Media.Water.IF97\_Utilsities.BaselF97.IceBoundaries.pmlceV\_T

Melting pressure of ice V (temperature range from 256.164 to 273.31 K)



### Information

Equation 3 from:  
The International Association for the Properties of Water and Steam  
Milan, Italy  
September 1993  
Release on the Pressure along the Melting and the Sublimation Curves of Ordinary Water Substance

## Inputs

Name	Default	Description
T		Temperature [K]

## Outputs

Name	Description
pm	Melting pressure of iceV(for T from 256.164 to 273.31 K) [Pa]

---

## Modelica.Media.Water.IF97\_Utilsities.BaselF97.IceBoundaries.sublimationPressure\_T

Sublimation pressure, valid from 190 to 273.16 K



## Information

Equation 6 from:

The International Association for the Properties of Water and Steam

Milan, Italy

September 1993

Release on the Pressure along the Melting and the Sublimation Curves of Ordinary Water Substance

## Inputs

Name	Default	Description
T		Temperature [K]

## Outputs

Name	Description
psubl	sublimation pressure (for T from 190 to 273.16) [Pa]

---

## Modelica.Media.Water.IF97\_Utilsities.BaselF97.Transport

transport properties for water according to IAPWS/IF97

## Information

### Package description

### Package contents

- Function **visc\_dTp** implements a function to compute the industrial formulation of the dynamic viscosity of water as a function of density and temperature. The details are described in the document [visc.pdf](#).
- Function **cond\_dTp** implements a function to compute the industrial formulation of the thermal conductivity of water as a function of density, temperature and pressure. **Important note:** Obviously only two of the three inputs are really needed, but using three inputs speeds up the computation and the three variables are known in most models anyways. The inputs d,T and p have to be consistent. The details are described in the document [surf.pdf](#).

- Function **surfaceTension** implements a function to compute the surface tension between vapour and liquid water as a function of temperature. The details are described in the document [thcond.pdf](#).

### Version Info and Revision history

- First implemented: October, 2002 by Hubertus Tummescheit

*Authors:* Hubertus Tummescheit and Jonas Eborn  
Lund University

Department of Automatic Control  
Box 118, 22100 Lund, Sweden  
email: hubertus@control.lth.se

- Initial version: October 2002

### Package Content

Name	Description
(f) <b>visc_dTp</b>	dynamic viscosity $\eta(d,T,p)$ , industrial formulation
(f) <b>cond_dTp</b>	Thermal conductivity $\lambda(d,T,p)$ (industrial use version) only in one-phase region
(f) <b>surfaceTension</b>	surface tension in region 4 between steam and water

---

### Modelica.Media.Water.IF97\_Utilsities.BaselF97.Transport.**visc\_dTp**

dynamic viscosity  $\eta(d,T,p)$ , industrial formulation



#### Inputs

Name	Default	Description
d		density [kg/m <sup>3</sup> ]
T		temperature (K) [K]
p		pressure (only needed for region of validity) [Pa]
phase	0	2 for two-phase, 1 for one-phase, 0 if not known

#### Outputs

Name	Description
eta	dynamic viscosity [Pa.s]

---

### Modelica.Media.Water.IF97\_Utilsities.BaselF97.Transport.**cond\_dTp**

Thermal conductivity  $\lambda(d,T,p)$  (industrial use version) only in one-phase region



#### Inputs

Name	Default	Description
d		density [kg/m <sup>3</sup> ]
T		temperature (K) [K]
p		pressure [Pa]

## 1660 Modelica.Media.Water.IF97\_Utils.BaselF97.Transport.cond\_dTp

---

phase	0	2 for two-phase, 1 for one-phase, 0 if not known
industrialMethod	true	if true, the industrial method is used, otherwise the scientific one

### Outputs

Name	Description
lambda	thermal conductivity [W/(m.K)]

---

## Modelica.Media.Water.IF97\_Utils.BaselF97.Transport.SurfaceTension

surface tension in region 4 between steam and water



### Inputs

Name	Default	Description
T		temperature (K) [K]

### Outputs

Name	Description
sigma	surface tension in SI units [N/m]

---

## Modelica.Media.Water.IF97\_Utils.BaselF97.Isentropic

functions for calculating the isentropic enthalpy from pressure p and specific entropy s

### Information

#### Package description

#### Package contents

- Function **hofpT1** computes  $h(p,T)$  in region 1.
- Function **handsofpT1** computes  $(s,h)=f(p,T)$  in region 1, needed for two-phase properties.
- Function **hofps1** computes  $h(p,s)$  in region 1.
- Function **hofpT2** computes  $h(p,T)$  in region 2.
- Function **handsofpT2** computes  $(s,h)=f(p,T)$  in region 2, needed for two-phase properties.
- Function **hofps2** computes  $h(p,s)$  in region 2.
- Function **hofdT3** computes  $h(d,T)$  in region 3.
- Function **hofpsdt3** computes  $h(p,s,d\text{guess},T\text{guess})$  in region 3, where dguess and Tguess are initial guess values for the density and temperature consistent with p and s.
- Function **hofps4** computes  $h(p,s)$  in region 4.
- Function **hofpT5** computes  $h(p,T)$  in region 5.
- Function **water\_hisentropic** computes  $h(p,s,\text{phase})$  in all regions. The phase input is needed due to discontinuous derivatives at the phase boundary.
- Function **water\_hisentropic\_dyn** computes  $h(p,s,d\text{guess},T\text{guess},\text{phase})$  in all regions. The phase input is needed due to discontinuous derivatives at the phase boundary. Tguess and dguess are initial guess values for the density and temperature consistent with p and s. This function should be preferred in dynamic simulations where good guesses are often available.

#### Version Info and Revision history

- First implemented: July, 2000 by Hubertus Tummescheit

*Author: Hubertus Tummescheit,  
Lund University  
Department of Automatic Control  
Box 118, 22100 Lund, Sweden  
email: hubertus@control.lth.se*

- Initial version: July 2000
- Documentation added: December 2002

## Package Content

Name	Description
(f) hofpT1	intermediate function for isentropic specific enthalpy in region 1
(f) handsofpT1	special function for specific enthalpy and specific entropy in region 1
(f) hofps1	function for isentropic specific enthalpy in region 1
(f) hofpT2	intermediate function for isentropic specific enthalpy in region 2
(f) handsofpT2	function for isentropic specific enthalpy and specific entropy in region 2
(f) hofps2	function for isentropic specific enthalpy in region 2
(f) hofdT3	function for isentropic specific enthalpy in region 3
(f) hofps3	isentropic specific enthalpy in region 3 $h(p,s)$
(f) hofpsdt3	isentropic specific enthalpy in region 3 $h(p,s)$ with given good guess in d and T
(f) hofps4	isentropic specific enthalpy in region 4 $h(p,s)$
(f) hofpT5	specific enthalpy in region 5 $h(p,T)$
(f) water_hisentropic	isentropic specific enthalpy from p,s (preferably use <code>water_hisentropic_dyn</code> in dynamic simulation!)
(f) water_hisentropic_dyn	isentropic specific enthalpy from p,s and good guesses of d and T

### Modelica.Media.Water.IF97\_Utils.BaselF97.Isentropic.hofpT1

intermediate function for isentropic specific enthalpy in region 1

#### Inputs

Name	Default	Description
p		pressure [Pa]
T		temperature (K) [K]

#### Outputs

Name	Description
h	specific enthalpy [J/kg]



---

**1662 Modelica.Media.Water.IF97\_Utilsities.BaselF97.Isentropic.hansofpT1****Modelica.Media.Water.IF97\_Utilsities.BaselF97.Isentropic.hansofpT1**

special function for specific enthalpy and specific entropy in region 1

**Inputs**

Name	Default	Description
p		pressure [Pa]
T		temperature (K) [K]

**Outputs**

Name	Description
h	specific enthalpy [J/kg]
s	specific entropy [J/(kg.K)]

---

**Modelica.Media.Water.IF97\_Utilsities.BaselF97.Isentropic.hofps1**

function for isentropic specific enthalpy in region 1

**Inputs**

Name	Default	Description
p		pressure [Pa]
s		specific entropy [J/(kg.K)]

**Outputs**

Name	Description
h	specific enthalpy [J/kg]

---

**Modelica.Media.Water.IF97\_Utilsities.BaselF97.Isentropic.hofpT2**

intermediate function for isentropic specific enthalpy in region 2

**Inputs**

Name	Default	Description
p		pressure [Pa]
T		temperature (K) [K]

**Outputs**

Name	Description
h	specific enthalpy [J/kg]

---

**Modelica.Media.Water.IF97\_Utilsities.BaselF97.Isentropic.hansofpT2**

function for isentropic specific enthalpy and specific entropy in region 2

## Inputs

Name	Default	Description
p		pressure [Pa]
T		temperature (K) [K]

## Outputs

Name	Description
h	specific enthalpy [J/kg]
s	specific entropy [J/(kg.K)]

## Modelica.Media.Water.IF97\_Utilsities.BaselF97.Isentropic.hofps2

function for isentropic specific enthalpy in region 2



## Inputs

Name	Default	Description
p		pressure [Pa]
s		specific entropy [J/(kg.K)]

## Outputs

Name	Description
h	specific enthalpy [J/kg]

## Modelica.Media.Water.IF97\_Utilsities.BaselF97.Isentropic.hofdT3

function for isentropic specific enthalpy in region 3



## Inputs

Name	Default	Description
d		density [kg/m3]
T		temperature (K) [K]

## Outputs

Name	Description
h	specific enthalpy [J/kg]

## Modelica.Media.Water.IF97\_Utilsities.BaselF97.Isentropic.hofps3

isentropic specific enthalpy in region 3 h(p,s)



## Inputs

Name	Default	Description
p		pressure [Pa]

## 1664 Modelica.Media.Water.IF97\_Utils.BaselF97.Isentropic.hofps3

---

s	specific entropy [J/(kg.K)]
---	-----------------------------

### Outputs

Name	Description
h	specific enthalpy [J/kg]

---

## Modelica.Media.Water.IF97\_Utils.BaselF97.Isentropic.hofpsdt3

isentropic specific enthalpy in region 3  $h(p,s)$  with given good guess in d and T



### Inputs

Name	Default	Description
p		pressure [Pa]
s		specific entropy [J/(kg.K)]
dguess		good guess density, e.g. from adjacent volume [kg/m <sup>3</sup> ]
Tguess		good guess temperature, e.g. from adjacent volume [K]
delp	IterationData.DELP	relative error in p [Pa]
dels	IterationData.DELS	relative error in s [J/(kg.K)]

### Outputs

Name	Description
h	specific enthalpy [J/kg]

---

## Modelica.Media.Water.IF97\_Utils.BaselF97.Isentropic.hofps4

isentropic specific enthalpy in region 4  $h(p,s)$



### Inputs

Name	Default	Description
p		pressure [Pa]
s		specific entropy [J/(kg.K)]

### Outputs

Name	Description
h	specific enthalpy [J/kg]

---

## Modelica.Media.Water.IF97\_Utils.BaselF97.Isentropic.hofpT5

specific enthalpy in region 5  $h(p,T)$



### Inputs

Name	Default	Description
p		pressure [Pa]
T		temperature (K) [K]

## Outputs

Name	Description
h	specific enthalpy [J/kg]

---

## Modelica.Media.Water.IF97\_Utilsities.BaseIF97.Isentropic.water\_hisentropic

isentropic specific enthalpy from p,s (preferably use water\_hisentropic\_dyn in dynamic simulation!)



## Inputs

Name	Default	Description
p		pressure [Pa]
s		specific entropy [J/(kg.K)]
phase	0	phase: 2 for two-phase, 1 for one phase, 0 if unknown

## Outputs

Name	Description
h	specific enthalpy [J/kg]

---

## Modelica.Media.Water.IF97\_Utilsities.BaseIF97.Isentropic.water\_hisentropic\_dyn

isentropic specific enthalpy from p,s and good guesses of d and T



## Inputs

Name	Default	Description
p		pressure [Pa]
s		specific entropy [J/(kg.K)]
dguess		good guess density, e.g. from adjacent volume [kg/m <sup>3</sup> ]
Tguess		good guess temperature, e.g. from adjacent volume [K]
phase		1 for one phase, 2 for two phase

## Outputs

Name	Description
h	specific enthalpy [J/kg]

---

## Modelica.Media.Water.IF97\_Utilsities.BaseIF97.Inverses

efficient inverses for selected pairs of variables

## Information

### Package description

### Package contents

- Function **fixdT** constrains density and temperature to allowed region
- Function **dofp13** computes d as a function of p at boundary between regions 1 and 3
- Function **dofp23** computes d as a function of p at boundary between regions 2 and 3
- Function **dofpt3** iteration to compute d as a function of p and T in region 3
- Function **dtofph3** iteration to compute d and T as a function of p and h in region 3
- Function **dtofps3** iteration to compute d and T as a function of p and s in region 3
- Function **dtofpsdt3** iteration to compute d and T as a function of p and s in region 3, with initial guesses
- Function **pofdt125** iteration to compute p as a function of p and T in regions 1, 2 and 5
- Function **tolph5** iteration to compute T as a function of p and h in region 5
- Function **tlops5** iteration to compute T as a function of p and s in region 5
- Function **toppst5** iteration to compute T as a function of p and s in region 5, with initial guess in T
- Function

### Version Info and Revision history

- First implemented: July, 2000 by Hubertus Tummescheit

*Author: Hubertus Tummescheit,*

*Lund University*

*Department of Automatic Control*

*Box 118, 22100 Lund, Sweden*

*email: hubertus@control.lth.se*

- Initial version: July 2000
- Documentation added: December 2002

### Package Content

Name	Description
(f) <b>fixdT</b>	region limits for inverse iteration in region 3
(f) <b>dofp13</b>	density at the boundary between regions 1 and 3
(f) <b>dofp23</b>	density at the boundary between regions 2 and 3
(f) <b>dofpt3</b>	inverse iteration in region 3: $(d) = f(p,T)$
(f) <b>dtofph3</b>	inverse iteration in region 3: $(d,T) = f(p,h)$
(f) <b>dtofps3</b>	inverse iteration in region 3: $(d,T) = f(p,s)$
(f) <b>dtofpsdt3</b>	inverse iteration in region 3: $(d,T) = f(p,s)$
(f) <b>pofdt125</b>	inverse iteration in region 1,2 and 5: $p = g(d,T)$
(f) <b>tolph5</b>	inverse iteration in region 5: $(p,T) = f(p,h)$
(f) <b>tlops5</b>	inverse iteration in region 5: $(p,T) = f(p,s)$
(f) <b>toppst5</b>	inverse iteration in region 5: $(p,T) = f(p,s)$

**Modelica.Media.Water.IF97\_Utilsities.BaselIF97.Inverses.fixdT**

region limits for inverse iteration in region 3

**Inputs**

Name	Default	Description
din		density [kg/m3]
Tin		temperature [K]

**Outputs**

Name	Description
dout	density [kg/m3]
Tout	temperature [K]

**Modelica.Media.Water.IF97\_Utilsities.BaselIF97.Inverses.dofp13**

density at the boundary between regions 1 and 3

**Inputs**

Name	Default	Description
p		pressure [Pa]

**Outputs**

Name	Description
d	density [kg/m3]

**Modelica.Media.Water.IF97\_Utilsities.BaselIF97.Inverses.dofp23**

density at the boundary between regions 2 and 3

**Inputs**

Name	Default	Description
p		pressure [Pa]

**Outputs**

Name	Description
d	density [kg/m3]

**Modelica.Media.Water.IF97\_Utilsities.BaselIF97.Inverses.dofpt3**inverse iteration in region 3:  $(d) = f(p,T)$ **Inputs**

Name	Default	Description

## 1668 Modelica.Media.Water.IF97\_Utils.BaselF97.Inverses.dofpt3

---

p	pressure [Pa]
T	temperature (K) [K]
delp	iteration converged if ( $p - \text{pre}(p) < \text{delp}$ ) [Pa]

### Outputs

Name	Description
d	density [kg/m <sup>3</sup> ]
error	error flag: iteration failed if different from 0

---

## Modelica.Media.Water.IF97\_Utils.BaselF97.Inverses.dtofph3



inverse iteration in region 3:  $(d, T) = f(p, h)$

### Inputs

Name	Default	Description
p		pressure [Pa]
h		specific enthalpy [J/kg]
delp		iteration accuracy [Pa]
delh		iteration accuracy [J/kg]

### Outputs

Name	Description
d	density [kg/m <sup>3</sup> ]
T	temperature (K) [K]
error	error flag: iteration failed if different from 0

---

## Modelica.Media.Water.IF97\_Utils.BaselF97.Inverses.dtofps3



inverse iteration in region 3:  $(d, T) = f(p, s)$

### Inputs

Name	Default	Description
p		pressure [Pa]
s		specific entropy [J/(kg.K)]
delp		iteration accuracy [Pa]
dels		iteration accuracy [J/(kg.K)]

### Outputs

Name	Description
d	density [kg/m <sup>3</sup> ]
T	temperature (K) [K]
error	error flag: iteration failed if different from 0

**Modelica.Media.Water.IF97\_Utilities.BaselF97.Inverses.dtofpsdt3**

inverse iteration in region 3:  $(d, T) = f(p, s)$

**Inputs**

Name	Default	Description
p		pressure [Pa]
s		specific entropy [J/(kg.K)]
dguess		guess density, e.g. from adjacent volume [kg/m3]
Tguess		guess temperature, e.g. from adjacent volume [K]
delp		iteration accuracy [Pa]
dels		iteration accuracy [J/(kg.K)]

**Outputs**

Name	Description
d	density [kg/m3]
T	temperature (K) [K]
error	error flag: iteration failed if different from 0

**Modelica.Media.Water.IF97\_Utilities.BaselF97.Inverses.pofdt125**

inverse iteration in region 1,2 and 5:  $p = g(d, T)$

**Inputs**

Name	Default	Description
d		density [kg/m3]
T		temperature (K) [K]
reldd		relative iteration accuracy of density [Pa]
region		region in IAPWS/IF97 in which inverse should be calculated

**Outputs**

Name	Description
p	pressure [Pa]
error	error flag: iteration failed if different from 0

**Modelica.Media.Water.IF97\_Utilities.BaselF97.Inverses.toph5**

inverse iteration in region 5:  $(p, T) = f(p, h)$

**Inputs**

Name	Default	Description
p		pressure [Pa]
h		specific enthalpy [J/kg]
reldh		iteration accuracy [J/kg]

## 1670 Modelica.Media.Water.IF97\_Utils.BaselF97.Inverses.tofph5

---

### Outputs

Name	Description
T	temperature (K) [K]
error	error flag: iteration failed if different from 0

---

## Modelica.Media.Water.IF97\_Utils.BaselF97.Inverses.tofps5



inverse iteration in region 5:  $(p, T) = f(p, s)$

### Inputs

Name	Default	Description
p		pressure [Pa]
s		specific entropy [J/(kg.K)]
relds		iteration accuracy [J/kg]

### Outputs

Name	Description
T	temperature (K) [K]
error	error flag: iteration failed if different from 0

---

## Modelica.Media.Water.IF97\_Utils.BaselF97.Inverses.tofpst5



inverse iteration in region 5:  $(p, T) = f(p, s)$

### Inputs

Name	Default	Description
p		pressure [Pa]
s		specific entropy [J/(kg.K)]
Tguess		guess temperature, e.g. from adjacent volume [K]
relds		iteration accuracy [J/(kg.K)]

### Outputs

Name	Description
T	temperature (K) [K]
error	error flag: iteration failed if different from 0

---

## Modelica.Media.Water.IF97\_Utils.BaselF97.ByRegion

simple explicit functions for one region only

### Information

#### Package description

Package ByRegion provides fast forward calls for dynamic property calculation records for all one phase

regions of IAPWS/IF97

### Package contents

- Function **waterR1\_pT** computes dynamic properties for region 1 using (p,T) as inputs
- Function **waterR2\_pT** computes dynamic properties for region 2 using (p,T) as inputs
- Function **waterR3\_dT** computes dynamic properties for region 3 using (d,T) as inputs
- Function **waterR5\_pT** computes dynamic properties for region 5 using (p,T) as inputs

### Version Info and Revision history

- First implemented: July, 2000 by Hubertus Tummescheit

*Author:* Hubertus Tummescheit,

Lund University

Department of Automatic Control

Box 118, 22100 Lund, Sweden

email: hubertus@control.lth.se

- Initial version: July 2000
- Documented and re-organized: January 2003

### Package Content

Name	Description
(f) <a href="#">waterR1_pT</a>	standard properties for region 1, (p,T) as inputs
(f) <a href="#">waterR2_pT</a>	standard properties for region 2, (p,T) as inputs
(f) <a href="#">waterR3_dT</a>	standard properties for region 3, (d,T) as inputs
(f) <a href="#">waterR5_pT</a>	standard properties for region 5, (p,T) as inputs

### [Modelica.Media.Water.IF97\\_Utilsities.BaseIF97.ByRegion.waterR1\\_pT](#)

standard properties for region 1, (p,T) as inputs



#### Inputs

Name	Default	Description
p		pressure [Pa]
T		temperature (K) [K]

#### Outputs

Name	Description
pro	thermodynamic property collection

### [Modelica.Media.Water.IF97\\_Utilsities.BaseIF97.ByRegion.waterR2\\_pT](#)

standard properties for region 2, (p,T) as inputs



---

## 1672 Modelica.Media.Water.IF97\_Utils.BaselF97.ByRegion.waterR2\_pT

---

### Inputs

Name	Default	Description
p		pressure [Pa]
T		temperature (K) [K]

### Outputs

Name	Description
pro	thermodynamic property collection

---

## Modelica.Media.Water.IF97\_Utils.BaselF97.ByRegion.waterR3\_dT

standard properties for region 3, (d,T) as inputs



### Inputs

Name	Default	Description
d		density [kg/m3]
T		temperature (K) [K]

### Outputs

Name	Description
pro	thermodynamic property collection

---

## Modelica.Media.Water.IF97\_Utils.BaselF97.ByRegion.waterR5\_pT

standard properties for region 5, (p,T) as inputs



### Inputs

Name	Default	Description
p		pressure [Pa]
T		temperature (K) [K]

### Outputs

Name	Description
pro	thermodynamic property collection

---

## Modelica.Media.Water.IF97\_Utils.BaselF97.TwoPhase

steam properties in the two-phase region and on the phase boundaries

### Information

#### Package description

Package TwoPhase provides functions to compute the steam properties in the two-phase region and on the phase boundaries

## Package contents

- Function **WaterLiq\_p** computes properties on the boiling boundary as a function of p
- Function **WaterVap\_p** computes properties on the dew line boundary as a function of p
- Function **WaterSat\_ph** computes properties on both phase boundaries and in the two phase region as a function of p
- Function **WaterR4\_ph** computes dynamic simulation properties in region 4 with (p,h) as inputs
- Function **WaterR4\_dT** computes dynamic simulation properties in region 4 with (d,T) as inputs

## Version Info and Revision history

- First implemented: July, 2000 by Hubertus Tummescheit

*Author: Hubertus Tummescheit,  
Lund University  
Department of Automatic Control  
Box 118, 22100 Lund, Sweden  
email: hubertus@control.lth.se*

- Initial version: July 2000
- Documented and re-organized: January 2003

## Package Content

Name	Description
(f) <b>waterLiq_p</b>	properties on the liquid phase boundary of region 4
(f) <b>waterVap_p</b>	properties on the vapour phase boundary of region 4
(f) <b>waterSat_ph</b>	Water saturation properties in the 2-phase region (4) as f(p,h)
(f) <b>waterR4_ph</b>	Water/Steam properties in region 4 of IAPWS/IF97 (two-phase)
(f) <b>waterR4_dT</b>	Water properties in region 4 as function of d and T

---

### Modelica.Media.Water.IF97\_Utilsities.BaselIF97.TwoPhase.**waterLiq\_p**

properties on the liquid phase boundary of region 4



#### Inputs

Name	Default	Description
p		pressure [Pa]

#### Outputs

Name	Description
liq	liquid thermodynamic property collection

---

### Modelica.Media.Water.IF97\_Utilsities.BaselIF97.TwoPhase.**waterVap\_p**

properties on the vapour phase boundary of region 4



#### Inputs

Name	Default	Description

---

## 1674 Modelica.Media.Water.IF97\_Utils.BaselF97.TwoPhase.waterVap\_p

---

p	pressure [Pa]
---	---------------

### Outputs

Name	Description
vap	vapour thermodynamic property collection

---

## Modelica.Media.Water.IF97\_Utils.BaselF97.TwoPhase.waterSat\_ph

Water saturation properties in the 2-phase region (4) as f(p,h)



### Inputs

Name	Default	Description
p		pressure [Pa]
h		specific enthalpy [J/kg]

### Outputs

Name	Description
pro	thermodynamic property collection

---

## Modelica.Media.Water.IF97\_Utils.BaselF97.TwoPhase.waterR4\_ph

Water/Steam properties in region 4 of IAPWS/IF97 (two-phase)



### Inputs

Name	Default	Description
p		pressure [Pa]
h		specific enthalpy [J/kg]

### Outputs

Name	Description
pro	thermodynamic property collection

---

## Modelica.Media.Water.IF97\_Utils.BaselF97.TwoPhase.waterR4\_dT

Water properties in region 4 as function of d and T



### Inputs

Name	Default	Description
d		Density [kg/m3]
T		temperature [K]

### Outputs

Name	Description
pro	thermodynamic property collection

**Modelica.Media.Water.IF97\_Utilsities.BaselF97.extraDerivs\_ph**

function to calculate some extra thermophysical properties in regions 1, 2, 3 and 5 as  $f(p,h)$

**Inputs**

Name	Default	Description
p		pressure [Pa]
h		specific enthalpy [J/kg]
phase	0	phase: 2 for two-phase, 1 for one phase, 0 if unknown

**Outputs**

Name	Description
dpro	thermodynamic property collection

**Modelica.Media.Water.IF97\_Utilsities.BaselF97.extraDerivs\_pT**

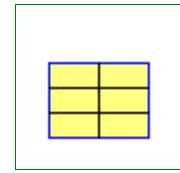
function to calculate some extra thermophysical properties in regions 1, 2, 3 and 5 as  $f(p,T)$

**Inputs**

Name	Default	Description
p		pressure [Pa]
T		temperature [K]

**Outputs**

Name	Description
dpro	thermodynamic property collection

**Modelica.Media.Water.IF97\_Utilsities.iter****Modelica definition**

```
replaceable record iter = BaselF97.IterationData;
```

**Modelica.Media.Water.IF97\_Utilsities.waterBaseProp\_ph**

intermediate property record for water

**Inputs**

Name	Default	Description
p		pressure [Pa]
h		specific enthalpy [J/kg]

## 1676 Modelica.Media.Water.IF97\_Utilsities.waterBaseProp\_ph

---

phase	0	phase: 2 for two-phase, 1 for one phase, 0 if unknown
region	0	if 0, do region computation, otherwise assume the region is this input

### Outputs

Name	Description
aux	auxiliary record

---

## Modelica.Media.Water.IF97\_Utilsities.waterBaseProp\_ps

intermediate property record for water



### Inputs

Name	Default	Description
p		pressure [Pa]
s		specific entropy [J/(kg.K)]
phase	0	phase: 2 for two-phase, 1 for one phase, 0 if unknown
region	0	if 0, do region computation, otherwise assume the region is this input

### Outputs

Name	Description
aux	auxiliary record

---

## Modelica.Media.Water.IF97\_Utilsities.rho\_props\_ps

density as function of pressure and specific entropy



### Inputs

Name	Default	Description
p		pressure [Pa]
s		specific entropy [J/(kg.K)]
properties		auxiliary record

### Outputs

Name	Description
rho	density [kg/m3]

---

## Modelica.Media.Water.IF97\_Utilsities.rho\_ps

density as function of pressure and specific entropy



### Inputs

Name	Default	Description
p		pressure [Pa]
s		specific entropy [J/(kg.K)]

---

phase	0	2 for two-phase, 1 for one-phase, 0 if not known
region	0	if 0, region is unknown, otherwise known and this input

## Outputs

Name	Description
rho	density [kg/m3]

---

## Modelica.Media.Water.IF97\_Utilsities.T\_props\_ps

temperature as function of pressure and specific entropy



## Inputs

Name	Default	Description
p		pressure [Pa]
s		specific entropy [J/(kg.K)]
properties		auxiliary record

## Outputs

Name	Description
T	temperature [K]

---

## Modelica.Media.Water.IF97\_Utilsities.T\_ps

temperature as function of pressure and specific entropy



## Inputs

Name	Default	Description
p		pressure [Pa]
s		specific entropy [J/(kg.K)]
phase	0	2 for two-phase, 1 for one-phase, 0 if not known
region	0	if 0, region is unknown, otherwise known and this input

## Outputs

Name	Description
T	Temperature [K]

---

## Modelica.Media.Water.IF97\_Utilsities.h\_props\_ps

specific enthalpy as function or pressure and temperature



## Inputs

Name	Default	Description
p		pressure [Pa]
s		specific entropy [J/(kg.K)]

---

**1678 Modelica.Media.Water.IF97\_Utils.h\_props\_ps**

---

aux		auxiliary record
-----	--	------------------

**Outputs**

Name	Description
h	specific enthalpy [J/kg]

---

**Modelica.Media.Water.IF97\_Utils.h\_ps****specific enthalpy as function of pressure and temperature****Inputs**

Name	Default	Description
p		pressure [Pa]
s		specific entropy [J/(kg.K)]
phase	0	2 for two-phase, 1 for one-phase, 0 if not known
region	0	if 0, region is unknown, otherwise known and this input

**Outputs**

Name	Description
h	specific enthalpy [J/kg]

---

**Modelica.Media.Water.IF97\_Utils.phase\_ps****phase as a function of pressure and specific entropy****Inputs**

Name	Default	Description
p		pressure [Pa]
s		specific entropy [J/(kg.K)]

**Outputs**

Name	Description
phase	true if in liquid or gas or supercritical region

---

**Modelica.Media.Water.IF97\_Utils.phase\_ph****phase as a function of pressure and specific enthalpy****Inputs**

Name	Default	Description
p		pressure [Pa]
h		specific enthalpy [J/kg]

## Outputs

Name	Description
phase	true if in liquid or gas or supercritical region

---

## Modelica.Media.Water.IF97\_Utilsities.phase\_dT

phase as a function of pressure and temperature



## Inputs

Name	Default	Description
rho		density [kg/m3]
T		temperature [K]

## Outputs

Name	Description
phase	true if in liquid or gas or supercritical region

---

## Modelica.Media.Water.IF97\_Utilsities.rho\_props\_ph

density as function of pressure and specific enthalpy



## Inputs

Name	Default	Description
p		pressure [Pa]
h		specific enthalpy [J/kg]
properties		auxiliary record

## Outputs

Name	Description
rho	density [kg/m3]

---

## Modelica.Media.Water.IF97\_Utilsities.rho\_ph

density as function of pressure and specific enthalpy



## Inputs

Name	Default	Description
p		pressure [Pa]
h		specific enthalpy [J/kg]
phase	0	2 for two-phase, 1 for one-phase, 0 if not known
region	0	if 0, region is unknown, otherwise known and this input

---

## 1680 Modelica.Media.Water.IF97\_Utils.rho\_ph

---

### Outputs

Name	Description
rho	density [kg/m3]

---



### Modelica.Media.Water.IF97\_Utils.rho\_ph\_der

derivative function of rho\_ph

### Inputs

Name	Default	Description
p		pressure [Pa]
h		specific enthalpy [J/kg]
aux		auxiliary record
p_der		derivative of pressure
h_der		derivative of specific enthalpy

### Outputs

Name	Description
rho_der	derivative of density

---



### Modelica.Media.Water.IF97\_Utils.T\_props\_ph

temperature as function of pressure and specific enthalpy

### Inputs

Name	Default	Description
p		pressure [Pa]
h		specific enthalpy [J/kg]
properties		auxiliary record

### Outputs

Name	Description
T	temperature [K]

---



### Modelica.Media.Water.IF97\_Utils.T\_ph

temperature as function of pressure and specific enthalpy

### Inputs

Name	Default	Description
p		pressure [Pa]
h		specific enthalpy [J/kg]
phase	0	2 for two-phase, 1 for one-phase, 0 if not known
region	0	if 0, region is unknown, otherwise known and this input

## Outputs

Name	Description
T	Temperature [K]

## Modelica.Media.Water.IF97\_Utils.T\_ph\_der

derivative function of T\_ph



## Inputs

Name	Default	Description
p		pressure [Pa]
h		specific enthalpy [J/kg]
aux		auxiliary record
p_der		derivative of pressure
h_der		derivative of specific enthalpy

## Outputs

Name	Description
T_der	derivative of temperature

## Modelica.Media.Water.IF97\_Utils.s\_props\_ph

specific entropy as function of pressure and specific enthalpy



## Inputs

Name	Default	Description
p		pressure [Pa]
h		specific enthalpy [J/kg]
properties		auxiliary record

## Outputs

Name	Description
s	specific entropy [J/(kg.K)]

## Modelica.Media.Water.IF97\_Utils.s\_ph

specific entropy as function of pressure and specific enthalpy



## Inputs

Name	Default	Description
p		pressure [Pa]
h		specific enthalpy [J/kg]
phase	0	2 for two-phase, 1 for one-phase, 0 if not known
region	0	if 0, region is unknown, otherwise known and this input

---

**1682 Modelica.Media.Water.IF97\_Utils.s\_ph****Outputs**

Name	Description
s	specific entropy [J/(kg.K)]

---

**Modelica.Media.Water.IF97\_Utils.s\_ph\_der**

specific entropy as function of pressure and specific enthalpy

**Inputs**

Name	Default	Description
p		pressure [Pa]
h		specific enthalpy [J/kg]
aux		auxiliary record
p_der		derivative of pressure
h_der		derivative of specific enthalpy

**Outputs**

Name	Description
s_der	derivative of entropy

---

**Modelica.Media.Water.IF97\_Utils.cv\_props\_ph**

specific heat capacity at constant volume as function of pressure and specific enthalpy

**Inputs**

Name	Default	Description
p		pressure [Pa]
h		specific enthalpy [J/kg]
aux		auxiliary record

**Outputs**

Name	Description
cv	specific heat capacity [J/(kg.K)]

---

**Modelica.Media.Water.IF97\_Utils.cv\_ph**

specific heat capacity at constant volume as function of pressure and specific enthalpy

**Inputs**

Name	Default	Description
p		pressure [Pa]
h		specific enthalpy [J/kg]
phase	0	2 for two-phase, 1 for one-phase, 0 if not known
region	0	if 0, region is unknown, otherwise known and this input

## Outputs

Name	Description
cv	specific heat capacity [J/(kg.K)]

---

## Modelica.Media.Water.IF97\_Utilsities.regionAssertReal

assert function for inlining



## Inputs

Name	Default	Description
check		condition to check

## Outputs

Name	Description
dummy	dummy output

---

## Modelica.Media.Water.IF97\_Utilsities.cp\_props\_ph

specific heat capacity at constant pressure as function of pressure and specific enthalpy



## Inputs

Name	Default	Description
p		pressure [Pa]
h		specific enthalpy [J/kg]
aux		auxiliary record

## Outputs

Name	Description
cp	specific heat capacity [J/(kg.K)]

---

## Modelica.Media.Water.IF97\_Utilsities.cp\_ph

specific heat capacity at constant pressure as function of pressure and specific enthalpy



## Inputs

Name	Default	Description
p		pressure [Pa]
h		specific enthalpy [J/kg]
phase	0	2 for two-phase, 1 for one-phase, 0 if not known
region	0	if 0, region is unknown, otherwise known and this input

## Outputs

Name	Description
cp	specific heat capacity [J/(kg.K)]

---

## Modelica.Media.Water.IF97\_Utilsities.beta\_props\_ph

isobaric expansion coefficient as function of pressure and specific enthalpy



## Inputs

Name	Default	Description
p		pressure [Pa]
h		specific enthalpy [J/kg]
aux		auxiliary record

## Outputs

Name	Description
beta	isobaric expansion coefficient [1/K]

---

## Modelica.Media.Water.IF97\_Utilsities.beta\_ph

isobaric expansion coefficient as function of pressure and specific enthalpy



## Inputs

Name	Default	Description
p		pressure [Pa]
h		specific enthalpy [J/kg]
phase	0	2 for two-phase, 1 for one-phase, 0 if not known
region	0	if 0, region is unknown, otherwise known and this input

## Outputs

Name	Description
beta	isobaric expansion coefficient [1/K]

---

## Modelica.Media.Water.IF97\_Utilsities.kappa\_props\_ph

isothermal compressibility factor as function of pressure and specific enthalpy



## Inputs

Name	Default	Description
p		pressure [Pa]
h		specific enthalpy [J/kg]
aux		auxiliary record

## Outputs

Name	Description
kappa	isothermal compressibility factor [1/Pa]

---

## Modelica.Media.Water.IF97\_Utils.kappa\_ph

isothermal compressibility factor as function of pressure and specific enthalpy



## Inputs

Name	Default	Description
p		pressure [Pa]
h		specific enthalpy [J/kg]
phase	0	2 for two-phase, 1 for one-phase, 0 if not known
region	0	if 0, region is unknown, otherwise known and this input

## Outputs

Name	Description
kappa	isothermal compressibility factor [1/Pa]

---

## Modelica.Media.Water.IF97\_Utils.velocityOfSound\_props\_ph

speed of sound as function of pressure and specific enthalpy



## Inputs

Name	Default	Description
p		pressure [Pa]
h		specific enthalpy [J/kg]
aux		auxiliary record

## Outputs

Name	Description
v_sound	speed of sound [m/s]

---

## Modelica.Media.Water.IF97\_Utils.velocityOfSound\_ph

## Inputs

Name	Default	Description
p		pressure [Pa]
h		specific enthalpy [J/kg]
phase	0	2 for two-phase, 1 for one-phase, 0 if not known
region	0	if 0, region is unknown, otherwise known and this input



---

**1686 Modelica.Media.Water.IF97\_Utils.iesentropicExponent\_props\_ph****Outputs**

Name	Description
v_sound	speed of sound [m/s]

---

**Modelica.Media.Water.IF97\_Utils.iesentropicExponent\_props\_ph**

isentropic exponent as function of pressure and specific enthalpy

**Inputs**

Name	Default	Description
p		pressure [Pa]
h		specific enthalpy [J/kg]
aux		auxiliary record

**Outputs**

Name	Description
gamma	isentropic exponent

---

**Modelica.Media.Water.IF97\_Utils.iesentropicExponent\_ph**

isentropic exponent as function of pressure and specific enthalpy

**Inputs**

Name	Default	Description
p		pressure [Pa]
h		specific enthalpy [J/kg]
phase	0	2 for two-phase, 1 for one-phase, 0 if not known
region	0	if 0, region is unknown, otherwise known and this input

**Outputs**

Name	Description
gamma	isentropic exponent

---

**Modelica.Media.Water.IF97\_Utils.ddph\_props**

density derivative by pressure

**Inputs**

Name	Default	Description
p		pressure [Pa]
h		specific enthalpy [J/kg]
aux		auxiliary record

## Outputs

Name	Description
ddph	density derivative by pressure [s2/m2]

---

## Modelica.Media.Water.IF97\_Utils ddph

density derivative by pressure



## Inputs

Name	Default	Description
p		pressure [Pa]
h		specific enthalpy [J/kg]
phase	0	2 for two-phase, 1 for one-phase, 0 if not known
region	0	if 0, region is unknown, otherwise known and this input

## Outputs

Name	Description
ddph	density derivative by pressure [s2/m2]

---

## Modelica.Media.Water.IF97\_Utils ddhp\_props

density derivative by specific enthalpy



## Inputs

Name	Default	Description
p		pressure [Pa]
h		specific enthalpy [J/kg]
aux		auxiliary record

## Outputs

Name	Description
ddhp	density derivative by specific enthalpy [kg.s2/m5]

---

## Modelica.Media.Water.IF97\_Utils ddhp

density derivative by specific enthalpy



## Inputs

Name	Default	Description
p		pressure [Pa]
h		specific enthalpy [J/kg]
phase	0	2 for two-phase, 1 for one-phase, 0 if not known
region	0	if 0, region is unknown, otherwise known and this input

## Outputs

Name	Description
ddhp	density derivative by specific enthalpy [kg.s <sup>2</sup> /m <sup>5</sup> ]

---

## Modelica.Media.Water.IF97\_Utilsities.waterBaseProp\_pT

intermediate property record for water (p and T prefered states)



## Inputs

Name	Default	Description
p		pressure [Pa]
T		temperature [K]
region	0	if 0, do region computation, otherwise assume the region is this input

---

## Outputs

Name	Description
aux	auxiliary record

---

## Modelica.Media.Water.IF97\_Utilsities.rho\_props\_pT

density as function or pressure and temperature



## Inputs

Name	Default	Description
p		pressure [Pa]
T		temperature [K]
aux		auxiliary record

---

## Outputs

Name	Description
rho	density [kg/m <sup>3</sup> ]

---

## Modelica.Media.Water.IF97\_Utilsities.rho\_pT

density as function or pressure and temperature



## Inputs

Name	Default	Description
p		pressure [Pa]
T		temperature [K]
region	0	if 0, region is unknown, otherwise known and this input

---

## Outputs

Name	Description
rho	density [kg/m3]

## Modelica.Media.Water.IF97\_Utilsities.h\_props\_pT

specific enthalpy as function of pressure and temperature



## Inputs

Name	Default	Description
p		pressure [Pa]
T		temperature [K]
aux		auxiliary record

## Outputs

Name	Description
h	specific enthalpy [J/kg]

## Modelica.Media.Water.IF97\_Utilsities.h\_pT

specific enthalpy as function of pressure and temperature



## Inputs

Name	Default	Description
p		pressure [Pa]
T		Temperature [K]
region	0	if 0, region is unknown, otherwise known and this input

## Outputs

Name	Description
h	specific enthalpy [J/kg]

## Modelica.Media.Water.IF97\_Utilsities.h\_pT\_der

derivative function of h\_pT



## Inputs

Name	Default	Description
p		pressure [Pa]
T		temperature [K]
aux		auxiliary record
p_der		derivative of pressure
T_der		derivative of temperature

---

## 1690 Modelica.Media.Water.IF97\_Utils.h\_pT\_der

---

### Outputs

Name	Description
h_der	derivative of specific enthalpy

---

## Modelica.Media.Water.IF97\_Utils.rho\_pT\_der

derivative function of rho\_pT



### Inputs

Name	Default	Description
p		pressure [Pa]
T		temperature [K]
aux		auxiliary record
p_der		derivative of pressure
T_der		derivative of temperature

### Outputs

Name	Description
rho_der	derivative of density

---

## Modelica.Media.Water.IF97\_Utils.s\_props\_pT

specific entropy as function of pressure and temperature



### Inputs

Name	Default	Description
p		pressure [Pa]
T		temperature [K]
aux		auxiliary record

### Outputs

Name	Description
s	specific entropy [J/(kg.K)]

---

## Modelica.Media.Water.IF97\_Utils.s\_pT

temperature as function of pressure and temperature



### Inputs

Name	Default	Description
p		pressure [Pa]
T		temperature [K]
region	0	if 0, region is unknown, otherwise known and this input

## Outputs

Name	Description
s	specific entropy [J/(kg.K)]

---

## Modelica.Media.Water.IF97\_Utilsities.CV\_props\_pT

specific heat capacity at constant volume as function of pressure and temperature



## Inputs

Name	Default	Description
p		pressure [Pa]
T		temperature [K]
aux		auxiliary record

## Outputs

Name	Description
cv	specific heat capacity [J/(kg.K)]

---

## Modelica.Media.Water.IF97\_Utilsities.CV\_pT

specific heat capacity at constant volume as function of pressure and temperature



## Inputs

Name	Default	Description
p		pressure [Pa]
T		temperature [K]
region	0	if 0, region is unknown, otherwise known and this input

## Outputs

Name	Description
cv	specific heat capacity [J/(kg.K)]

---

## Modelica.Media.Water.IF97\_Utilsities.cp\_props\_pT

specific heat capacity at constant pressure as function of pressure and temperature



## Inputs

Name	Default	Description
p		pressure [Pa]
T		temperature [K]
aux		auxiliary record

---

## 1692 Modelica.Media.Water.IF97\_Utilsities.cp\_props\_pT

---

### Outputs

Name	Description
cp	specific heat capacity [J/(kg.K)]

---

### Modelica.Media.Water.IF97\_Utilsities.cp\_pT

specific heat capacity at constant pressure as function of pressure and temperature



### Inputs

Name	Default	Description
p		pressure [Pa]
T		temperature [K]
region	0	if 0, region is unknown, otherwise known and this input

---

### Outputs

Name	Description
cp	specific heat capacity [J/(kg.K)]

---

### Modelica.Media.Water.IF97\_Utilsities.beta\_props\_pT

isobaric expansion coefficient as function of pressure and temperature



### Inputs

Name	Default	Description
p		pressure [Pa]
T		temperature [K]
aux		auxiliary record

---

### Outputs

Name	Description
beta	isobaric expansion coefficient [1/K]

---

### Modelica.Media.Water.IF97\_Utilsities.beta\_pT

isobaric expansion coefficient as function of pressure and temperature



### Inputs

Name	Default	Description
p		pressure [Pa]
T		temperature [K]
region	0	if 0, region is unknown, otherwise known and this input

---

## Outputs

Name	Description
beta	isobaric expansion coefficient [1/K]

---

## Modelica.Media.Water.IF97\_Utilsities.kappa\_props\_pT

isothermal compressibility factor as function of pressure and temperature



## Inputs

Name	Default	Description
p		pressure [Pa]
T		temperature [K]
aux		auxiliary record

## Outputs

Name	Description
kappa	isothermal compressibility factor [1/Pa]

---

## Modelica.Media.Water.IF97\_Utilsities.kappa\_pT

isothermal compressibility factor as function of pressure and temperature



## Inputs

Name	Default	Description
p		pressure [Pa]
T		temperature [K]
region	0	if 0, region is unknown, otherwise known and this input

## Outputs

Name	Description
kappa	isothermal compressibility factor [1/Pa]

---

## Modelica.Media.Water.IF97\_Utilsities.velocityOfSound\_props\_pT

speed of sound as function of pressure and temperature



## Inputs

Name	Default	Description
p		pressure [Pa]
T		temperature [K]
aux		auxiliary record

---

**1694 Modelica.Media.Water.IF97\_Utilsities.velocityOfSound\_props\_pT**

---

**Outputs**

Name	Description
v_sound	speed of sound [m/s]

---

**Modelica.Media.Water.IF97\_Utilsities.velocityOfSound\_pT**

speed of sound as function of pressure and temperature

**Inputs**

Name	Default	Description
p		pressure [Pa]
T		temperature [K]
region	0	if 0, region is unknown, otherwise known and this input

---

**Outputs**

Name	Description
v_sound	speed of sound [m/s]

---

**Modelica.Media.Water.IF97\_Utilsities.isentropicExponent\_props\_pT**

isentropic exponent as function of pressure and temperature

**Inputs**

Name	Default	Description
p		pressure [Pa]
T		temperature [K]
aux		auxiliary record

---

**Outputs**

Name	Description
gamma	isentropic exponent

---

**Modelica.Media.Water.IF97\_Utilsities.isentropicExponent\_pT**

isentropic exponent as function of pressure and temperature

**Inputs**

Name	Default	Description
p		pressure [Pa]
T		temperature [K]
region	0	if 0, region is unknown, otherwise known and this input

---

## Outputs

Name	Description
gamma	isentropic exponent

---

## Modelica.Media.Water.IF97\_Utilsies.waterBaseProp\_dT

intermediate property record for water (d and T prefered states)



## Inputs

Name	Default	Description
rho		density [kg/m3]
T		temperature [K]
phase	0	phase: 2 for two-phase, 1 for one phase, 0 if unknown
region	0	if 0, do region computation, otherwise assume the region is this input

## Outputs

Name	Description
aux	auxiliary record

---

## Modelica.Media.Water.IF97\_Utilsies.h\_props\_dT

specific enthalpy as function of density and temperature



## Inputs

Name	Default	Description
d		density [kg/m3]
T		Temperature [K]
aux		auxiliary record

## Outputs

Name	Description
h	specific enthalpy [J/kg]

---

## Modelica.Media.Water.IF97\_Utilsies.h\_dT

specific enthalpy as function of density and temperature



## Inputs

Name	Default	Description
d		density [kg/m3]
T		Temperature [K]
phase	0	2 for two-phase, 1 for one-phase, 0 if not known
region	0	if 0, region is unknown, otherwise known and this input

## Outputs

Name	Description
h	specific enthalpy [J/kg]

---

## Modelica.Media.Water.IF97\_Utils.h\_dT\_der

derivative function of h\_dT



## Inputs

Name	Default	Description
d		density [kg/m3]
T		temperature [K]
aux		auxiliary record
d_der		derivative of density
T_der		derivative of temperature

## Outputs

Name	Description
h_der	derivative of specific enthalpy

---

## Modelica.Media.Water.IF97\_Utils.p\_props\_dT

pressure as function of density and temperature



## Inputs

Name	Default	Description
d		density [kg/m3]
T		Temperature [K]
aux		auxiliary record

## Outputs

Name	Description
p	pressure [Pa]

---

## Modelica.Media.Water.IF97\_Utils.p\_dT

pressure as function of density and temperature



## Inputs

Name	Default	Description
d		density [kg/m3]
T		Temperature [K]
phase	0	2 for two-phase, 1 for one-phase, 0 if not known
region	0	if 0, region is unknown, otherwise known and this input

## Outputs

Name	Description
p	pressure [Pa]

---

## Modelica.Media.Water.IF97\_Utils.p\_dT\_der

derivative function of p\_dT



## Inputs

Name	Default	Description
d		density [kg/m3]
T		temperature [K]
aux		auxiliary record
d_der		derivative of density
T_der		derivative of temperature

## Outputs

Name	Description
p_der	derivative of pressure

---

## Modelica.Media.Water.IF97\_Utils.s\_props\_dT

specific entropy as function of density and temperature



## Inputs

Name	Default	Description
d		density [kg/m3]
T		Temperature [K]
aux		auxiliary record

## Outputs

Name	Description
s	specific entropy [J/(kg.K)]

---

## Modelica.Media.Water.IF97\_Utils.s\_dT

temperature as function of density and entropy



## Inputs

Name	Default	Description
d		density [kg/m3]
T		Temperature [K]
phase	0	2 for two-phase, 1 for one-phase, 0 if not known
region	0	if 0, region is unknown, otherwise known and this input

---

## 1698 Modelica.Media.Water.IF97\_Utils.s\_dT

---

### Outputs

Name	Description
s	specific entropy [J/(kg.K)]

---

### Modelica.Media.Water.IF97\_Utils.CV\_props\_dT

specific heat capacity at constant volume as function of density and temperature



### Inputs

Name	Default	Description
d		density [kg/m3]
T		temperature [K]
aux		auxiliary record

### Outputs

Name	Description
cv	specific heat capacity [J/(kg.K)]

---

### Modelica.Media.Water.IF97\_Utils.CV\_dT

specific heat capacity at constant volume as function of density and temperature



### Inputs

Name	Default	Description
d		density [kg/m3]
T		temperature [K]
phase	0	2 for two-phase, 1 for one-phase, 0 if not known
region	0	if 0, region is unknown, otherwise known and this input

### Outputs

Name	Description
cv	specific heat capacity [J/(kg.K)]

---

### Modelica.Media.Water.IF97\_Utils.cp\_props\_dT

specific heat capacity at constant pressure as function of density and temperature



### Inputs

Name	Default	Description
d		density [kg/m3]
T		temperature [K]
aux		auxiliary record

## Outputs

Name	Description
cp	specific heat capacity [J/(kg.K)]

---

## Modelica.Media.Water.IF97\_Utilsies.cp\_dT

specific heat capacity at constant pressure as function of density and temperature



## Inputs

Name	Default	Description
d		density [kg/m3]
T		temperature [K]
phase	0	2 for two-phase, 1 for one-phase, 0 if not known
region	0	if 0, region is unknown, otherwise known and this input

## Outputs

Name	Description
cp	specific heat capacity [J/(kg.K)]

---

## Modelica.Media.Water.IF97\_Utilsies.beta\_props\_dT

isobaric expansion coefficient as function of density and temperature



## Inputs

Name	Default	Description
d		density [kg/m3]
T		temperature [K]
aux		auxiliary record

## Outputs

Name	Description
beta	isobaric expansion coefficient [1/K]

---

## Modelica.Media.Water.IF97\_Utilsies.beta\_dT

isobaric expansion coefficient as function of density and temperature



## Inputs

Name	Default	Description
d		density [kg/m3]
T		temperature [K]
phase	0	2 for two-phase, 1 for one-phase, 0 if not known
region	0	if 0, region is unknown, otherwise known and this input

---

## 1700 Modelica.Media.Water.IF97\_Utilsities.beta\_dT

---

### Outputs

Name	Description
beta	isobaric expansion coefficient [1/K]

---

## Modelica.Media.Water.IF97\_Utilsities.kappa\_props\_dT

isothermal compressibility factor as function of density and temperature



### Inputs

Name	Default	Description
d		density [kg/m3]
T		temperature [K]
aux		auxiliary record

### Outputs

Name	Description
kappa	isothermal compressibility factor [1/Pa]

---

## Modelica.Media.Water.IF97\_Utilsities.kappa\_dT

isothermal compressibility factor as function of density and temperature



### Inputs

Name	Default	Description
d		density [kg/m3]
T		temperature [K]
phase	0	2 for two-phase, 1 for one-phase, 0 if not known
region	0	if 0, region is unknown, otherwise known and this input

### Outputs

Name	Description
kappa	isothermal compressibility factor [1/Pa]

---

## Modelica.Media.Water.IF97\_Utilsities.velocityOfSound\_props\_dT

speed of sound as function of density and temperature



### Inputs

Name	Default	Description
d		density [kg/m3]
T		temperature [K]
aux		auxiliary record

## Outputs

Name	Description
v_sound	speed of sound [m/s]

---

## Modelica.Media.Water.IF97\_Utilsities.velocityOfSound\_dT

speed of sound as function of density and temperature



## Inputs

Name	Default	Description
d		density [kg/m3]
T		temperature [K]
phase	0	2 for two-phase, 1 for one-phase, 0 if not known
region	0	if 0, region is unknown, otherwise known and this input

## Outputs

Name	Description
v_sound	speed of sound [m/s]

---

## Modelica.Media.Water.IF97\_Utilsities.isentropicExponent\_props\_dT

isentropic exponent as function of density and temperature



## Inputs

Name	Default	Description
d		density [kg/m3]
T		temperature [K]
aux		auxiliary record

## Outputs

Name	Description
gamma	isentropic exponent

---

## Modelica.Media.Water.IF97\_Utilsities.isentropicExponent\_dT

isentropic exponent as function of density and temperature



## Inputs

Name	Default	Description
d		density [kg/m3]
T		temperature [K]
phase	0	2 for two-phase, 1 for one-phase, 0 if not known
region	0	if 0, region is unknown, otherwise known and this input

---

**1702 Modelica.Media.Water.IF97\_Utilsies.isentropicExponent\_dT**

---

**Outputs**

Name	Description
gamma	isentropic exponent

---

**Modelica.Media.Water.IF97\_Utilsies.hl\_p**

compute the saturated liquid specific h(p)

**Inputs**

Name	Default	Description
p		pressure [Pa]

---

**Outputs**

Name	Description
h	specific enthalpy [J/kg]

---

**Modelica.Media.Water.IF97\_Utilsies.hv\_p**

compute the saturated vapour specific h(p)

**Inputs**

Name	Default	Description
p		pressure [Pa]

---

**Outputs**

Name	Description
h	specific enthalpy [J/kg]

---

**Modelica.Media.Water.IF97\_Utilsies.sl\_p**

compute the saturated liquid specific s(p)

**Inputs**

Name	Default	Description
p		pressure [Pa]

---

**Outputs**

Name	Description
s	specific entropy [J/(kg.K)]

---

**Modelica.Media.Water.IF97\_Utilsies.sv\_p**

compute the saturated vapour specific s(p)



**Inputs**

Name	Default	Description
p		pressure [Pa]

**Outputs**

Name	Description
s	specific entropy [J/(kg.K)]

**Modelica.Media.Water.IF97\_Utilsies.rhol\_T**

compute the saturated liquid d(T)

**Inputs**

Name	Default	Description
T		temperature [K]

**Outputs**

Name	Description
d	density of water at the boiling point [kg/m3]

**Modelica.Media.Water.IF97\_Utilsies.rhov\_T**

compute the saturated vapour d(T)

**Inputs**

Name	Default	Description
T		temperature [K]

**Outputs**

Name	Description
d	density of steam at the condensation point [kg/m3]

**Modelica.Media.Water.IF97\_Utilsies.rhol\_p**

compute the saturated liquid d(p)

**Inputs**

Name	Default	Description
p		saturation pressure [Pa]

**Outputs**

Name	Description
rho	density of steam at the condensation point [kg/m3]

**Modelica.Media.Water.IF97\_Utils.rhol\_p**

compute the saturated vapour d(p)

**Inputs**

Name	Default	Description
p		saturation pressure [Pa]

**Outputs**

Name	Description
rho	density of steam at the condensation point [kg/m3]

---

**Modelica.Media.Water.IF97\_Utils.dynamicViscosity**

compute eta(d,T) in the one-phase region

**Inputs**

Name	Default	Description
d		density [kg/m3]
T		temperature (K) [K]
p		pressure (only needed for region of validity) [Pa]
phase	0	2 for two-phase, 1 for one-phase, 0 if not known

**Outputs**

Name	Description
eta	dynamic viscosity [Pa.s]

---

**Modelica.Media.Water.IF97\_Utils.thermalConductivity**

compute lambda(d,T,p) in the one-phase region

**Inputs**

Name	Default	Description
d		density [kg/m3]
T		temperature (K) [K]
p		pressure [Pa]
phase	0	2 for two-phase, 1 for one-phase, 0 if not known
industrialMethod	true	if true, the industrial method is used, otherwise the scientific one

**Outputs**

Name	Description
lambda	thermal conductivity [W/(m.K)]

---

**Modelica.Media.Water.IF97\_Utilsities.surfaceTension**

compute sigma(T) at saturation T

**Inputs**

Name	Default	Description
T		temperature (K) [K]

**Outputs**

Name	Description
sigma	surface tension in SI units [N/m]

**Modelica.Media.Water.IF97\_Utilsities.isentropicEnthalpy**

isentropic specific enthalpy from p,s (preferably use dynamicIsentropicEnthalpy in dynamic simulation!)

**Inputs**

Name	Default	Description
p		pressure [Pa]
s		specific entropy [J/(kg.K)]
phase	0	2 for two-phase, 1 for one-phase, 0 if not known
region	0	if 0, region is unknown, otherwise known and this input

**Outputs**

Name	Description
h	specific enthalpy [J/kg]

**Modelica.Media.Water.IF97\_Utilsities.isentropicEnthalpy\_props****Inputs**

Name	Default	Description
p		pressure [Pa]
s		specific entropy [J/(kg.K)]
aux		auxiliary record

**Outputs**

Name	Description
h	isentropic enthalpy [J/kg]

**Modelica.Media.Water.IF97\_Utilsities.isentropicEnthalpy\_der**

derivative of isentropic specific enthalpy from p,s

## 1706 Modelica.Media.Water.IF97\_Utils.isentropicEnthalpy\_der

---

### Inputs

Name	Default	Description
p		pressure [Pa]
s		specific entropy [J/(kg.K)]
aux		auxiliary record
p_der		pressure derivative
s_der		entropy derivative

### Outputs

Name	Description
h_der	specific enthalpy derivative

---

## Modelica.Media.Water.IF97\_Utils.dynamicIsentropicEnthalpy

isentropic specific enthalpy from p,s and good guesses of d and T



### Inputs

Name	Default	Description
p		pressure [Pa]
s		specific entropy [J/(kg.K)]
dguess		good guess density, e.g. from adjacent volume [kg/m3]
Tguess		good guess temperature, e.g. from adjacent volume [K]
phase	0	2 for two-phase, 1 for one-phase, 0 if not known

### Outputs

Name	Description
h	specific enthalpy [J/kg]

---

## Modelica.Slunits

Library of type and unit definitions based on SI units according to ISO 31-1992

### Information

This package provides predefined types, such as *Mass*, *Angle*, *Time*, based on the international standard on units, e.g.,

```
type Angle = Real(final quantity = "Angle",
                     final unit      = "rad",
                     displayUnit    = "deg");
```

as well as conversion functions from non SI-units to SI-units and vice versa in subpackage *Conversions*.

For an introduction how units are used in the Modelica standard library with package *Slunits*, have a look at: [How to use Slunits](#).

Copyright © 1998-2007, Modelica Association and DLR.

*This Modelica package is free software; it can be redistributed and/or modified under the terms of the Modelica license, see the license conditions and the accompanying disclaimer here.*

## Package Content

Name	Description
 <a href="#">UsersGuide</a>	User's Guide of SIunits Library
 <a href="#">Conversions</a>	Conversion functions to/from non SI units and type definitions of non SI units
<a href="#">Angle</a>	
<a href="#">SolidAngle</a>	
<a href="#">Length</a>	
<a href="#">PathLength</a>	
<a href="#">Position</a>	
<a href="#">Distance</a>	
<a href="#">Breadth</a>	
<a href="#">Height</a>	
<a href="#">Thickness</a>	
<a href="#">Radius</a>	
<a href="#">Diameter</a>	
<a href="#">Area</a>	
<a href="#">Volume</a>	
<a href="#">Time</a>	
<a href="#">Duration</a>	
<a href="#">AngularVelocity</a>	
<a href="#">AngularAcceleration</a>	
<a href="#">Velocity</a>	
<a href="#">Acceleration</a>	
<a href="#">Period</a>	
<a href="#">Frequency</a>	
<a href="#">AngularFrequency</a>	
<a href="#">Wavelength</a>	
<a href="#">Wavelenght</a>	
<a href="#">WaveNumber</a>	
<a href="#">CircularWaveNumber</a>	
<a href="#">AmplitudeLevelDifference</a>	
<a href="#">PowerLevelDifference</a>	
<a href="#">DampingCoefficient</a>	
<a href="#">LogarithmicDecrement</a>	
<a href="#">AttenuationCoefficient</a>	
<a href="#">PhaseCoefficient</a>	
<a href="#">PropagationCoefficient</a>	
<a href="#">Damping</a>	
<a href="#">Mass</a>	
<a href="#">Density</a>	
<a href="#">RelativeDensity</a>	
<a href="#">SpecificVolume</a>	
<a href="#">LinearDensity</a>	
<a href="#">SurfaceDensity</a>	

Momentum	
Impulse	
AngularMomentum	
AngularImpulse	
MomentOfInertia	
Inertia	
Force	
Weight	
Torque	
MomentOfForce	
Pressure	
AbsolutePressure	
BulkModulus	
Stress	
NormalStress	
ShearStress	
Strain	
LinearStrain	
ShearStrain	
VolumeStrain	
PoissonNumber	
ModulusOfElasticity	
ShearModulus	
SecondMomentOfArea	
SecondPolarMomentOfArea	
SectionModulus	
CoefficientOfFriction	
DynamicViscosity	
KinematicViscosity	
SurfaceTension	
Work	
Energy	
EnergyDensity	
PotentialEnergy	
KineticEnergy	
Power	
EnergyFlowRate	
EnthalpyFlowRate	
Efficiency	
MassFlowRate	
VolumeFlowRate	
MomentumFlux	
AngularMomentumFlux	
ThermodynamicTemperature	
Temp_K	
Temperature	

TemperatureDifference	
CelsiusTemperature	
Temp_C	
LinearExpansionCoefficient	
CubicExpansionCoefficient	
RelativePressureCoefficient	
PressureCoefficient	
Compressibility	
IsothermalCompressibility	
IsentropicCompressibility	
Heat	
HeatFlowRate	
HeatFlux	
DensityOfHeatFlowRate	
ThermalConductivity	
CoefficientOfHeatTransfer	
SurfaceCoefficientOfHeatTransfer	
ThermalInsulance	
ThermalResistance	
ThermalConductance	
ThermalDiffusivity	
HeatCapacity	
SpecificHeatCapacity	
SpecificHeatCapacityAtConstantPressure	
SpecificHeatCapacityAtConstantVolume	
SpecificHeatCapacityAtSaturation	
RatioOfSpecificHeatCapacities	
IsentropicExponent	
Entropy	
SpecificEntropy	
InternalEnergy	
Enthalpy	
HelmholtzFreeEnergy	
GibbsFreeEnergy	
SpecificEnergy	
SpecificInternalEnergy	
SpecificEnthalpy	
SpecificHelmholtzFreeEnergy	
SpecificGibbsFreeEnergy	
MassieuFunction	
PlanckFunction	
DerDensityByEnthalpy	
DerDensityByPressure	
DerDensityByTemperature	
DerEnthalpyByPressure	
DerEnergyByDensity	

## 1710 Modelica.SIunits

---

DerEnergyByPressure	
ElectricCurrent	
Current	
ElectricCharge	
Charge	
VolumeDensityOfCharge	
SurfaceDensityOfCharge	
ElectricFieldStrength	
ElectricPotential	
Voltage	
PotentialDifference	
ElectromotiveForce	
ElectricFluxDensity	
ElectricFlux	
Capacitance	
Permittivity	
PermittivityOfVacuum	
RelativePermittivity	
ElectricSusceptibility	
ElectricPolarization	
Electrization	
ElectricDipoleMoment	
CurrentDensity	
LinearCurrentDensity	
MagneticFieldStrength	
MagneticPotential	
MagneticPotentialDifference	
MagnetomotiveForce	
CurrentLinkage	
MagneticFluxDensity	
MagneticFlux	
MagneticVectorPotential	
Inductance	
SelfInductance	
MutualInductance	
CouplingCoefficient	
LeakageCoefficient	
Permeability	
PermeabilityOfVacuum	
RelativePermeability	
MagneticSusceptibility	
ElectromagneticMoment	
MagneticDipoleMoment	
Magnetization	
MagneticPolarization	
ElectromagneticEnergyDensity	

PoyntingVector	
Resistance	
Resistivity	
Conductivity	
Reluctance	
Permeance	
PhaseDifference	
Impedance	
ModulusOfImpedance	
Reactance	
QualityFactor	
LossAngle	
Conductance	
Admittance	
ModulusOfAdmittance	
Susceptance	
InstantaneousPower	
ActivePower	
ApparentPower	
ReactivePower	
PowerFactor	
Transconductance	
InversePotential	
RadiantEnergy	
RadiantEnergyDensity	
SpectralRadiantEnergyDensity	
RadiantPower	
RadiantEnergyFluenceRate	
RadiantIntensity	
Radiance	
RadiantExtiance	
Irradiance	
Emissivity	
SpectralEmissivity	
DirectionalSpectralEmissivity	
LuminousIntensity	
LuminousFlux	
QuantityOfLight	
Luminance	
LuminousExitance	
Illuminance	
LightExposure	
LuminousEfficacy	
SpectralLuminousEfficacy	
LuminousEfficiency	
SpectralLuminousEfficiency	

CIESpectralTristimulusValues	
ChromaticityCoordinates	
SpectralAbsorptionFactor	
SpectralReflectionFactor	
SpectralTransmissionFactor	
SpectralRadianceFactor	
LinearAttenuationCoefficient	
LinearAbsorptionCoefficient	
MolarAbsorptionCoefficient	
RefractiveIndex	
StaticPressure	
SoundPressure	
SoundParticleDisplacement	
SoundParticleVelocity	
SoundParticleAcceleration	
VelocityOfSound	
SoundEnergyDensity	
SoundPower	
SoundIntensity	
AcousticImpedance	
SpecificAcousticImpedance	
MechanicalImpedance	
SoundPressureLevel	
SoundPowerLevel	
DissipationCoefficient	
ReflectionCoefficient	
TransmissionCoefficient	
AcousticAbsorptionCoefficient	
SoundReductionIndex	
EquivalentAbsorptionArea	
ReverberationTime	
LoundnessLevel	
Loundness	
RelativeAtomicMass	
RelativeMolecularMass	
NumberOfMolecules	
AmountOfSubstance	
MolarMass	
MolarVolume	
MolarInternalEnergy	
MolarHeatCapacity	
MolarEntropy	
NumberDensityOfMolecules	
MolecularConcentration	
MassConcentration	
MassFraction	

Concentration	
VolumeFraction	
MoleFraction	
ChemicalPotential	
AbsoluteActivity	
PartialPressure	
Fugacity	
StandardAbsoluteActivity	
ActivityCoefficient	
ActivityOfSolute	
ActivityCoefficientOfSolute	
StandardAbsoluteActivityOfSolute	
ActivityOfSolvent	
OsmoticCoefficientOfSolvent	
StandardAbsoluteActivityOfSolvent	
OsmoticPressure	
StoichiometricNumber	
Affinity	
MassOfMolecule	
ElectricDipoleMomentOfMolecule	
ElectricPolarizabilityOfAMolecule	
MicrocanonicalPartitionFunction	
CanonicalPartitionFunction	
GrandCanonicalPartitionFunction	
MolecularPartitionFunction	
StatisticalWeight	
MeanFreePath	
DiffusionCoefficient	
ThermalDiffusionRatio	
ThermalDiffusionFactor	
ThermalDiffusionCoefficient	
ElementaryCharge	
ChargeNumberOfIon	
FaradayConstant	
IonicStrength	
DegreeOfDissociation	
ElectrolyticConductivity	
MolarConductivity	
TransportNumberOfIonic	
ProtonNumber	
NeutronNumber	
NucleonNumber	
AtomicMassConstant	
MassOfElectron	
MassOfProton	
MassOfNeutron	

HartreeEnergy
MagneticMomentOfParticle
BohrMagneton
NuclearMagneton
GyromagneticCoefficient
GFactorOfAtom
GFactorOfNucleus
LarmorAngularFrequency
NuclearPrecessionAngularFrequency
CyclotronAngularFrequency
NuclearQuadrupoleMoment
NuclearRadius
ElectronRadius
ComptonWavelength
MassExcess
MassDefect
RelativeMassExcess
RelativeMassDefect
PackingFraction
BindingFraction
MeanLife
LevelWidth
Activity
SpecificActivity
DecayConstant
HalfLife
AlphaDisintegrationEnergy
MaximumBetaParticleEnergy
BetaDisintegrationEnergy
ReactionEnergy
ResonanceEnergy
CrossSection
TotalCrossSection
AngularCrossSection
SpectralCrossSection
SpectralAngularCrossSection
MacroscopicCrossSection
TotalMacroscopicCrossSection
ParticleFluence
ParticleFluenceRate
EnergyFluence
EnergyFluenceRate
CurrentDensityOfParticles
MassAttenuationCoefficient
MolarAttenuationCoefficient
AtomicAttenuationCoefficient

HalfThickness	
TotalLinearStoppingPower	
TotalAtomicStoppingPower	
TotalMassStoppingPower	
MeanLinearRange	
MeanMassRange	
LinearIonization	
TotalIonization	
Mobility	
IonNumberDensity	
RecombinationCoefficient	
NeutronNumberDensity	
NeutronSpeed	
NeutronFluenceRate	
TotalNeutronSourceDensity	
SlowingDownDensity	
ResonanceEscapeProbability	
Lethargy	
SlowingDownArea	
DiffusionArea	
MigrationArea	
SlowingDownLength	
DiffusionLength	
MigrationLength	
NeutronYieldPerFission	
NeutronYieldPerAbsorption	
FastFissionFactor	
ThermalUtilizationFactor	
NonLeakageProbability	
Reactivity	
ReactorTimeConstant	
EnergyImparted	
MeanEnergyImparted	
SpecificEnergyImparted	
AbsorbedDose	
DoseEquivalent	
AbsorbedDoseRate	
LinearEnergyTransfer	
Kerma	
KermaRate	
MassEnergyTransferCoefficient	
Exposure	
ExposureRate	
ReynoldsNumber	
EulerNumber	
FroudeNumber	

GrashofNumber
WeberNumber
MachNumber
KnudsenNumber
StrouhalNumber
FourierNumber
PecletNumber
RayleighNumber
NusseltNumber
BiotNumber
StantonNumber
FourierNumberOfMassTransfer
PecletNumberOfMassTransfer
GrashofNumberOfMassTransfer
NusseltNumberOfMassTransfer
StantonNumberOfMassTransfer
PrandtlNumber
SchmidtNumber
LewisNumber
MagneticReynoldsNumber
AlfvenNumber
HartmannNumber
CowlingNumber
BraggAngle
OrderOfReflexion
ShortRangeOrderParameter
LongRangeOrderParameter
DebyeWallerFactor
CircularWavenumber
FermiCircularWavenumber
DebyeCircularWavenumber
DebyeCircularFrequency
DebyeTemperature
SpectralConcentration
GrueneisenParameter
MadelungConstant
DensityOfStates
ResidualResistivity
LorenzCoefficient
HallCoefficient
ThermoelectromotiveForce
SeebeckCoefficient
PeltierCoefficient
ThomsonCoefficient
RichardsonConstant
FermiEnergy

GapEnergy
DonorIonizationEnergy
AcceptorIonizationEnergy
FermiTemperature
ElectronNumberDensity
HoleNumberDensity
IntrinsicNumberDensity
DonorNumberDensity
AcceptorNumberDensity
EffectiveMass
MobilityRatio
RelaxationTime
CarrierLifetime
ExchangeIntegral
CurieTemperature
NeelTemperature
LondonPenetrationDepth
CoherenceLength
LandauGinzburgParameter
FluxoidQuantum

## Types and constants

```

type Angle = Real (
  final quantity="Angle",
  final unit="rad",
  displayUnit="deg");

type SolidAngle = Real (final quantity="SolidAngle", final unit="sr");

type Length = Real (final quantity="Length", final unit="m");

type PathLength = Length;

type Position = Length;

type Distance = Length (min=0);

type Breadth = Length(min=0);

type Height = Length(min=0);

type Thickness = Length(min=0);

type Radius = Length(min=0);

type Diameter = Length(min=0);

type Area = Real (final quantity="Area", final unit="m2");

```

## 1718 Modelica.SIunits

---

```
type Volume = Real (final quantity="Volume", final unit="m3");  
  
type Time = Real (final quantity="Time", final unit="s");  
  
type Duration = Time;  
  
type AngularVelocity = Real (  
    final quantity="AngularVelocity",  
    final unit="rad/s",  
    displayUnit="rev/min");  
  
type AngularAcceleration = Real (final quantity="AngularAcceleration", final  
unit  
= "rad/s2");  
  
type Velocity = Real (final quantity="Velocity", final unit="m/s");  
  
type Acceleration = Real (final quantity="Acceleration", final unit="m/s2");  
  
type Period = Real (final quantity="Time", final unit="s");  
  
type Frequency = Real (final quantity="Frequency", final unit="Hz");  
  
type AngularFrequency = Real (final quantity="AngularFrequency", final unit=  
"s-1");  
  
type Wavelength = Real (final quantity="Wavelength", final unit="m");  
  
type Wavelenght = Wavelength;  
  
type WaveNumber = Real (final quantity="WaveNumber", final unit="m-1");  
  
type CircularWaveNumber = Real (final quantity="CircularWaveNumber", final  
unit  
= "rad/m");  
  
type AmplitudeLevelDifference = Real (final quantity=  
"AmplitudeLevelDifference", final unit="dB");  
  
type PowerLevelDifference = Real (final quantity="PowerLevelDifference",  
    final unit="dB");  
  
type DampingCoefficient = Real (final quantity="DampingCoefficient", final  
unit  
= "s-1");  
  
type LogarithmicDecrement = Real (final quantity="LogarithmicDecrement",  
    final unit="1/s");  
  
type AttenuationCoefficient = Real (final quantity="AttenuationCoefficient",  
    final unit="m-1");  
  
type PhaseCoefficient = Real (final quantity="PhaseCoefficient", final unit=  
"m-1");
```

```

type PropagationCoefficient = Real (final quantity="PropagationCoefficient",
                                     final unit="m-1");

type Damping = DampingCoefficient;

type Mass = Real (
  quantity="Mass",
  final unit="kg",
  min=0);

type Density = Real (
  final quantity="Density",
  final unit="kg/m3",
  displayUnit="g/cm3",
  min=0);

type RelativeDensity = Real (
  final quantity="RelativeDensity",
  final unit="1",
  min=0);

type SpecificVolume = Real (
  final quantity="SpecificVolume",
  final unit="m3/kg",
  min=0);

type LinearDensity = Real (
  final quantity="LinearDensity",
  final unit="kg/m",
  min=0);

type SurfaceDensity = Real (
  final quantity="SurfaceDensity",
  final unit="kg/m2",
  min=0);

type Momentum = Real (final quantity="Momentum", final unit="kg.m/s");

type Impulse = Real (final quantity="Impulse", final unit="N.s");

type AngularMomentum = Real (final quantity="AngularMomentum", final unit=
  "kg.m2/s");

type AngularImpulse = Real (final quantity="AngularImpulse", final unit=
  "N.m.s");

type MomentOfInertia = Real (final quantity="MomentOfInertia", final unit=
  "kg.m2");

type Inertia = MomentOfInertia;

type Force = Real (final quantity="Force", final unit="N");

type Weight = Force;

```

```
type Torque = Real (final quantity="Torque", final unit="N.m");

type MomentOfForce = Torque;

type Pressure = Real (
  final quantity="Pressure",
  final unit="Pa",
  displayUnit="bar");

type AbsolutePressure = Pressure (min=0);

type BulkModulus = AbsolutePressure;

type Stress = Real (final unit="Pa");

type NormalStress = Stress;

type ShearStress = Stress;

type Strain = Real (final quantity="Strain", final unit="1");

type LinearStrain = Strain;

type ShearStrain = Strain;

type VolumeStrain = Real (final quantity="VolumeStrain", final unit="1");

type PoissonNumber = Real (final quantity="PoissonNumber", final unit="1");

type ModulusOfElasticity = Stress;

type ShearModulus = Stress;

type SecondMomentOfArea = Real (final quantity="SecondMomentOfArea", final
unit
  =
  "m4"); 

type SecondPolarMomentOfArea = SecondMomentOfArea;

type SectionModulus = Real (final quantity="SectionModulus", final unit="m3");

type CoefficientOfFriction = Real (final quantity="CoefficientOfFriction",
  final unit="1");

type DynamicViscosity = Real (
  final quantity="DynamicViscosity",
  final unit="Pa.s",
  min=0);

type KinematicViscosity = Real (
  final quantity="KinematicViscosity",
  final unit="m2/s",
  min=0);
```

```

type SurfaceTension = Real (final quantity="SurfaceTension", final
unit="N/m");

type Work = Real (final quantity="Work", final unit="J");

type Energy = Real (final quantity="Energy", final unit="J");

type EnergyDensity = Real (final quantity="EnergyDensity", final unit="J/m3");

type PotentialEnergy = Energy;

type KineticEnergy = Energy;

type Power = Real (final quantity="Power", final unit="W");

type EnergyFlowRate = Power;

type EnthalpyFlowRate = Real (final quantity="EnthalpyFlowRate", final unit=
"W");

type Efficiency = Real (
  final quantity="Efficiency",
  final unit="1",
  min=0);

type MassFlowRate = Real (quantity="MassFlowRate", final unit="kg/s");

type VolumeFlowRate = Real (final quantity="VolumeFlowRate", final unit=
"m3/s");

type MomentumFlux = Real (final quantity="MomentumFlux", final unit="N");

type AngularMomentumFlux = Real (final quantity="AngularMomentumFlux", final
unit
=
"N.m");

type ThermodynamicTemperature = Real (
  final quantity="ThermodynamicTemperature",
  final unit="K",
  min = 0,
  displayUnit="degC");

type Temp_K = ThermodynamicTemperature;

type Temperature = ThermodynamicTemperature;

type TemperatureDifference = Real (
  final quantity="ThermodynamicTemperature",
  final unit="K");

type CelsiusTemperature = Real (final quantity="CelsiusTemperature", final
unit
=
"degC");

```

```
type Temp_C = CelsiusTemperature;

type LinearExpansionCoefficient = Real (final quantity=
    "LinearExpansionCoefficient", final unit="1/K");

type CubicExpansionCoefficient = Real (final quantity=
    "CubicExpansionCoefficient", final unit="1/K");

type RelativePressureCoefficient = Real (final quantity=
    "RelativePressureCoefficient", final unit="1/K");

type PressureCoefficient = Real (final quantity="PressureCoefficient", final
unit
= "Pa/K");

type Compressibility = Real (final quantity="Compressibility", final unit=
    "1/Pa");

type IsothermalCompressibility = Compressibility;

type IsentropicCompressibility = Compressibility;

type Heat = Real (final quantity="Energy", final unit="J");

type HeatFlowRate = Real (final quantity="Power", final unit="W");

type HeatFlux = Real (final quantity="HeatFlux", final unit="W/m2");

type DensityOfHeatFlowRate = Real (final quantity="DensityOfHeatFlowRate",
    final unit="W/m2");

type ThermalConductivity = Real (final quantity="ThermalConductivity", final
unit
= "W/(m.K)");

type CoefficientOfHeatTransfer = Real (final quantity=
    "CoefficientOfHeatTransfer", final unit="W/(m2.K)");

type SurfaceCoefficientOfHeatTransfer = CoefficientOfHeatTransfer;

type ThermalInsulance = Real (final quantity="ThermalInsulance", final unit=
    "m2.K/W");

type ThermalResistance = Real (final quantity="ThermalResistance", final unit
= "K/W");

type ThermalConductance = Real (final quantity="ThermalConductance", final
unit
= "W/K");

type ThermalDiffusivity = Real (final quantity="ThermalDiffusivity", final
unit
= "m2/s");
```

```

type HeatCapacity = Real (final quantity="HeatCapacity", final unit="J/K");

type SpecificHeatCapacity = Real (final quantity="SpecificHeatCapacity",
    final unit="J/(kg.K)");

type SpecificHeatCapacityAtConstantPressure = SpecificHeatCapacity;

type SpecificHeatCapacityAtConstantVolume = SpecificHeatCapacity;

type SpecificHeatCapacityAtSaturation = SpecificHeatCapacity;

type RatioOfSpecificHeatCapacities = Real (final quantity=
    "RatioOfSpecificHeatCapacities", final unit="1");

type IsentropicExponent = Real (final quantity="IsentropicExponent", final
unit
= "1");

type Entropy = Real (final quantity="Entropy", final unit="J/K");

type SpecificEntropy = Real (final quantity="SpecificEntropy", final unit=
    "J/(kg.K)");

type InternalEnergy = Heat;

type Enthalpy = Heat;

type HelmholtzFreeEnergy = Heat;

type GibbsFreeEnergy = Heat;

type SpecificEnergy = Real (final quantity="SpecificEnergy", final unit=
    "J/kg");

type SpecificInternalEnergy = SpecificEnergy;

type SpecificEnthalpy = SpecificEnergy;

type SpecificHelmholtzFreeEnergy = SpecificEnergy;

type SpecificGibbsFreeEnergy = SpecificEnergy;

type MassieuFunction = Real (final quantity="MassieuFunction", final unit=
    "J/K");

type PlanckFunction = Real (final quantity="PlanckFunction", final
unit="J/K");

type DerDensityByEnthalpy = Real (final unit="kg.s2/m5");

type DerDensityByPressure = Real (final unit="s2/m2");

```

```
type DerDensityByTemperature = Real (final unit="kg/(m3.K)") ;

type DerEnthalpyByPressure = Real (final unit="J.m.s2/kg2) ;

type DerEnergyByDensity = Real (final unit="J.m3/kg") ;

type DerEnergyByPressure = Real (final unit="J.m.s2/kg") ;

type ElectricCurrent = Real (final quantity="ElectricCurrent", final
unit="A") ;

type Current = ElectricCurrent ;

type ElectricCharge = Real (final quantity="ElectricCharge", final unit="C") ;

type Charge = ElectricCharge ;

type VolumeDensityOfCharge = Real (
  final quantity="VolumeDensityOfCharge",
  final unit="C/m3",
  min=0) ;

type SurfaceDensityOfCharge = Real (
  final quantity="SurfaceDensityOfCharge",
  final unit="C/m2",
  min=0) ;

type ElectricFieldStrength = Real (final quantity="ElectricFieldStrength",
  final unit="V/m") ;

type ElectricPotential = Real (final quantity="ElectricPotential", final unit
= "V") ;

type Voltage = ElectricPotential ;

type PotentialDifference = ElectricPotential ;

type ElectromotiveForce = ElectricPotential ;

type ElectricFluxDensity = Real (final quantity="ElectricFluxDensity", final
unit
= "C/m2") ;

type ElectricFlux = Real (final quantity="ElectricFlux", final unit="C") ;

type Capacitance = Real (
  final quantity="Capacitance",
  final unit="F",
  min=0) ;

type Permittivity = Real (
  final quantity="Permittivity",
  final unit="F/m",
  min=0) ;
```

```

type PermittivityOfVacuum = Permittivity;  

type RelativePermittivity = Real (final quantity="RelativePermittivity",
    final unit="1");  

type ElectricSusceptibility = Real (final quantity="ElectricSusceptibility",
    final unit="1");  

type ElectricPolarization = Real (final quantity="ElectricPolarization",
    final unit="C/m2");  

type Electrization = Real (final quantity="Electrization", final unit="V/m");  

type ElectricDipoleMoment = Real (final quantity="ElectricDipoleMoment",
    final unit="C.m");  

type CurrentDensity = Real (final quantity="CurrentDensity", final unit=
    "A/m2");  

type LinearCurrentDensity = Real (final quantity="LinearCurrentDensity",
    final unit="A/m");  

type MagneticFieldStrength = Real (final quantity="MagneticFieldStrength",
    final unit="A/m");  

type MagneticPotential = Real (final quantity="MagneticPotential", final unit
    =
    "A");  

type MagneticPotentialDifference = Real (final quantity=
    "MagneticPotential", final unit="A");  

type MagnetomotiveForce = Real (final quantity="MagnetomotiveForce", final unit
    =
    "A");  

type CurrentLinkage = Real (final quantity="CurrentLinkage", final unit="A");  

type MagneticFluxDensity = Real (final quantity="MagneticFluxDensity", final unit
    =
    "T");  

type MagneticFlux = Real (final quantity="MagneticFlux", final unit="Wb");  

type MagneticVectorPotential = Real (final quantity="MagneticVectorPotential",
    final unit="Wb/m");  

type Inductance = Real (
    final quantity="Inductance",
    final unit="H");  

type SelfInductance = Inductance(min=0);  

type MutualInductance = Inductance;

```

```
type CouplingCoefficient = Real (final quantity="CouplingCoefficient", final
unit
= "1");

type LeakageCoefficient = Real (final quantity="LeakageCoefficient", final
unit
= "1");

type Permeability = Real (final quantity="Permeability", final unit="H/m");

type PermeabilityOfVacuum = Permeability;

type RelativePermeability = Real (final quantity="RelativePermeability",
final unit="1");

type MagneticSusceptibility = Real (final quantity="MagneticSusceptibility",
final unit="1");

type ElectromagneticMoment = Real (final quantity="ElectromagneticMoment",
final unit="A.m2");

type MagneticDipoleMoment = Real (final quantity="MagneticDipoleMoment",
final unit="Wb.m");

type Magnetization = Real (final quantity="Magnetization", final unit="A/m");

type MagneticPolarization = Real (final quantity="MagneticPolarization",
final unit="T");

type ElectromagneticEnergyDensity = Real (final quantity="EnergyDensity",
final unit="J/m3");

type PoyntingVector = Real (final quantity="PoyntingVector", final unit=
"W/m2");

type Resistance = Real (
final quantity="Resistance",
final unit="Ohm");

type Resistivity = Real (final quantity="Resistivity", final unit="Ohm.m");

type Conductivity = Real (final quantity="Conductivity", final unit="S/m");

type Reluctance = Real (final quantity="Reluctance", final unit="H-1");

type Permeance = Real (final quantity="Permeance", final unit="H");

type PhaseDifference = Real (
final quantity="Angle",
final unit="rad",
displayUnit="deg");

type Impedance = Resistance;
```

```

type ModulusOfImpedance = Resistance;
type Reactance = Resistance;
type QualityFactor = Real (final quantity="QualityFactor", final unit="1");
type LossAngle = Real (
  final quantity="Angle",
  final unit="rad",
  displayUnit="deg");
type Conductance = Real (
  final quantity="Conductance",
  final unit="S");
type Admittance = Conductance;
type ModulusOfAdmittance = Conductance;
type Susceptance = Conductance;
type InstantaneousPower = Real (final quantity="Power", final unit="W");
type ActivePower = Real (final quantity="Power", final unit="W");
type ApparentPower = Real (final quantity="Power", final unit="VA");
type ReactivePower = Real (final quantity="Power", final unit="var");
type PowerFactor = Real (final quantity="PowerFactor", final unit="1");
type Transconductance = Real (final quantity="Transconductance", final unit=
  "A/V2");
type InversePotential = Real (final quantity="InversePotential", final unit=
  "1/V");
type RadiantEnergy = Real (final quantity="Energy", final unit="J");
type RadiantEnergyDensity = Real (final quantity="EnergyDensity", final unit=
  "J/m3");
type SpectralRadiantEnergyDensity = Real (final quantity=
  "SpectralRadiantEnergyDensity", final unit="J/m4");
type RadiantPower = Real (final quantity="Power", final unit="W");
type RadiantEnergyFluenceRate = Real (final quantity=
  "RadiantEnergyFluenceRate", final unit="W/m2");
type RadiantIntensity = Real (final quantity="RadiantIntensity", final unit=
  "W/sr");

```

```
type Radiance = Real (final quantity="Radiance", final unit="W/(sr.m2));  
  
type RadiantExtiance = Real (final quantity="RadiantExtiance", final unit= "W/m2");  
  
type Irradiance = Real (final quantity="Irradiance", final unit="W/m2");  
  
type Emissivity = Real (final quantity="Emissivity", final unit="1");  
  
type SpectralEmissivity = Real (final quantity="SpectralEmissivity", final unit  
= "1");  
  
type DirectionalSpectralEmissivity = Real (final quantity= "DirectionalSpectralEmissivity", final unit="1");  
  
type LuminousIntensity = Real (final quantity="LuminousIntensity", final unit  
= "cd");  
  
type LuminousFlux = Real (final quantity="LuminousFlux", final unit="lm");  
  
type QuantityOfLight = Real (final quantity="QuantityOfLight", final unit= "lm.s");  
  
type Luminance = Real (final quantity="Luminance", final unit="cd/m2");  
  
type LuminousExitance = Real (final quantity="LuminousExitance", final unit= "lm/m2");  
  
type Illuminance = Real (final quantity="Illuminance", final unit="lx");  
  
type LightExposure = Real (final quantity="LightExposure", final unit="lx.s");  
  
type LuminousEfficacy = Real (final quantity="LuminousEfficacy", final unit= "lm/W");  
  
type SpectralLuminousEfficacy = Real (final quantity= "SpectralLuminousEfficacy", final unit="lm/W");  
  
type LuminousEfficiency = Real (final quantity="LuminousEfficiency", final unit  
= "1");  
  
type SpectralLuminousEfficiency = Real (final quantity= "SpectralLuminousEfficiency", final unit="1");  
  
type CIESpectralTristimulusValues = Real (final quantity= "CIESpectralTristimulusValues", final unit="1");  
  
type ChromaticityCoordinates = Real (final quantity="ChromaticityCoordinates", final unit="1");  
  
type SpectralAbsorptionFactor = Real (final quantity= "SpectralAbsorptionFactor", final unit="1");
```

```

type SpectralReflectionFactor = Real (final quantity=
    "SpectralReflectionFactor", final unit="1");

type SpectralTransmissionFactor = Real (final quantity=
    "SpectralTransmissionFactor", final unit="1");

type SpectralRadianceFactor = Real (final quantity="SpectralRadianceFactor",
    final unit="1");

type LinearAttenuationCoefficient = Real (final quantity=
    "AttenuationCoefficient", final unit="m-1");

type LinearAbsorptionCoefficient = Real (final quantity=
    "LinearAbsorptionCoefficient", final unit="m-1");

type MolarAbsorptionCoefficient = Real (final quantity=
    "MolarAbsorptionCoefficient", final unit="m2/mol");

type RefractiveIndex = Real (final quantity="RefractiveIndex", final
unit="1");

type StaticPressure = Real (
    final quantity="Pressure",
    final unit="Pa",
    displayUnit="bar",
    min=0);

type SoundPressure = StaticPressure;

type SoundParticleDisplacement = Real (final quantity="Length", final unit=
    "m");

type SoundParticleVelocity = Real (final quantity="Velocity", final unit=
    "m/s");

type SoundParticleAcceleration = Real (final quantity="Acceleration", final
unit
    = "m/s2");

type VelocityOfSound = Real (final quantity="Velocity", final unit="m/s");

type SoundEnergyDensity = Real (final quantity="EnergyDensity", final unit=
    "J/m3");

type SoundPower = Real (final quantity="Power", final unit="W");

type SoundIntensity = Real (final quantity="SoundIntensity", final unit=
    "W/m2");

type AcousticImpedance = Real (final quantity="AcousticImpedance", final unit
    = "Pa.s/m3");

type SpecificAcousticImpedance = Real (final quantity=
    "SpecificAcousticImpedance", final unit="Pa.s/m");

```

```
type MechanicalImpedance = Real (final quantity="MechanicalImpedance", final
unit
= "N.s/m");

type SoundPressureLevel = Real (final quantity="SoundPressureLevel", final
unit
= "dB");

type SoundPowerLevel = Real (final quantity="SoundPowerLevel", final unit=
"dB");

type DissipationCoefficient = Real (final quantity="DissipationCoefficient",
final unit="1");

type ReflectionCoefficient = Real (final quantity="ReflectionCoefficient",
final unit="1");

type TransmissionCoefficient = Real (final quantity="TransmissionCoefficient",
final unit="1");

type AcousticAbsorptionCoefficient = Real (final quantity=
"AcousticAbsorptionCoefficient", final unit="1");

type SoundReductionIndex = Real (final quantity="SoundReductionIndex", final
unit
= "dB");

type EquivalentAbsorptionArea = Real (final quantity="Area", final unit="m2");

type ReverberationTime = Real (final quantity="Time", final unit="s");

type LoundnessLevel = Real (final quantity="LoundnessLevel", final unit=
"phon");

type Loundness = Real (final quantity="Loundness", final unit="sone");

type RelativeAtomicMass = Real (final quantity="RelativeAtomicMass", final
unit
= "1");

type RelativeMolecularMass = Real (final quantity="RelativeMolecularMass",
final unit="1");

type NumberOfMolecules = Real (final quantity="NumberOfMolecules", final unit
= "1");

type AmountOfSubstance = Real (
final quantity="AmountOfSubstance",
final unit="mol",
min=0);

type MolarMass = Real (final quantity="MolarMass", final unit="kg/mol");

type MolarVolume = Real (final quantity="MolarVolume", final unit="m3/mol");
```

```

type MolarInternalEnergy = Real (final quantity="MolarInternalEnergy", final
unit
= "J/mol");

type MolarHeatCapacity = Real (final quantity="MolarHeatCapacity", final unit
= "J/(mol.K)");

type MolarEntropy = Real (final quantity="MolarEntropy", final unit=
"J/(mol.K)");

type NumberDensityOfMolecules = Real (final quantity=
"NumberDensityOfMolecules", final unit="m-3");

type MolecularConcentration = Real (final quantity="MolecularConcentration",
final unit="m-3");

type MassConcentration = Real (final quantity="MassConcentration", final unit
= "kg/m3");

type MassFraction = Real (final quantity="MassFraction", final unit="1");

type Concentration = Real (final quantity="Concentration", final unit=
"mol/m3");

type VolumeFraction = Real (final quantity="VolumeFraction", final unit="1");

type MoleFraction = Real (final quantity="MoleFraction", final unit="1");

type ChemicalPotential = Real (final quantity="ChemicalPotential", final unit
= "J/mol");

type AbsoluteActivity = Real (final quantity="AbsoluteActivity", final unit=
"1");

type PartialPressure = Real (
  final quantity="Pressure",
  final unit="Pa",
  displayUnit="bar",
  min=0);

type Fugacity = Real (final quantity="Fugacity", final unit="Pa");

type StandardAbsoluteActivity = Real (final quantity=
"StandardAbsoluteActivity", final unit="1");

type ActivityCoefficient = Real (final quantity="ActivityCoefficient", final
unit
= "1");

type ActivityOfSolute = Real (final quantity="ActivityOfSolute", final unit=
"1");

type ActivityCoefficientOfSolute = Real (final quantity=
"ActivityCoefficientOfSolute", final unit="1");

```

```
type StandardAbsoluteActivityOfSolute = Real (final quantity=
    "StandardAbsoluteActivityOfSolute", final unit="1");

type ActivityOfSolvent = Real (final quantity="ActivityOfSolvent", final unit
= "1");

type OsmoticCoefficientOfSolvent = Real (final quantity=
    "OsmoticCoefficientOfSolvent", final unit="1");

type StandardAbsoluteActivityOfSolvent = Real (final quantity=
    "StandardAbsoluteActivityOfSolvent", final unit="1");

type OsmoticPressure = Real (
    final quantity="Pressure",
    final unit="Pa",
    displayUnit="bar",
    min=0);

type StoichiometricNumber = Real (final quantity="StoichiometricNumber",
    final unit="1");

type Affinity = Real (final quantity="Affinity", final unit="J/mol");

type MassOfMolecule = Real (final quantity="Mass", final unit="kg");

type ElectricDipoleMomentOfMolecule = Real (final quantity=
    "ElectricDipoleMomentOfMolecule", final unit="C.m");

type ElectricPolarizabilityOfAMolecule = Real (final quantity=
    "ElectricPolarizabilityOfAMolecule", final unit="C.m2/V");

type MicrocanonicalPartitionFunction = Real (final quantity=
    "MicrocanonicalPartitionFunction", final unit="1");

type CanonicalPartitionFunction = Real (final quantity=
    "CanonicalPartitionFunction", final unit="1");

type GrandCanonicalPartitionFunction = Real (final quantity=
    "GrandCanonicalPartitionFunction", final unit="1");

type MolecularPartitionFunction = Real (final quantity=
    "MolecularPartitionFunction", final unit="1");

type StatisticalWeight = Real (final quantity="StatisticalWeight", final unit
= "1");

type MeanFreePath = Length;

type DiffusionCoefficient = Real (final quantity="DiffusionCoefficient",
    final unit="m2/s");

type ThermalDiffusionRatio = Real (final quantity="ThermalDiffusionRatio",
    final unit="1");
```

```

type ThermalDiffusionFactor = Real (final quantity="ThermalDiffusionFactor",
                                    final unit="1");

type ThermalDiffusionCoefficient = Real (final quantity=
                                         "ThermalDiffusionCoefficient", final unit="m2/s");

type ElementaryCharge = Real (final quantity="ElementaryCharge", final unit=
                               "C");

type ChargeNumberOfIon = Real (final quantity="ChargeNumberOfIon", final unit=
                                "1");

type FaradayConstant = Real (final quantity="FaradayConstant", final unit=
                               "C/mol");

type IonicStrength = Real (final quantity="IonicStrength", final unit=
                            "mol/kg");

type DegreeOfDissociation = Real (final quantity="DegreeOfDissociation",
                                   final unit="1");

type ElectrolyticConductivity = Real (final quantity=
                                         "ElectrolyticConductivity", final unit="S/m");

type MolarConductivity = Real (final quantity="MolarConductivity", final unit=
                                "S.m2/mol");

type TransportNumberOfIonic = Real (final quantity="TransportNumberOfIonic",
                                   final unit="1");

type ProtonNumber = Real (final quantity="ProtonNumber", final unit="1");

type NeutronNumber = Real (final quantity="NeutronNumber", final unit="1");

type NucleonNumber = Real (final quantity="NucleonNumber", final unit="1");

type AtomicMassConstant = Real (final quantity="Mass", final unit="kg");

type MassOfElectron = Real (final quantity="Mass", final unit="kg");

type MassOfProton = Real (final quantity="Mass", final unit="kg");

type MassOfNeutron = Real (final quantity="Mass", final unit="kg");

type HartreeEnergy = Real (final quantity="Energy", final unit="J");

type MagneticMomentOfParticle = Real (final quantity=
                                         "MagneticMomentOfParticle", final unit="A.m2");

type BohrMagnetons = MagneticMomentOfParticle;

type NuclearMagnetons = MagneticMomentOfParticle;

```

```
type GyromagneticCoefficient = Real (final quantity="GyromagneticCoefficient",
                                     final unit="A.m2/(J.s)");

type GFactorOfAtom = Real (final quantity="GFactorOfAtom", final unit="1");

type GFactorOfNucleus = Real (final quantity="GFactorOfNucleus", final unit=
                               "1");

type LarmorAngularFrequency = Real (final quantity="AngularFrequency", final
unit
=           "s-1");

type NuclearPrecessionAngularFrequency = Real (final quantity=
                                               "AngularFrequency", final unit="s-1");

type CyclotronAngularFrequency = Real (final quantity="AngularFrequency",
                                       final unit="s-1");

type NuclearQuadrupoleMoment = Real (final quantity="NuclearQuadrupoleMoment",
                                       final unit="m2");

type NuclearRadius = Real (final quantity="Length", final unit="m");

type ElectronRadius = Real (final quantity="Length", final unit="m");

type ComptonWavelength = Real (final quantity="Length", final unit="m");

type MassExcess = Real (final quantity="Mass", final unit="kg");

type MassDefect = Real (final quantity="Mass", final unit="kg");

type RelativeMassExcess = Real (final quantity="RelativeMassExcess", final
unit
=           "1");

type RelativeMassDefect = Real (final quantity="RelativeMassDefect", final
unit
=           "1");

type PackingFraction = Real (final quantity="PackingFraction", final
unit="1");

type BindingFraction = Real (final quantity="BindingFraction", final
unit="1");

type MeanLife = Real (final quantity="Time", final unit="s");

type LevelWidth = Real (final quantity="LevelWidth", final unit="J");

type Activity = Real (final quantity="Activity", final unit="Bq");

type SpecificActivity = Real (final quantity="SpecificActivity", final unit=
                               "Bq/kg");
```

```

type DecayConstant = Real (final quantity="DecayConstant", final unit="s-1");

type HalfLife = Real (final quantity="Time", final unit="s");

type AlphaDisintegrationEnergy = Real (final quantity="Energy", final unit=
    "J");

type MaximumBetaParticleEnergy = Real (final quantity="Energy", final unit=
    "J");

type BetaDisintegrationEnergy = Real (final quantity="Energy", final unit="J");

type ReactionEnergy = Real (final quantity="Energy", final unit="J");

type ResonanceEnergy = Real (final quantity="Energy", final unit="J");

type CrossSection = Real (final quantity="Area", final unit="m2");

type TotalCrossSection = Real (final quantity="Area", final unit="m2");

type AngularCrossSection = Real (final quantity="AngularCrossSection", final
unit
    = "m2/sr");

type SpectralCrossSection = Real (final quantity="SpectralCrossSection",
    final unit="m2/J");

type SpectralAngularCrossSection = Real (final quantity=
    "SpectralAngularCrossSection", final unit="m2/(sr.J)");

type MacroscopicCrossSection = Real (final quantity="MacroscopicCrossSection",
    final unit="m-1");

type TotalMacroscopicCrossSection = Real (final quantity=
    "TotalMacroscopicCrossSection", final unit="m-1");

type ParticleFluence = Real (final quantity="ParticleFluence", final unit=
    "m-2");

type ParticleFluenceRate = Real (final quantity="ParticleFluenceRate", final
unit
    = "s-1.m2");

type EnergyFluence = Real (final quantity="EnergyFluence", final unit="J/m2");

type EnergyFluenceRate = Real (final quantity="EnergyFluenceRate", final unit
    = "W/m2");

type CurrentDensityOfParticles = Real (final quantity=
    "CurrentDensityOfParticles", final unit="m-2.s-1");

type MassAttenuationCoefficient = Real (final quantity=
    "MassAttenuationCoefficient", final unit="m2/kg");

```

```
type MolarAttenuationCoefficient = Real (final quantity=
    "MolarAttenuationCoefficient", final unit="m2/mol");

type AtomicAttenuationCoefficient = Real (final quantity=
    "AtomicAttenuationCoefficient", final unit="m2");

type HalfThickness = Real (final quantity="Length", final unit="m");

type TotalLinearStoppingPower = Real (final quantity=
    "TotalLinearStoppingPower", final unit="J/m");

type TotalAtomicStoppingPower = Real (final quantity=
    "TotalAtomicStoppingPower", final unit="J.m2");

type TotalMassStoppingPower = Real (final quantity="TotalMassStoppingPower",
    final unit="J.m2/kg");

type MeanLinearRange = Real (final quantity="Length", final unit="m");

type MeanMassRange = Real (final quantity="MeanMassRange", final
unit="kg/m2");

type LinearIonization = Real (final quantity="LinearIonization", final unit=
    "m-1");

type TotalIonization = Real (final quantity="TotalIonization", final
unit="1");

type Mobility = Real (final quantity="Mobility", final unit="m2/(V.s)");

type IonNumberDensity = Real (final quantity="IonNumberDensity", final unit=
    "m-3");

type RecombinationCoefficient = Real (final quantity=
    "RecombinationCoefficient", final unit="m3/s");

type NeutronNumberDensity = Real (final quantity="NeutronNumberDensity",
    final unit="m-3");

type NeutronSpeed = Real (final quantity="Velocity", final unit="m/s");

type NeutronFluenceRate = Real (final quantity="NeutronFluenceRate", final
unit
    =
        "s-1.m-2");

type TotalNeutronSourceDensity = Real (final quantity=
    "TotalNeutronSourceDesity", final unit="s-1.m-3");

type SlowingDownDensity = Real (final quantity="SlowingDownDensity", final
unit
    =
        "s-1.m-3");

type ResonanceEscapeProbability = Real (final quantity=
    "ResonanceEscapeProbability", final unit="1");
```

```

type Lethargy = Real (final quantity="Lethargy", final unit="1");

type SlowingDownArea = Real (final quantity="Area", final unit="m2");

type DiffusionArea = Real (final quantity="Area", final unit="m2");

type MigrationArea = Real (final quantity="Area", final unit="m2");

type SlowingDownLength = Real (final quantity="SLength", final unit="m");

type DiffusionLength = Length;

type MigrationLength = Length;

type NeutronYieldPerFission = Real (final quantity="NeutronYieldPerFission",
                                    final unit="1");

type NeutronYieldPerAbsorption = Real (final quantity=
                                         "NeutronYieldPerAbsorption", final unit="1");

type FastFissionFactor = Real (final quantity="FastFissionFactor", final unit
                               = "1");

type ThermalUtilizationFactor = Real (final quantity=
                                         "ThermalUtilizationFactor", final unit="1");

type NonLeakageProbability = Real (final quantity="NonLeakageProbability",
                                    final unit="1");

type Reactivity = Real (final quantity="Reactivity", final unit="1");

type ReactorTimeConstant = Real (final quantity="Time", final unit="s");

type EnergyImparted = Real (final quantity="Energy", final unit="J");

type MeanEnergyImparted = Real (final quantity="Energy", final unit="J");

type SpecificEnergyImparted = Real (final quantity="SpecificEnergy", final
unit
                               = "Gy");

type AbsorbedDose = Real (final quantity="AbsorbedDose", final unit="Gy");

type DoseEquivalent = Real (final quantity="DoseEquivalent", final unit="Sv");

type AbsorbedDoseRate = Real (final quantity="AbsorbedDoseRate", final unit=
                               "Gy/s");

type LinearEnergyTransfer = Real (final quantity="LinearEnergyTransfer",
                                 final unit="J/m");

type Kerma = Real (final quantity="Kerma", final unit="Gy");

```

```
type KermaRate = Real (final quantity="KermaRate", final unit="Gy/s");

type MassEnergyTransferCoefficient = Real (final quantity=
    "MassEnergyTransferCoefficient", final unit="m2/kg");

type Exposure = Real (final quantity="Exposure", final unit="C/kg");

type ExposureRate = Real (final quantity="ExposureRate", final unit=
    "C/(kg.s)");

type ReynoldsNumber = Real (final quantity="ReynoldsNumber", final unit="1");

type EulerNumber = Real (final quantity="EulerNumber", final unit="1");

type FroudeNumber = Real (final quantity="FroudeNumber", final unit="1");

type GrashofNumber = Real (final quantity="GrashofNumber", final unit="1");

type WeberNumber = Real (final quantity="WeberNumber", final unit="1");

type MachNumber = Real (final quantity="MachNumber", final unit="1");

type KnudsenNumber = Real (final quantity="KnudsenNumber", final unit="1");

type StrouhalNumber = Real (final quantity="StrouhalNumber", final unit="1");

type FourierNumber = Real (final quantity="FourierNumber", final unit="1");

type PecletNumber = Real (final quantity="PecletNumber", final unit="1");

type RayleighNumber = Real (final quantity="RayleighNumber", final unit="1");

type NusseltNumber = Real (final quantity="NusseltNumber", final unit="1");

type BiotNumber = NusseltNumber;

type StantonNumber = Real (final quantity="StantonNumber", final unit="1");

type FourierNumberOfMassTransfer = Real (final quantity=
    "FourierNumberOfMassTransfer", final unit="1");

type PecletNumberOfMassTransfer = Real (final quantity=
    "PecletNumberOfMassTransfer", final unit="1");

type GrashofNumberOfMassTransfer = Real (final quantity=
    "GrashofNumberOfMassTransfer", final unit="1");

type NusseltNumberOfMassTransfer = Real (final quantity=
    "NusseltNumberOfMassTransfer", final unit="1");

type StantonNumberOfMassTransfer = Real (final quantity=
    "StantonNumberOfMassTransfer", final unit="1");
```

```

type PrandtlNumber = Real (final quantity="PrandtlNumber", final unit="1");

type SchmidtNumber = Real (final quantity="SchmidtNumber", final unit="1");

type LewisNumber = Real (final quantity="LewisNumber", final unit="1");

type MagneticReynoldsNumber = Real (final quantity="MagneticReynoldsNumber",
                                    final unit="1");

type AlfvenNumber = Real (final quantity="AlfvenNumber", final unit="1");

type HartmannNumber = Real (final quantity="HartmannNumber", final unit="1");

type CowlingNumber = Real (final quantity="CowlingNumber", final unit="1");

type BraggAngle = Angle;

type OrderOfReflexion = Real (final quantity="OrderOfReflexion", final unit=
                            "1");

type ShortRangeOrderParameter = Real (final quantity="RangeOrderParameter",
                                       final unit="1");

type LongRangeOrderParameter = Real (final quantity="RangeOrderParameter",
                                       final unit="1");

type DebyeWallerFactor = Real (final quantity="DebyeWallerFactor", final unit
                               = "1");

type CircularWavenumber = Real (final quantity="CircularWavenumber", final
                                 unit
                               = "m-1");

type FermiCircularWavenumber = Real (final quantity="FermiCircularWavenumber",
                                       final unit="m-1");

type DebyeCircularWavenumber = Real (final quantity="DebyeCircularWavenumber",
                                       final unit="m-1");

type DebyeCircularFrequency = Real (final quantity="AngularFrequency", final
                                   unit
                               = "s-1");

type DebyeTemperature = ThermodynamicTemperature;

type SpectralConcentration = Real (final quantity="SpectralConcentration",
                                   final unit="s/m3");

type GrueneisenParameter = Real (final quantity="GrueneisenParameter", final
                                 unit
                               = "1");

type MadelungConstant = Real (final quantity="MadelungConstant", final unit=
                                "1");

```

```
type DensityOfStates = Real (final quantity="DensityOfStates", final unit=
    "J-1/m-3");

type ResidualResistivity = Real (final quantity="ResidualResistivity", final
unit
    = "Ohm.m");

type LorenzCoefficient = Real (final quantity="LorenzCoefficient", final unit
    = "V2/K2");

type HallCoefficient = Real (final quantity="HallCoefficient", final unit=
    "m3/C");

type ThermoelectromotiveForce = Real (final quantity=
    "ThermoelectromotiveForce", final unit="V");

type SeebeckCoefficient = Real (final quantity="SeebeckCoefficient", final
unit
    = "V/K");

type PeltierCoefficient = Real (final quantity="PeltierCoefficient", final
unit
    = "V");

type ThomsonCoefficient = Real (final quantity="ThomsonCoefficient", final
unit
    = "V/K");

type RichardsonConstant = Real (final quantity="RichardsonConstant", final
unit
    = "A/(m2.K2)");

type FermiEnergy = Real (final quantity="Energy", final unit="eV");

type GapEnergy = Real (final quantity="Energy", final unit="eV");

type DonorIonizationEnergy = Real (final quantity="Energy", final unit="eV");

type AcceptorIonizationEnergy = Real (final quantity="Energy", final unit=
    "eV");

type FermiTemperature = ThermodynamicTemperature;

type ElectronNumberDensity = Real (final quantity="ElectronNumberDensity",
    final unit="m-3");

type HoleNumberDensity = Real (final quantity="HoleNumberDensity", final unit
    = "m-3");

type IntrinsicNumberDensity = Real (final quantity="IntrinsicNumberDensity",
    final unit="m-3");

type DonorNumberDensity = Real (final quantity="DonorNumberDensity", final
unit
```

```

= "m-3");

type AcceptorNumberDensity = Real (final quantity="AcceptorNumberDensity",
final unit="m-3");

type EffectiveMass = Mass;

type MobilityRatio = Real (final quantity="MobilityRatio", final unit="1");

type RelaxationTime = Time;

type CarrierLifeTime = Time;

type ExchangeIntegral = Real (final quantity="Energy", final unit="eV");

type CurieTemperature = ThermodynamicTemperature;

type NeelTemperature = ThermodynamicTemperature;

type LondonPenetrationDepth = Length;

type CoherenceLength = Length;

type LandauGinzburgParameter = Real (final quantity="LandauGinzburgParameter",
final unit="1");

type FluxiodQuantum = Real (final quantity="FluxiodQuantum", final unit="Wb");

```

---

## Modelica.Slunits.UsersGuide

### User's Guide of Slunits Library

Library **Slunits** is a **free** Modelica package providing predefined types, such as *Mass*, *Length*, *Time*, based on the international standard on units.



### Package Content

Name	Description
i HowToUseSlunits	How to use Slunits
i Conventions	Conventions
i Literature	Literature
i Contact	Contact

---

## Modelica.Slunits.UsersGuide.HowToUseSlunits

When implementing a Modelica model, every variable needs to be declared. Physical variables should be declared with a unit. The basic approach in Modelica is that the unit attribute of a variable is the **unit** in which the **equations** are **written**, for example:

```
model MassOnGround
```



```
parameter Real m(quantity="Mass", unit="kg") "Mass";
parameter Real f(quantity="Force", unit="N") "Driving force";
Real s(unit="m") "Position of mass";
Real v(unit="m/s") "Velocity of mass";
equation
  der(s) = v;
  m*der(v) = f;
end MassOnGround;
```

This means that the equations in the equation section are only correct for the specified units. A different issue is the user interface, i.e., in which unit the variable is presented to the user in graphical user interfaces, both for input (e.g., parameter menu), as well as for output (e.g., in the plot window). Preferably, the Modelica tool should provide a list of units from which the user can select, e.g., "m", "cm", "km", "inch" for quantity "Length". When storing the value in the model as a Modelica modifier, it has to be converted to the unit defined in the declaration. Additionally, the unit used in the graphical user interface has to be stored. In order to have a standardized way of doing this, Modelica provides the following three attributes for a variable of type Real:

- **quantity** to define the physical quantity (e.g. "Length", or "Energy").
- **unit** to define the unit that has to be used in order that the equations are correct (e.g. "N.m").
- **displayUnit** to define the unit used in the graphical user interface as default display unit for input and/or output.

Note, a unit, such as "N.m", is not sufficient to define uniquely the physical quantity, since, e.g., "N.m" might be either "torque" or "energy". The "quantity" attribute might therefore be used by a tool to select the corresponding menu from which the user can select a unit for the corresponding variable.

For example, after providing a value for "m" and "f" in a parameter menu of an instance of MassOnGround, a tool might generate the following code:

```
MassOnGround myObject(m(displayUnit="g")=2, f=3);
```

The meaning is that in the equations a value of "2" is used and that in the graphical user interface a value of "2000" should be used, together with the unit "g" from the unit set "Mass" (= the quantity name). Note, according to the Modelica specification a tool might ignore the "displayUnit" attribute.

In order to help the Modelica model developer, the Modelica.SIunits library provides about 450 predefined type names, together with values for the attributes **quantity**, **unit** and sometimes **displayUnit** and **min**. The unit is always selected as SI-unit according to the ISO standard. The type and the quantity names are the quantity names used in the ISO standard. "quantity" and "unit" are defined as "**final**" in order that they cannot be modified. Attributes "displayUnit" and "min" can, however, be changed in a model via a modification. The example above, might therefore be alternatively also defined as:

```
model MassOnGround
  parameter Modelica.SIunits.Mass m "Mass";
  parameter Modelica.SIunits.Force f "Driving force";
  ...
end MassOnGround;
```

or in a short hand notation as

```
model MassOnGround
  import SI = Modelica.SIunits;
  parameter SI.Mass m "Mass";
  parameter SI.Force f "Driving force";
  ...
end MassOnGround;
```

For some often used Non SI-units (like hour), some additional type definitions are present as Modelica.SIunits.Conversions.NonSIunits. If this is not sufficient, the user has to define its own types or use the attributes directly in the declaration as in the example at the beginning.

## Modelica.Slunits.UsersGuide.Conventions

The following conventions are used in package Slunits:

- Modelica quantity names are defined according to the recommendations of ISO 31. Some of these name are rather long, such as "ThermodynamicTemperature". Shorter alias names are defined, e.g., "type Temp\_K = ThermodynamicTemperature;".
- Modelica units are defined according to the SI base units without multiples (only exception "kg").
- For some quantities, more convenient units for an engineer are defined as "displayUnit", i.e., the default unit for display purposes (e.g., displayUnit="deg" for quantity="Angle").
- The type name is identical to the quantity name, following the convention of type names.
- All quantity and unit attributes are defined as final in order that they cannot be redefined to another value.
- Similar quantities, such as "Length, Breadth, Height, Thickness, Radius" are defined as the same quantity (here: "Length").
- The ordering of the type declarations in this package follows ISO 31:

```
Chapter 1: Space and Time
Chapter 2: Periodic and Related Phenomena
Chapter 3: Mechanics
Chapter 4: Heat
Chapter 5: Electricity and Magnetism
Chapter 6: Light and Related Electro-Magnetic Radiations
Chapter 7: Acoustics
Chapter 8: Physical Chemistry
Chapter 9: Atomic and Nuclear Physics
Chapter 10: Nuclear Reactions and Ionizing Radiations
Chapter 11: (not defined in ISO 31-1992)
Chapter 12: Characteristic Numbers
Chapter 13: Solid State Physics
```

- Conversion functions between SI and non-SI units are available in subpackage **Conversions**.



## Modelica.Slunits.UsersGuide.Literature

This package is based on the following references



ISO 31-1992:

**General principles concerning quantities, units and symbols.**

ISO 1000-1992:

**SI units and recommendations for the use of their multiples and of certain other units.**

Cardarelli F.:

**Scientific Unit Conversion - A Practical Guide to Metrication.** Springer 1997.

## Modelica.Slunits.UsersGuide.Contact

**Main Author:**

[Martin Otter](#)

Deutsches Zentrum für Luft und Raumfahrt e.V. (DLR)

Institut für Robotik und Mechatronik

Abteilung für Entwurfsorientierte Regelungstechnik

Postfach 1116



D-82230 Wessling  
Germany  
email: [Martin.Otter@dlr.de](mailto:Martin.Otter@dlr.de)

### Acknowledgements:

- Astrid Jaschinski, Hubertus Tummescheit and Christian Schweiger contributed to the implementation of this package.
- 

## Modelica.Slunits.Conversions

### Conversion functions to/from non SI units and type definitions of non SI units

#### Information

This package provides conversion functions from the non SI Units defined in package Modelica.Slunits.Conversions.NonSlunits to the corresponding SI Units defined in package Modelica.Slunits and vice versa. It is recommended to use these functions in the following way (note, that all functions have one Real input and one Real output argument):

```
import SI = Modelica.SIunits;
import Modelica.SIunits.Conversions.*;
...
parameter SI.Temperature      T    = from_degC(25);    // convert 25 degree
Celsius to Kelvin
parameter SI.Angle              phi = from_deg(180);   // convert 180 degree to
radian
parameter SI.AngularVelocity w   = from_rpm(3600);   // convert 3600
revolutions per minutes
                                         // to radian per seconds
```

#### Package Content

Name	Description
 NonSlunits	Type definitions of non SI units
 to_degC	Convert from Kelvin to °Celsius
 from_degC	Convert from °Celsius to Kelvin
 to_degF	Convert from Kelvin to °Fahrenheit
 from_degF	Convert from °Fahrenheit to Kelvin
 to_degRk	Convert from Kelvin to °Rankine
 from_degRk	Convert from °Rankine to Kelvin
 to_deg	Convert from radian to degree
 from_deg	Convert from degree to radian
 to_rpm	Convert from radian per second to revolutions per minute
 from_rpm	Convert from revolutions per minute to radian per second
 to_kmh	Convert from metre per second to kilometre per hour
 from_kmh	Convert from kilometre per hour to metre per second
 to_day	Convert from second to day

	Convert from day to second
	Convert from second to hour
	Convert from hour to second
	Convert from second to minute
	Convert from minute to second
	Convert from cubic metre to litre
	Convert from litre to cubic metre
	Convert from Joule to kilo Watt hour
	Convert from kilo Watt hour to Joule
	Convert from Pascal to bar
	Convert from bar to Pascal
	Convert from kilogram per second to gram per second
	Convert from gram per second to kilogram per second
	Base icon for conversion functions

## Modelica.Slunits.Conversions.NonSlunits

### Type definitions of non SI units

#### Information

This package provides predefined types, such as **Angle\_deg** (angle in degree), **AngularVelocity\_rpm** (angular velocity in revolutions per minute) or **Temperature\_degF** (temperature in degree Fahrenheit), which are in common use but are not part of the international standard on units according to ISO 31-1992 "General principles concerning quantities, units and symbols" and ISO 1000-1992 "SI units and recommendations for the use of their multiples and of certain other units".

If possible, the types in this package should not be used. Use instead types of package Modelica.Slunits. For more information on units, see also the book of Francois Cardarelli **Scientific Unit Conversion - A Practical Guide to Metrification** (Springer 1997).

Some units, such as **Temperature\_degC/Temp\_C** are both defined in Modelica.Slunits and in Modelica.Conversions.NonSlunits. The reason is that these definitions have been placed erroneously in Modelica.Slunits although they are not Slunits. For backward compatibility, these type definitions are still kept in Modelica.Slunits.

#### Package Content

Name	Description
Temperature_degC	
Temperature_degF	
Temperature_degRk	
Angle_deg	
AngularVelocity_rpm	
Velocity_kmh	
Time_day	
Time_hour	

Time_minute	
Volume_litre	
Energy_kWh	
Pressure_bar	
MassFlowRate_gps	

### Types and constants

```
type Temperature_degC = Real (final quantity="ThermodynamicTemperature",
    final unit="degC");

type Temperature_degF = Real (final quantity="ThermodynamicTemperature",
    final unit="degF");

type Temperature_degRk = Real (final quantity="ThermodynamicTemperature",
    final unit="degRk");

type Angle_deg = Real (final quantity="Angle", final unit="deg");

type AngularVelocity_rpm = Real (final quantity="AngularVelocity", final unit
    = "rev/min");

type Velocity_kmh = Real (final quantity="Velocity", final unit="km/h");

type Time_day = Real (final quantity="Time", final unit="d");

type Time_hour = Real (final quantity="Time", final unit="h");

type Time_minute = Real (final quantity="Time", final unit="min");

type Volume_litre = Real (final quantity="Volume", final unit="l");

type Energy_kWh = Real (final quantity="Energy", final unit="kW.h");

type Pressure_bar = Real (final quantity="Pressure", final unit="bar");

type MassFlowRate_gps = Real (final quantity="MassFlowRate", final unit
    = "g/s");
```

---

### Modelica.Slunits.Conversions.NonSlunits.Temperature\_degC

---

### Modelica.Slunits.Conversions.NonSlunits.Temperature\_degF

---

### Modelica.Slunits.Conversions.NonSlunits.Temperature\_degRk

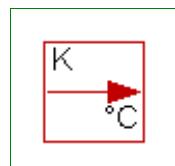
---

**Modelica.Slunits.Conversions.NonSlunits.Angle\_deg****Modelica.Slunits.Conversions.NonSlunits.AngularVelocity\_rpm****Modelica.Slunits.Conversions.NonSlunits.Velocity\_kmh****Modelica.Slunits.Conversions.NonSlunits.Time\_day****Modelica.Slunits.Conversions.NonSlunits.Time\_hour****Modelica.Slunits.Conversions.NonSlunits.Time\_minute****Modelica.Slunits.Conversions.NonSlunits.Volume\_litre****Modelica.Slunits.Conversions.NonSlunits.Energy\_kWh****Modelica.Slunits.Conversions.NonSlunits.Pressure\_bar****Modelica.Slunits.Conversions.NonSlunits.MassFlowRate\_gps****Modelica.Slunits.Conversions.to\_degC**

Convert from Kelvin to °Celsius

**Inputs**

Name	Default	Description
Kelvin		Kelvin value [K]

**Outputs**

Name	Description
Celsius	Celsius value [degC]

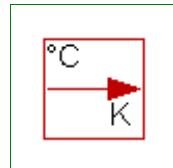
---

## 1748 Modelica.Slunits.Conversions.from\_degC

---

### Modelica.Slunits.Conversions.from\_degC

Convert from °Celsius to Kelvin



#### Inputs

Name	Default	Description
Celsius		Celsius value [degC]

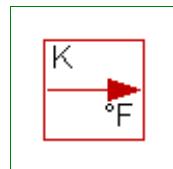
#### Outputs

Name	Description
Kelvin	Kelvin value [K]

---

### Modelica.Slunits.Conversions.to\_degF

Convert from Kelvin to °Fahrenheit



#### Inputs

Name	Default	Description
Kelvin		Kelvin value [K]

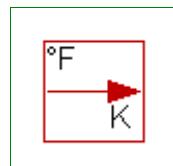
#### Outputs

Name	Description
Fahrenheit	Fahrenheit value [degF]

---

### Modelica.Slunits.Conversions.from\_degF

Convert from °Fahrenheit to Kelvin



#### Inputs

Name	Default	Description
Fahrenheit		Fahrenheit value [degF]

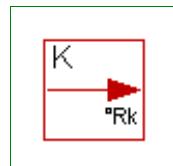
#### Outputs

Name	Description
Kelvin	Kelvin value [K]

---

### Modelica.Slunits.Conversions.to\_degRk

Convert from Kelvin to °Rankine



#### Inputs

Name	Default	Description
Kelvin		Kelvin value [K]

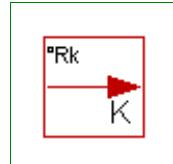
## Outputs

Name	Description
Rankine	Rankine value [degRk]

---

## Modelica.Slunits.Conversions.from\_degRk

Convert from °Rankine to Kelvin



## Inputs

Name	Default	Description
Rankine		Rankine value [degRk]

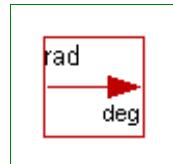
## Outputs

Name	Description
Kelvin	Kelvin value [K]

---

## Modelica.Slunits.Conversions.to\_deg

Convert from radian to degree



## Inputs

Name	Default	Description
radian		radian value [rad]

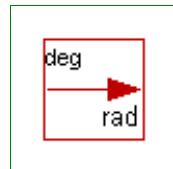
## Outputs

Name	Description
degree	degree value [deg]

---

## Modelica.Slunits.Conversions.from\_deg

Convert from degree to radian



## Inputs

Name	Default	Description
degree		degree value [deg]

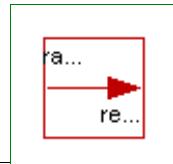
## Outputs

Name	Description
radian	radian value [rad]

---

## Modelica.Slunits.Conversions.to\_rpm

Convert from radian per second to revolutions per minute



## 1750 Modelica.Slunits.Conversions.to\_rpm

---

### Inputs

Name	Default	Description
rs		radian per second value [rad/s]

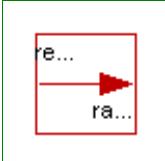
### Outputs

Name	Description
rpm	revolutions per minute value [rev/min]

---

## Modelica.Slunits.Conversions.from\_rpm

Convert from revolutions per minute to radian per second



### Inputs

Name	Default	Description
rpm		revolutions per minute value [rev/min]

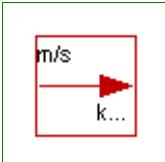
### Outputs

Name	Description
rs	radian per second value [rad/s]

---

## Modelica.Slunits.Conversions.to\_kmh

Convert from metre per second to kilometre per hour



### Inputs

Name	Default	Description
ms		metre per second value [m/s]

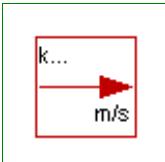
### Outputs

Name	Description
kmh	kilometre per hour value [km/h]

---

## Modelica.Slunits.Conversions.from\_kmh

Convert from kilometre per hour to metre per second



### Inputs

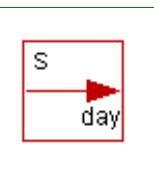
Name	Default	Description
kmh		kilometre per hour value [km/h]

### Outputs

Name	Description
ms	metre per second value [m/s]

**Modelica.Slunits.Conversions.to\_day**

Convert from second to day

**Inputs**

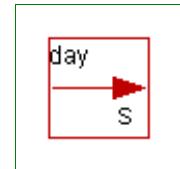
Name	Default	Description
s		second value [s]

**Outputs**

Name	Description
day	day value [d]

**Modelica.Slunits.Conversions.from\_day**

Convert from day to second

**Inputs**

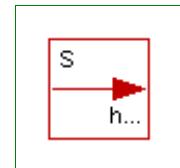
Name	Default	Description
day		day value [d]

**Outputs**

Name	Description
s	second value [s]

**Modelica.Slunits.Conversions.to\_hour**

Convert from second to hour

**Inputs**

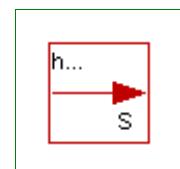
Name	Default	Description
s		second value [s]

**Outputs**

Name	Description
hour	hour value [h]

**Modelica.Slunits.Conversions.from\_hour**

Convert from hour to second

**Inputs**

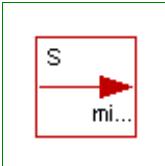
Name	Default	Description
hour		hour value [h]

## 1752 Modelica.Slunits.Conversions.from\_hour

---

### Outputs

Name	Description
s	second value [s]



## Modelica.Slunits.Conversions.to\_minute

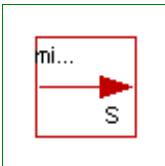
Convert from second to minute

### Inputs

Name	Default	Description
s		second value [s]

### Outputs

Name	Description
minute	minute value [min]



## Modelica.Slunits.Conversions.from\_minute

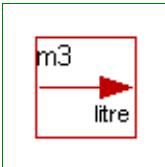
Convert from minute to second

### Inputs

Name	Default	Description
minute		minute value [min]

### Outputs

Name	Description
s	second value [s]



## Modelica.Slunits.Conversions.to\_litre

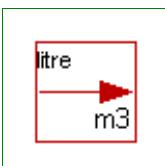
Convert from cubic metre to litre

### Inputs

Name	Default	Description
m3		cubic metre value [m3]

### Outputs

Name	Description
litre	litre value [l]



## Modelica.Slunits.Conversions.from\_litre

Convert from litre to cubic metre

## Inputs

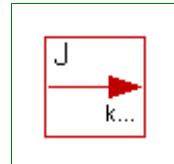
Name	Default	Description
litre		litre value [l]

## Outputs

Name	Description
m3	cubic metre value [m <sup>3</sup> ]

## Modelica.Slunits.Conversions.to\_kWh

Convert from Joule to kilo Watt hour



## Inputs

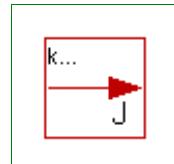
Name	Default	Description
J		Joule value [J]

## Outputs

Name	Description
kWh	kWh value [kW.h]

## Modelica.Slunits.Conversions.from\_kWh

Convert from kilo Watt hour to Joule



## Inputs

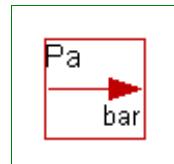
Name	Default	Description
kWh		kWh value [kW.h]

## Outputs

Name	Description
J	Joule value [J]

## Modelica.Slunits.Conversions.to\_bar

Convert from Pascal to bar



## Inputs

Name	Default	Description
Pa		Pascal value [Pa]

## Outputs

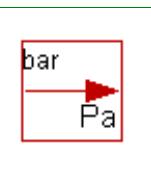
Name	Description
bar	bar value [bar]

## 1754 Modelica.Slunits.Conversions.to\_bar

---

### Modelica.Slunits.Conversions.from\_bar

Convert from bar to Pascal



#### Inputs

Name	Default	Description
bar		bar value [bar]

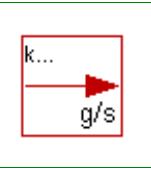
#### Outputs

Name	Description
Pa	Pascal value [Pa]

---

### Modelica.Slunits.Conversions.to\_gps

Convert from kilogram per second to gram per second



#### Inputs

Name	Default	Description
kgps		kg/s value [kg/s]

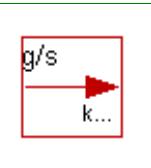
#### Outputs

Name	Description
gps	g/s value [g/s]

---

### Modelica.Slunits.Conversions.from\_gps

Convert from gram per second to kilogram per second



#### Inputs

Name	Default	Description
gps		g/s value [g/s]

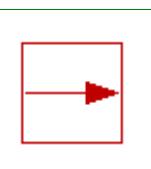
#### Outputs

Name	Description
kgps	kg/s value [kg/s]

---

### Modelica.Slunits.Conversions.ConversionIcon

Base icon for conversion functions



**Modelica.Slunits.Angle****Parameters**

Name	Default	Description
displayUnit	"deg"	

---

**Modelica.Slunits.SolidAngle**

---

**Modelica.Slunits.Length**

---

**Modelica.Slunits.PathLength**

---

**Modelica.Slunits.Position**

---

**Modelica.Slunits.Distance****Parameters**

Name	Default	Description
min	0	

---

**Modelica.Slunits.Breadth**

---

**Parameters**

Name	Default	Description
min	0	

---

**Modelica.Slunits.Height**

---

**Parameters**

Name	Default	Description
min	0	

**Modelica.Slunits.Thickness****Parameters**

Name	Default	Description
min	0	

---

**Modelica.Slunits.Radius****Parameters**

Name	Default	Description
min	0	

---

**Modelica.Slunits.Diameter****Parameters**

Name	Default	Description
min	0	

---

**Modelica.Slunits.Area****Modelica.Slunits.Volume****Modelica.Slunits.Time****Modelica.Slunits.Duration****Modelica.Slunits.AngularVelocity****Parameters**

Name	Default	Description
displayUnit	"rev/min"	

---

**Modelica.Slunits.AngularAcceleration**

**Modelica.Slunits.Velocity**

---

**Modelica.Slunits.Acceleration**

---

**Modelica.Slunits.Period**

---

**Modelica.Slunits.Frequency**

---

**Modelica.Slunits.AngularFrequency**

---

**Modelica.Slunits.Wavelength**

---

**Modelica.Slunits.Wavelenght**

---

**Modelica.Slunits.WaveNumber**

---

**Modelica.Slunits.CircularWaveNumber**

---

**Modelica.Slunits.AmplitudeLevelDifference**

---

**Modelica.Slunits.PowerLevelDifference**

---

**Modelica.Slunits.DampingCoefficient**

---

**Modelica.Slunits.LogarithmicDecrement**

---

**Modelica.Slunits.AttenuationCoefficient**

---

**Modelica.Slunits.PhaseCoefficient**

---

**Modelica.Slunits.PropagationCoefficient**

---

**Modelica.Slunits.Damping**

---

**Modelica.Slunits.Mass****Parameters**

Name	Default	Description
quantity	"Mass"	
min	0	

---

**Modelica.Slunits.Density****Parameters**

Name	Default	Description
displayUnit	"g/cm3"	
min	0	

---

**Modelica.Slunits.RelativeDensity****Parameters**

Name	Default	Description
min	0	

---

**Modelica.Slunits.SpecificVolume****Parameters**

Name	Default	Description
min	0	

---

**Modelica.Slunits.LinearDensity****Parameters**

Name	Default	Description

---

---

min	0	
-----	---	--

---

## Modelica.Slunits.SurfaceDensity

### Parameters

Name	Default	Description
min	0	

---

## Modelica.Slunits.Momentum

---

## Modelica.Slunits.Impulse

---

## Modelica.Slunits.AngularMomentum

---

## Modelica.Slunits.AngularImpulse

---

## Modelica.Slunits.MomentOfInertia

---

## Modelica.Slunits.Inertia

---

## Modelica.Slunits.Force

---

## Modelica.Slunits.Weight

---

## Modelica.Slunits.Torque

---

## Modelica.Slunits.MomentOfForce

**Modelica.Slunits.Pressure****Parameters**

Name	Default	Description
displayUnit	"bar"	

---

**Modelica.Slunits.AbsolutePressure****Parameters**

Name	Default	Description
min	0	

---

**Modelica.Slunits.BulkModulus****Modelica.Slunits.Stress****Modelica.Slunits.NormalStress****Modelica.Slunits.ShearStress****Modelica.Slunits.Strain****Modelica.Slunits.LinearStrain****Modelica.Slunits.ShearStrain****Modelica.Slunits.VolumeStrain****Modelica.Slunits.PoissonNumber****Modelica.Slunits.ModulusOfElasticity**

---

**Modelica.Slunits.ShearModulus**

---

**Modelica.Slunits.SecondMomentOfArea**

---

**Modelica.Slunits.SecondPolarMomentOfArea**

---

**Modelica.Slunits.SectionModulus**

---

**Modelica.Slunits.CoefficientOfFriction**

---

**Modelica.Slunits.DynamicViscosity****Parameters**

Name	Default	Description
min	0	

---

**Modelica.Slunits.KinematicViscosity****Parameters**

Name	Default	Description
min	0	

---

**Modelica.Slunits.SurfaceTension****Modelica.Slunits.Work****Modelica.Slunits.Energy****Modelica.Slunits.EnergyDensity****Modelica.Slunits.PotentialEnergy**

**Modelica.Slunits.KineticEnergy**

---

**Modelica.Slunits.Power**

---

**Modelica.Slunits.EnergyFlowRate**

---

**Modelica.Slunits.EnthalpyFlowRate**

---

**Modelica.Slunits.Efficiency**

---

**Parameters**

Name	Default	Description
min	0	

---

**Modelica.Slunits.MassFlowRate**

---

**Parameters**

Name	Default	Description
quantity	"MassFlowRate"	

---

**Modelica.Slunits.VolumeFlowRate**

---

**Modelica.Slunits.MomentumFlux**

---

**Modelica.Slunits.AngularMomentumFlux**

---

**Modelica.Slunits.ThermodynamicTemperature**

---

**Parameters**

Name	Default	Description
displayUnit	"degC"	
min	0	

---

[Modelica.Slunits.Temp\\_K](#)

---

[Modelica.Slunits.Temperature](#)

---

[Modelica.Slunits.TemperatureDifference](#)

---

[Modelica.Slunits.CelsiusTemperature](#)

---

[Modelica.Slunits.Temp\\_C](#)

---

[Modelica.Slunits.LinearExpansionCoefficient](#)

---

[Modelica.Slunits.CubicExpansionCoefficient](#)

---

[Modelica.Slunits.RelativePressureCoefficient](#)

---

[Modelica.Slunits.PressureCoefficient](#)

---

[Modelica.Slunits.Compressibility](#)

---

[Modelica.Slunits.IsothermalCompressibility](#)

---

[Modelica.Slunits.IsentropicCompressibility](#)

---

[Modelica.Slunits.Heat](#)

---

[Modelica.Slunits.HeatFlowRate](#)

---

**Modelica.Slunits.HeatFlux**

---

**Modelica.Slunits.DensityOfHeatFlowRate**

---

**Modelica.Slunits.ThermalConductivity**

---

**Modelica.Slunits.CoefficientOfHeatTransfer**

---

**Modelica.Slunits.SurfaceCoefficientOfHeatTransfer**

---

**Modelica.Slunits.ThermalInsulance**

---

**Modelica.Slunits.ThermalResistance**

---

**Modelica.Slunits.ThermalConductance**

---

**Modelica.Slunits.ThermalDiffusivity**

---

**Modelica.Slunits.HeatCapacity**

---

**Modelica.Slunits.SpecificHeatCapacity**

---

**Modelica.Slunits.SpecificHeatCapacityAtConstantPressure**

---

**Modelica.Slunits.SpecificHeatCapacityAtConstantVolume**

---

**Modelica.Slunits.SpecificHeatCapacityAtSaturation**

---

**Modelica.Slunits.RatioOfSpecificHeatCapacities**

---

**Modelica.Slunits.IsentropicExponent**

---

**Modelica.Slunits.Entropy**

---

**Modelica.Slunits.SpecificEntropy**

---

**Modelica.Slunits.InternalEnergy**

---

**Modelica.Slunits.Enthalpy**

---

**Modelica.Slunits.HelmholtzFreeEnergy**

---

**Modelica.Slunits.GibbsFreeEnergy**

---

**Modelica.Slunits.SpecificEnergy**

---

**Modelica.Slunits.SpecificInternalEnergy**

---

**Modelica.Slunits.SpecificEnthalpy**

---

**Modelica.Slunits.SpecificHelmholtzFreeEnergy**

---

**Modelica.Slunits.SpecificGibbsFreeEnergy**

---

**Modelica.Slunits.MassieuFunction**

---

**Modelica.Slunits.PlanckFunction**

---

**Modelica.Slunits.DerDensityByEnthalpy**

---

**Modelica.Slunits.DerDensityByPressure**

---

**Modelica.Slunits.DerDensityByTemperature**

---

**Modelica.Slunits.DerEnthalpyByPressure**

---

**Modelica.Slunits.DerEnergyByDensity**

---

**Modelica.Slunits.DerEnergyByPressure**

---

**Modelica.Slunits.ElectricCurrent**

---

**Modelica.Slunits.Current**

---

**Modelica.Slunits.ElectricCharge**

---

**Modelica.Slunits.Charge**

---

**Modelica.Slunits.VolumeDensityOfCharge**

---

#### Parameters

Name	Default	Description
min	0	

---

**Modelica.Slunits.SurfaceDensityOfCharge****Parameters**

Name	Default	Description
min	0	

---

**Modelica.Slunits.ElectricFieldStrength****Modelica.Slunits.ElectricPotential****Modelica.Slunits.Voltage****Modelica.Slunits.PotentialDifference****Modelica.Slunits.ElectromotiveForce****Modelica.Slunits.ElectricFluxDensity****Modelica.Slunits.ElectricFlux****Modelica.Slunits.Capacitance****Parameters**

Name	Default	Description
min	0	

---

**Modelica.Slunits.Permittivity****Parameters**

Name	Default	Description
min	0	

---

**Modelica.Slunits.PermittivityOfVacuum**

---

**Modelica.Slunits.RelativePermittivity**

---

**Modelica.Slunits.ElectricSusceptibility**

---

**Modelica.Slunits.ElectricPolarization**

---

**Modelica.Slunits.Electrization**

---

**Modelica.Slunits.ElectricDipoleMoment**

---

**Modelica.Slunits.CurrentDensity**

---

**Modelica.Slunits.LinearCurrentDensity**

---

**Modelica.Slunits.MagneticFieldStrength**

---

**Modelica.Slunits.MagneticPotential**

---

**Modelica.Slunits.MagneticPotentialDifference**

---

**Modelica.Slunits.MagnetomotiveForce**

---

**Modelica.Slunits.CurrentLinkage**

---

**Modelica.Slunits.MagneticFluxDensity**

---

**Modelica.Slunits.MagneticFlux**

---

**Modelica.Slunits.MagneticVectorPotential**

---

**Modelica.Slunits.Inductance**

---

**Modelica.Slunits.SelfInductance**

---

#### Parameters

Name	Default	Description
min	0	

---

**Modelica.Slunits.MutualInductance**

---

**Modelica.Slunits.CouplingCoefficient**

---

**Modelica.Slunits.LeakageCoefficient**

---

**Modelica.Slunits.Permeability**

---

**Modelica.Slunits.PermeabilityOfVacuum**

---

**Modelica.Slunits.RelativePermeability**

---

**Modelica.Slunits.MagneticSusceptibility**

---

**Modelica.Slunits.ElectromagneticMoment**

---

**Modelica.Slunits.MagneticDipoleMoment**

---

**Modelica.Slunits.Magnetization**

---

**Modelica.Slunits.MagneticPolarization**

---

**Modelica.Slunits.ElectromagneticEnergyDensity**

---

**Modelica.Slunits.PoyntingVector**

---

**Modelica.Slunits.Resistance**

---

**Modelica.Slunits.Resistivity**

---

**Modelica.Slunits.Conductivity**

---

**Modelica.Slunits.Reluctance**

---

**Modelica.Slunits.Permeance**

---

**Modelica.Slunits.PhaseDifference**

---

**Parameters**

Name	Default	Description
displayUnit	"deg"	

---

**Modelica.Slunits.Impedance**

---

**Modelica.Slunits.ModulusOfImpedance**

---

**Modelica.Slunits.Reactance**

---

**Modelica.Slunits.QualityFactor**

---

**Modelica.Slunits.LossAngle****Parameters**

Name	Default	Description
displayUnit	"deg"	

---

**Modelica.Slunits.Conductance**

---

**Modelica.Slunits.Admittance**

---

**Modelica.Slunits.ModulusOfAdmittance**

---

**Modelica.Slunits.Susceptance**

---

**Modelica.Slunits.InstantaneousPower**

---

**Modelica.Slunits.ActivePower**

---

**Modelica.Slunits.ApparentPower**

---

**Modelica.Slunits.ReactivePower**

---

**Modelica.Slunits.PowerFactor**

---

**Modelica.Slunits.Transconductance**

---

**Modelica.Slunits.InversePotential**

**Modelica.Slunits.RadiantEnergy**

---

**Modelica.Slunits.RadiantEnergyDensity**

---

**Modelica.Slunits.SpectralRadiantEnergyDensity**

---

**Modelica.Slunits.RadiantPower**

---

**Modelica.Slunits.RadiantEnergyFluenceRate**

---

**Modelica.Slunits.RadiantIntensity**

---

**Modelica.Slunits.Radiance**

---

**Modelica.Slunits.RadiantExtiance**

---

**Modelica.Slunits.Irradiance**

---

**Modelica.Slunits.Emissivity**

---

**Modelica.Slunits.SpectralEmissivity**

---

**Modelica.Slunits.DirectionalSpectralEmissivity**

---

**Modelica.Slunits.LuminousIntensity**

---

**Modelica.Slunits.LuminousFlux**

---

[\*\*Modelica.Slunits.QuantityOfLight\*\*](#)

---

[\*\*Modelica.Slunits.Luminance\*\*](#)

---

[\*\*Modelica.Slunits.LuminousExitance\*\*](#)

---

[\*\*Modelica.Slunits.Illuminance\*\*](#)

---

[\*\*Modelica.Slunits.LightExposure\*\*](#)

---

[\*\*Modelica.Slunits.LuminousEfficacy\*\*](#)

---

[\*\*Modelica.Slunits.SpectralLuminousEfficacy\*\*](#)

---

[\*\*Modelica.Slunits.LuminousEfficiency\*\*](#)

---

[\*\*Modelica.Slunits.SpectralLuminousEfficiency\*\*](#)

---

[\*\*Modelica.Slunits.CIESpectralTristimulusValues\*\*](#)

---

[\*\*Modelica.Slunits.ChromaticityCoordinates\*\*](#)

---

[\*\*Modelica.Slunits.SpectralAbsorptionFactor\*\*](#)

---

[\*\*Modelica.Slunits.SpectralReflectionFactor\*\*](#)

---

[\*\*Modelica.Slunits.SpectralTransmissionFactor\*\*](#)

---

**Modelica.Slunits.SpectralRadianceFactor**

---

**Modelica.Slunits.LinearAttenuationCoefficient**

---

**Modelica.Slunits.LinearAbsorptionCoefficient**

---

**Modelica.Slunits.MolarAbsorptionCoefficient**

---

**Modelica.Slunits.RefractiveIndex**

---

**Modelica.Slunits.StaticPressure****Parameters**

Name	Default	Description
displayUnit	"bar"	
min	0	

---

**Modelica.Slunits.SoundPressure**

---

**Modelica.Slunits.SoundParticleDisplacement**

---

**Modelica.Slunits.SoundParticleVelocity**

---

**Modelica.Slunits.SoundParticleAcceleration**

---

**Modelica.Slunits.VelocityOfSound**

---

**Modelica.Slunits.SoundEnergyDensity**

---

**Modelica.Slunits.SoundPower**

---

**Modelica.Slunits.SoundIntensity**

---

**Modelica.Slunits.AcousticImpedance**

---

**Modelica.Slunits.SpecificAcousticImpedance**

---

**Modelica.Slunits.MechanicalImpedance**

---

**Modelica.Slunits.SoundPressureLevel**

---

**Modelica.Slunits.SoundPowerLevel**

---

**Modelica.Slunits.DissipationCoefficient**

---

**Modelica.Slunits.ReflectionCoefficient**

---

**Modelica.Slunits.TransmissionCoefficient**

---

**Modelica.Slunits.AcousticAbsorptionCoefficient**

---

**Modelica.Slunits.SoundReductionIndex**

---

**Modelica.Slunits.EquivalentAbsorptionArea**

---

**Modelica.Slunits.ReverberationTime**

---

**Modelica.Slunits.LoundnessLevel**

---

**Modelica.Slunits.Loundness**

---

**Modelica.Slunits.RelativeAtomicMass**

---

**Modelica.Slunits.RelativeMolecularMass**

---

**Modelica.Slunits.NumberOfMolecules**

---

**Modelica.Slunits.AmountOfSubstance**

---

**Parameters**

Name	Default	Description
min	0	

---

**Modelica.Slunits.MolarMass**

---

**Modelica.Slunits.MolarVolume**

---

**Modelica.Slunits.MolarInternalEnergy**

---

**Modelica.Slunits.MolarHeatCapacity**

---

**Modelica.Slunits.MolarEntropy**

---

**Modelica.Slunits.NumberDensityOfMolecules**

---

**Modelica.Slunits.MolecularConcentration**

---

**Modelica.Slunits.MassConcentration**

---

**Modelica.Slunits.MassFraction**

---

**Modelica.Slunits.Concentration**

---

**Modelica.Slunits.VolumeFraction**

---

**Modelica.Slunits.MoleFraction**

---

**Modelica.Slunits.ChemicalPotential**

---

**Modelica.Slunits.AbsoluteActivity**

---

**Modelica.Slunits.PartialPressure**

---

#### Parameters

Name	Default	Description
displayUnit	"bar"	
min	0	

---

**Modelica.Slunits.Fugacity**

---

**Modelica.Slunits.StandardAbsoluteActivity**

---

**Modelica.Slunits.ActivityCoefficient**

---

**Modelica.Slunits.ActivityOfSolute**

---

**Modelica.Slunits.ActivityCoefficientOfSolute**

---

**Modelica.Slunits.StandardAbsoluteActivityOfSolute**

---

**Modelica.Slunits.ActivityOfSolvent**

---

**Modelica.Slunits.OsmoticCoefficientOfSolvent**

---

**Modelica.Slunits.StandardAbsoluteActivityOfSolvent**

---

**Modelica.Slunits.OsmoticPressure**

**Parameters**

Name	Default	Description
displayUnit	"bar"	
min	0	

---

**Modelica.Slunits.StoichiometricNumber**

---

**Modelica.Slunits.Affinity**

---

**Modelica.Slunits.MassOfMolecule**

---

**Modelica.Slunits.ElectricDipoleMomentOfMolecule**

---

**Modelica.Slunits.ElectricPolarizabilityOfAMolecule**

---

**Modelica.Slunits.MicrocanonicalPartitionFunction**

---

**Modelica.Slunits.CanonicalPartitionFunction**

---

**Modelica.Slunits.GrandCanonicalPartitionFunction**

---

**Modelica.Slunits.MolecularPartitionFunction**

---

**Modelica.Slunits.StatisticalWeight**

---

**Modelica.Slunits.MeanFreePath**

---

**Modelica.Slunits.DiffusionCoefficient**

---

**Modelica.Slunits.ThermalDiffusionRatio**

---

**Modelica.Slunits.ThermalDiffusionFactor**

---

**Modelica.Slunits.ThermalDiffusionCoefficient**

---

**Modelica.Slunits.ElementaryCharge**

---

**Modelica.Slunits.ChargeNumberOflon**

---

**Modelica.Slunits.FaradayConstant**

---

**Modelica.Slunits.IonicStrength**

---

**Modelica.Slunits.DegreeOfDissociation**

---

**Modelica.Slunits.ElectrolyticConductivity**

---

**Modelica.Slunits.MolarConductivity**

---

**Modelica.Slunits.TransportNumberOfIonic**

---

**Modelica.Slunits.ProtonNumber**

---

**Modelica.Slunits.NeutronNumber**

---

**Modelica.Slunits.NucleonNumber**

---

**Modelica.Slunits.AtomicMassConstant**

---

**Modelica.Slunits.MassOfElectron**

---

**Modelica.Slunits.MassOfProton**

---

**Modelica.Slunits.MassOfNeutron**

---

**Modelica.Slunits.HartreeEnergy**

---

**Modelica.Slunits.MagneticMomentOfParticle**

---

**Modelica.Slunits.BohrMagnetons**

---

**Modelica.Slunits.NuclearMagnetons**

---

**Modelica.Slunits.GyromagneticCoefficient**

---

**Modelica.Slunits.GFactorOfAtom**

---

**Modelica.Slunits.GFactorOfNucleus**

---

**Modelica.Slunits.LarmorAngularFrequency**

---

**Modelica.Slunits.NuclearPrecessionAngularFrequency**

---

**Modelica.Slunits.CyclotronAngularFrequency**

---

**Modelica.Slunits.NuclearQuadrupoleMoment**

---

**Modelica.Slunits.NuclearRadius**

---

**Modelica.Slunits.ElectronRadius**

---

**Modelica.Slunits.ComptonWavelength**

---

**Modelica.Slunits.MassExcess**

---

**Modelica.Slunits.MassDefect**

---

**Modelica.Slunits.RelativeMassExcess**

---

**Modelica.Slunits.RelativeMassDefect**

---

**Modelica.Slunits.PackingFraction**

---

**Modelica.Slunits.BindingFraction**

---

**Modelica.Slunits.MeanLife**

---

**Modelica.Slunits.LevelWidth**

---

**Modelica.Slunits.Activity**

---

**Modelica.Slunits.SpecificActivity**

---

**Modelica.Slunits.DecayConstant**

---

**Modelica.Slunits.HalfLife**

---

**Modelica.Slunits.AlphaDisintegrationEnergy**

---

**Modelica.Slunits.MaximumBetaParticleEnergy**

---

**Modelica.Slunits.BetaDisintegrationEnergy**

---

**Modelica.Slunits.ReactionEnergy**

---

**Modelica.Slunits.ResonanceEnergy**

---

**Modelica.Slunits.CrossSection**

---

**Modelica.Slunits.TotalCrossSection**

---

**Modelica.Slunits.AngularCrossSection**

---

**Modelica.Slunits.SpectralCrossSection**

---

**Modelica.Slunits.SpectralAngularCrossSection**

---

**Modelica.Slunits.MacroscopicCrossSection**

---

**Modelica.Slunits.TotalMacroscopicCrossSection**

---

**Modelica.Slunits.ParticleFluence**

---

**Modelica.Slunits.ParticleFluenceRate**

---

**Modelica.Slunits.EnergyFluence**

---

**Modelica.Slunits.EnergyFluenceRate**

---

**Modelica.Slunits.CurrentDensityOfParticles**

---

**Modelica.Slunits.MassAttenuationCoefficient**

---

**Modelica.Slunits.MolarAttenuationCoefficient**

---

**Modelica.Slunits.AtomicAttenuationCoefficient**

---

**Modelica.Slunits.HalfThickness**

---

**Modelica.Slunits.TotalLinearStoppingPower**

---

**Modelica.Slunits.TotalAtomicStoppingPower**

---

**Modelica.Slunits.TotalMassStoppingPower**

---

**Modelica.Slunits.MeanLinearRange**

---

**Modelica.Slunits.MeanMassRange**

---

**Modelica.Slunits.LinearIonization**

---

**Modelica.Slunits.TotalIonization**

---

**Modelica.Slunits.Mobility**

---

**Modelica.Slunits.IonNumberDensity**

---

**Modelica.Slunits.RecombinationCoefficient**

---

**Modelica.Slunits.NeutronNumberDensity**

---

**Modelica.Slunits.NeutronSpeed**

---

**Modelica.Slunits.NeutronFluenceRate**

---

**Modelica.Slunits.TotalNeutronSourceDensity**

---

**Modelica.Slunits.SlowingDownDensity**

---

**Modelica.Slunits.ResonanceEscapeProbability**

---

**Modelica.Slunits.Lethargy**

---

**Modelica.Slunits.SlowingDownArea**

---

**Modelica.Slunits.DiffusionArea**

---

**Modelica.Slunits.MigrationArea**

---

**Modelica.Slunits.SlowingDownLength**

---

**Modelica.Slunits.DiffusionLength**

---

**Modelica.Slunits.MigrationLength**

---

**Modelica.Slunits.NeutronYieldPerFission**

---

**Modelica.Slunits.NeutronYieldPerAbsorption**

---

**Modelica.Slunits.FastFissionFactor**

---

**Modelica.Slunits.ThermalUtilizationFactor**

---

**Modelica.Slunits.NonLeakageProbability**

---

**Modelica.Slunits.Reactivity**

---

**Modelica.Slunits.ReactorTimeConstant**

---

**Modelica.Slunits.EnergyImparted**

---

**Modelica.Slunits.MeanEnergyImparted**

---

**Modelica.Slunits.SpecificEnergyImparted**

---

**Modelica.Slunits.AbsorbedDose**

---

**Modelica.Slunits.DoseEquivalent**

---

**Modelica.Slunits.AbsorbedDoseRate**

---

**Modelica.Slunits.LinearEnergyTransfer**

---

**Modelica.Slunits.Kerma**

---

**Modelica.Slunits.KermaRate**

---

**Modelica.Slunits.MassEnergyTransferCoefficient**

---

**Modelica.Slunits.Exposure**

---

**Modelica.Slunits.ExposureRate**

---

**Modelica.Slunits.ReynoldsNumber**

---

**Modelica.Slunits.EulerNumber**

---

**Modelica.Slunits.FroudeNumber**

---

**Modelica.Slunits.GrashofNumber**

---

**Modelica.Slunits.WeberNumber**

---

**Modelica.Slunits.MachNumber**

---

**Modelica.Slunits.KnudsenNumber**

---

**Modelica.Slunits.StrouhalNumber**

---

**Modelica.Slunits.FourierNumber**

---

**Modelica.Slunits.PecletNumber**

---

**Modelica.Slunits.RayleighNumber**

---

**Modelica.Slunits.NusseltNumber**

---

**Modelica.Slunits.BiotNumber**

---

**Modelica.Slunits.StantonNumber**

---

**Modelica.Slunits.FourierNumberOfMassTransfer**

---

**Modelica.Slunits.PecletNumberOfMassTransfer**

---

**Modelica.Slunits.GrahschofNumberOfMassTransfer**

---

**Modelica.Slunits.NusseltNumberOfMassTransfer**

---

**Modelica.Slunits.StantonNumberOfMassTransfer**

---

**Modelica.Slunits.PrandtlNumber**

---

**Modelica.Slunits.SchmidtNumber**

---

**Modelica.Slunits.LewisNumber**

---

**Modelica.Slunits.MagneticReynoldsNumber**

---

**Modelica.Slunits.AlfvenNumber**

---

**Modelica.Slunits.HartmannNumber**

---

**Modelica.Slunits.CowlingNumber**

---

**Modelica.Slunits.BraggAngle**

---

**Modelica.Slunits.OrderOfReflexion**

---

**Modelica.Slunits.ShortRangeOrderParameter**

---

**Modelica.Slunits.LongRangeOrderParameter**

---

**Modelica.Slunits.DebyeWallerFactor**

---

**Modelica.Slunits.CircularWavenumber**

---

**Modelica.Slunits.FermiCircularWavenumber**

---

**Modelica.Slunits.DebyeCircularWavenumber**

---

**Modelica.Slunits.DebyeCircularFrequency**

---

**Modelica.Slunits.DebyeTemperature**

---

**Modelica.Slunits.SpectralConcentration**

---

**Modelica.Slunits.GrueneisenParameter**

---

**Modelica.Slunits.MadelungConstant**

---

**Modelica.Slunits.DensityOfStates**

---

**Modelica.Slunits.ResidualResistivity**

---

**Modelica.Slunits.LorenzCoefficient**

---

**Modelica.Slunits.HallCoefficient**

---

**Modelica.Slunits.ThermoelectromotiveForce**

---

**Modelica.Slunits.SeebbeckCoefficient**

---

**Modelica.Slunits.PeltierCoefficient**

---

**Modelica.Slunits.ThomsonCoefficient**

---

**Modelica.Slunits.RichardsonConstant**

---

**Modelica.Slunits.FermiEnergy**

---

**Modelica.Slunits.GapEnergy**

---

**Modelica.Slunits.DonorIonizationEnergy**

---

**Modelica.Slunits.AcceptorIonizationEnergy**

---

**Modelica.Slunits.FermiTemperature**

---

**Modelica.Slunits.ElectronNumberDensity**

---

**Modelica.Slunits.HoleNumberDensity**

---

**Modelica.Slunits.IntrinsicNumberDensity**

---

**Modelica.Slunits.DonorNumberDensity**

---

**Modelica.Slunits.AcceptorNumberDensity**

---

**Modelica.Slunits.EffectiveMass**

---

**Modelica.Slunits.MobilityRatio**

---

**Modelica.Slunits.RelaxationTime**

---

**Modelica.Slunits.CarrierLifetime**

---

**Modelica.Slunits.ExchangeIntegral**

---

**Modelica.Slunits.CurieTemperature**

---

**Modelica.Slunits.NeelTemperature**

---

**Modelica.Slunits.LondonPenetrationDepth**

---

**Modelica.Slunits.CoherenceLength**

---

**Modelica.Slunits.LandauGinzburgParameter**

---

**Modelica.Slunits.FluxiodQuantum**

---

## Modelica.StateGraph

Library of hierarchical state machine components to model discrete event and reactive systems

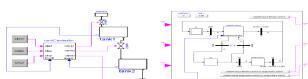
### Information

Library **StateGraph** is a **free** Modelica package providing components to model **discrete event** and **reactive** systems in a convenient way. It is based on the JGraphChart method and takes advantage of Modelica features for the "action" language. JGraphChart is a further development of Grafset to include elements of StateCharts that are not present in Grafset/Sequential Function Charts. Therefore, the StateGraph library has a similar modeling power as StateCharts but avoids some deficiencies of StateCharts.

For an introduction, have especially a look at:

- [StateGraph.UsersGuide](#) discusses the most important aspects how to use this library.
- [StateGraph.Examples](#) contains examples that demonstrate the usage of this library.

A typical model generated with this library is shown in the next figure where on the left hand side a two-tank system with a tank controller and on the right hand side the top-level part of the tank controller as a StateGraph is shown:



The unique feature of the StateGraph library with respect to JGraphCharts, Grafset, Sequential Function Charts, and StateCharts, is Modelica's "single assignment rule" that requires that every variable is defined by exactly one equation. This leads to a different "action" definition as in these formalisms. The advantage is that the translator can either determine a useful evaluation sequence by equation sorting or reports an error if this is not possible, e.g., because a model would lead to a non-determinism or to a dead-lock. As a side effect, this leads also to simpler and more easier to understand models and global variables are no longer needed (whereas in JGraphCharts, Grafset, Sequential Function Charts and StateCharts global variables are nearly always needed).

The StateGraph library is currently available in a beta release. The available components will most likely not be changed for the release version. It is planned to improve the convenience of building models with the StateGraph library for the release version (this may require to introduce some additional annotations). It is planned to include the StateGraph library in the Modelica standard library. It is most useful to combine this library with the Modelica libraries

- [Modelica.Blocks.Logical](#) that provides components available in PLCs (programmable logic controllers).
- [UserInteraction](#) that provides components to interactively communicate with models in a running simulation.

Copyright © 1998-2007, Modelica Association and DLR

*This Modelica package is **free** software; it can be redistributed and/or modified under the terms of the [Modelica license](#), see the license conditions and the accompanying [disclaimer](#) [here](#).*

### Package Content

Name	Description
<a href="#">UsersGuide</a>	User's Guide of StateGraph Library
<a href="#">Examples</a>	Examples to demonstrate the usage of the components of the StateGraph library
<a href="#">Interfaces</a>	Connectors and partial models
<a href="#">InitialStep</a>	Initial step (= step that is active when simulation starts)
<a href="#">InitialStepWithSignal</a>	Initial step (= step that is active when simulation starts). Connector 'active' is true

	when the step is active
 Step	Ordinary step (= step that is not active when simulation starts)
 StepWithSignal	Ordinary step (= step that is not active when simulation starts). Connector 'active' is true when the step is active
 Transition	Transition where the fire condition is set by a modification of variable condition
 TransitionWithSignal	Transition where the fire condition is set by a Boolean input signal
 Alternative	Alternative splitting of execution path (use component between two steps)
 Parallel	Parallel splitting of execution path (use component between two transitions)
 PartialCompositeStep	Superclass of a subgraph, i.e., a composite step that has internally a StateGraph
 StateGraphRoot	Root of a StateGraph (has to be present on the highest level of a StateGraph)
 Temporary	Components that will be provided by other libraries in the future

## Modelica.StateGraph.UsersGuide

Library **StateGraph** is a **free** Modelica package providing components to model **discrete event** and **reactive** systems in a convenient way. This package contains the **user's Guide** for the library and has the following content:



1. [Overview of library](#) gives an overview of the library.
2. [A first example](#) demonstrates at hand of a first example how to use this library.
3. [An application example](#) demonstrates varies features at hand of the control of a two tank system.
4. [Release Notes](#) summarizes the differences between different versions of this library.
5. [Literature](#) provides references that have been used to design and implement this library.
6. [Contact](#) provides information about the authors of the library as well as acknowledgments.

## Modelica.StateGraph.Examples

Examples to demonstrate the usage of the components of the StateGraph library

### Information

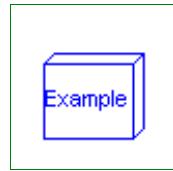
### Package Content

Name	Description
 FirstExample	A first simple StateGraph example
 FirstExample_Variant2	A variant of the first simple StateGraph example
 FirstExample_Variant3	A variant of the first simple StateGraph example
 ExecutionPaths	Example to demonstrate parallel and alternative execution paths
 ShowCompositeStep	Example to demonstrate parallel activities described by a StateGraph
 ShowExceptions	Example to demonstrate how a hierarchically structured StateGraph can suspend and resume actions on different levels
 ControlledTanks	Demonstrating the controller of a tank filling/emptying system
 Utilities	Utility components for the examples

**Modelica.StateGraph.Examples.FirstExample**

A first simple StateGraph example

Information

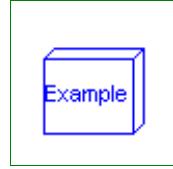


---

**Modelica.StateGraph.Examples.FirstExample\_Variant2**

A variant of the first simple StateGraph example

Information

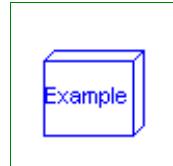


---

**Modelica.StateGraph.Examples.FirstExample\_Variant3**

A variant of the first simple StateGraph example

Information

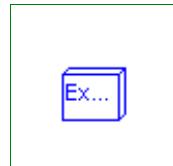


---

**Modelica.StateGraph.Examples.ExecutionPaths**

Example to demonstrate parallel and alternative execution paths

Information



This is an example to demonstrate in which way **parallel** activities can be modelled by a StateGraph. When transition1 fires (after 1 second), two branches are executed in parallel. After 6 seconds the two branches are synchronized in order to arrive at step6.

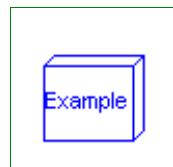
Before simulating the model, try to figure out whether which branch of the alternative sequence is executed. Note, that alternatives have priorities according to the port index (alternative.split[1] has a higher priority to fire as alternative.split[2]).

---

**Modelica.StateGraph.Examples>ShowCompositeStep**

Example to demonstrate parallel activities described by a StateGraph

Information

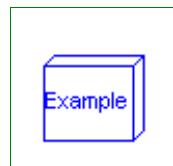


This is the same example as "ExecutionPaths". The only difference is that the alternative paths are included in a "CompositeStep".

---

**Modelica.StateGraph.Examples>ShowExceptions**

Example to demonstrate how a hierarchically structured StateGraph can suspend and resume actions on different levels

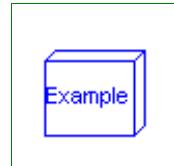


## Information

CompositeStep "compositeStep" is a hierarchical StateGraph consisting of two other subgraphs. Whenever component "compositeStep" is suspended, all steps within "compositeStep" are deactivated. By entering "compositeStep" via its "resume" port, all steps within "compositeStep" are activated according to their setting before leaving the "compositeStep" via its "suspend" port.

## Modelica.StateGraph.Examples.ControlledTanks

Demonstrating the controller of a tank filling/emptying system



## Information

With this example the controller of a tank filling/emptying system is demonstrated. This example is from Dressler (2004), see [Literature](#). The basic operation is to fill and empty the two tanks:

1. Valve 1 is opened and tank 1 is filled.
2. When tank 1 reaches its fill level limit, valve 1 is closed.
3. After a waiting time, valve 2 is opened and the fluid flows from tank 1 into tank 2.
4. When tank 1 is empty, valve 2 is closed.
5. After a waiting time, valve 3 is opened and the fluid flows out of tank 2
6. When tank 3 is empty, valve 3 is closed

The above "normal" process can be influenced by three buttons:

- Button **start** starts the above process. When this button is pressed after a "stop" or "shut" operation, the process operation continues..
- Button **stop** stops the above process by closing all valves. Then, the controller waits for further input (either "start" or "shut" operation).
- Button **shut** is used to shutdown the process, by emptying at once both tanks. When this is achieved, the process goes back to its start configuration. Clicking on "start", restarts the process.

## Modelica.StateGraph.Examples.Utilities

Utility components for the examples

## Package Content

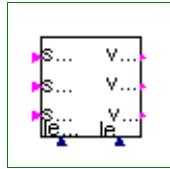
Name	Description
<a href="#">TankController</a>	Controller for tank system
<a href="#">MakeProduct</a>	
<a href="#">inflow</a>	Inflow connector (this is a copy from Isolde Dressler's master thesis project)
<a href="#">outflow</a>	Outflow connector (this is a copy from Isolde Dressler's master thesis project)
<a href="#">valve</a>	Simple valve model (this is a copy from Isolde Dressler's master thesis project)
<a href="#">Tank</a>	Simple tank model (this is a copy from Isolde Dressler's master thesis project)
<a href="#">Source</a>	Simple source model (this is a copy from Isolde Dressler's master thesis project)
<a href="#">CompositeStep</a>	
<a href="#">CompositeStep1</a>	
<a href="#">CompositeStep2</a>	

## 1796 Modelica.StateGraph.Examples.Utilities.TankController

---

### Modelica.StateGraph.Examples.Utilities.TankController

Controller for tank system



#### Parameters

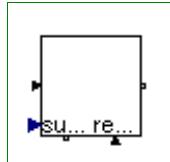
Name	Default	Description
limit	0.98	Limit level of tank 1
waitTime	3	Wait time

#### Connectors

Name	Description
start	
stop	
shut	
level1	
level2	
valve1	
valve2	
valve3	

---

### Modelica.StateGraph.Examples.Utilities.MakeProduct



#### Parameters

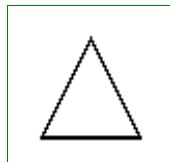
Name	Default	Description
limit	0.98	Limit level of tank 1
waitTime	3	Wait time
Exception connections		
nSuspend	1	Number of suspend ports
nResume	1	Number of resume ports

#### Connectors

Name	Description
inPort	
outPort	
suspend[nSuspend]	
resume[nResume]	
level1	

---

### Modelica.StateGraph.Examples.Utilities.Inflow



Inflow connector (this is a copy from Isolde Dressler's master thesis project)

#### Contents

Name	Description

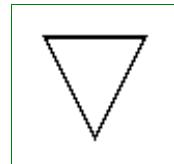
---

Fi	inflow [m <sup>3</sup> /s]
----	----------------------------

---

**Modelica.StateGraph.Examples.Utilities.outflow**

Outflow connector (this is a copy from Isolde Dressler's master thesis project)

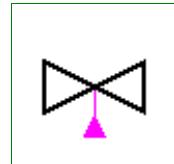
**Contents**

Name	Description
Fo	outflow [m <sup>3</sup> /s]
open	valve open

---

**Modelica.StateGraph.Examples.Utilities.valve**

Simple valve model (this is a copy from Isolde Dressler's master thesis project)

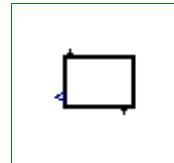
**Connectors**

Name	Description
valveControl	
inflow1	
outflow1	

---

**Modelica.StateGraph.Examples.Utilities.Tank**

Simple tank model (this is a copy from Isolde Dressler's master thesis project)

**Parameters**

Name	Default	Description
A	1	ground area of tank in m <sup>2</sup>
a	0.2	area of drain hole in m <sup>2</sup>
hmax	1	max height of tank in m

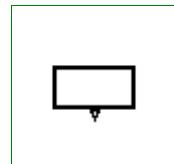
**Connectors**

Name	Description
levelSensor	
inflow1	
outflow1	

---

**Modelica.StateGraph.Examples.Utilities.Source**

Simple source model (this is a copy from Isolde Dressler's master thesis project)

**Parameters**

Name	Default	Description

## 1798 Modelica.StateGraph.Examples.Utilities.Source

---

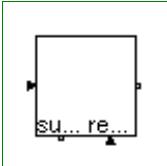
maxflow	1	maximal flow out of source
---------	---	----------------------------

### Connectors

Name	Description
outflow1	

---

## Modelica.StateGraph.Examples.Utilities.CompositeStep



### Parameters

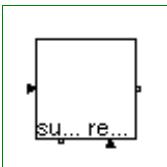
Name	Default	Description
Exception connections		
nSuspend	1	Number of suspend ports
nResume	1	Number of resume ports

### Connectors

Name	Description
inPort	
outPort	
suspend[nSuspend]	
resume[nResume]	

---

## Modelica.StateGraph.Examples.Utilities.CompositeStep1



### Parameters

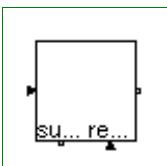
Name	Default	Description
Exception connections		
nSuspend	1	Number of suspend ports
nResume	1	Number of resume ports

### Connectors

Name	Description
inPort	
outPort	
suspend[nSuspend]	
resume[nResume]	

---

## Modelica.StateGraph.Examples.Utilities.CompositeStep2



### Parameters

Name	Default	Description
waitTime	2	waiting time in this composite step

Exception connections		
nSuspend	1	Number of suspend ports
nResume	1	Number of resume ports

## Connectors

Name	Description
inPort	
outPort	
suspend[nSuspend]	
resume[nResume]	

---

## Modelica.StateGraph.Interfaces

### Connectors and partial models

## Information

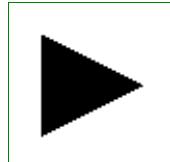
### Package Content

Name	Description
 Step_in	Input port of a step
 Step_out	Output port of a step
 Transition_in	Input port of a transition
 Transition_out	Output port of a transition
 CompositeStep_resume	Input port of a step (used for resume connector of a CompositeStep)
 CompositeStep_suspend	Output port of a step (used for suspend connector of a CompositeStep)
 CompositeStepStatePort_in	Communication port between a CompositeStep and the ordinary steps within the CompositeStep (suspend/resume are inputs)
 CompositeStepStatePort_out	Communication port between a CompositeStep and the ordinary steps within the CompositeStep (suspend/resume are outputs)
 PartialStep	Partial step with one input and one output transition port
 PartialTransition	Partial transition with input and output connections
 PartialStateGraphIcon	Icon for a StateGraph object
CompositeStepState	Communication channel between CompositeSteps and steps in the CompositeStep

---

### Modelica.StateGraph.Interfaces.Step\_in

#### Input port of a step



## 1800 Modelica.StateGraph.Interfaces.Step\_in

---

### Information

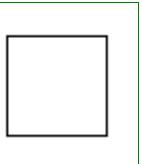
### Contents

Name	Description
occupied	true, if step is active
set	true, if transition fires and step is activated

---

## Modelica.StateGraph.Interfaces.Step\_out

Output port of a step



### Information

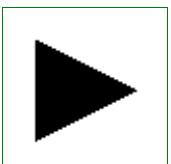
### Contents

Name	Description
available	true, if step is active
reset	true, if transition fires and step is deactivated

---

## Modelica.StateGraph.Interfaces.Transition\_in

Input port of a transition



### Information

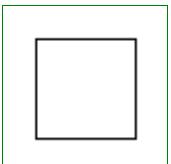
### Contents

Name	Description
available	true, if step connected to the transition input is active
reset	true, if transition fires and the step connected to the transition input is deactivated

---

## Modelica.StateGraph.Interfaces.Transition\_out

Output port of a transition



### Information

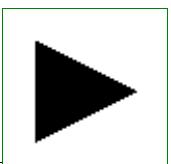
### Contents

Name	Description
occupied	true, if step connected to the transition output is active
set	true, if transition fires and step connected to the transition output becomes active

---

## Modelica.StateGraph.Interfaces.CompositeStep\_resume

Input port of a step (used for resume connector of a CompositeStep)

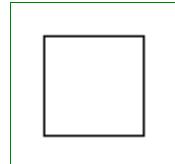


**Information****Contents**

Name	Description
occupied	true, if step is active
set	true, if transition fires and step is activated

**Modelica.StateGraph.Interfaces.CompositeStep\_suspend**

Output port of a step (used for suspend connector of a CompositeStep)

**Information****Contents**

Name	Description
available	true, if step is active
reset	true, if transition fires and step is deactivated

**Modelica.StateGraph.Interfaces.CompositeStepStatePort\_in**

Communication port between a CompositeStep and the ordinary steps within the CompositeStep (suspend/resume are inputs)

**Information****Contents**

Name	Description
suspend	= true, if suspend transition of CompositeStep fires
resume	= true, if resume transition of CompositeStep fires
activeSteps	Number of active steps in the CompositeStep

**Modelica.StateGraph.Interfaces.CompositeStepStatePort\_out**

Communication port between a CompositeStep and the ordinary steps within the CompositeStep (suspend/resume are outputs)

**Information****Contents**

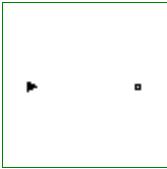
Name	Description
suspend	= true, if suspend transition of CompositeStep fires
resume	= true, if resume transition of CompositeStep fires
activeSteps	Number of active steps in the CompositeStep

---

## 1802 Modelica.StateGraph.Interfaces.PartialStep

### Modelica.StateGraph.Interfaces.PartialStep

Partial step with one input and one output transition port



#### Information

#### Parameters

Name	Default	Description
nIn	1	Number of input connections
nOut	1	Number of output connections

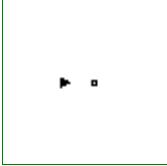
#### Connectors

Name	Description
inPort[nIn]	Vector of step input connectors
outPort[nOut]	Vector of step output connectors

---

### Modelica.StateGraph.Interfaces.PartialTransition

Partial transition with input and output connections



#### Information

#### Parameters

Name	Default	Description
Timer		
enableTimer	false	= true, if timer is enabled
waitTime	0	Wait time before transition fires [s]

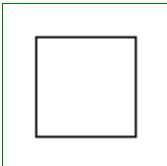
#### Connectors

Name	Description
inPort	Vector of transition input connectors
outPort	Vector of transition output connectors

---

### Modelica.StateGraph.Interfaces.PartialStateGraphIcon

Icon for a StateGraph object



#### Information

---

### Modelica.StateGraph.Interfaces.CompositeStepState

Communication channel between CompositeSteps and steps in the CompositeStep

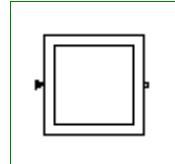
## Information

## Connectors

Name	Description
subgraphStatePort	

## Modelica.StateGraph.InitialStep

Initial step (= step that is active when simulation starts)



## Information

## Parameters

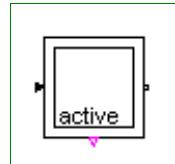
Name	Default	Description
nIn	1	Number of input connections
nOut	1	Number of output connections
localActive	active	= true if step is active, otherwise the step is not active

## Connectors

Name	Description
inPort[nIn]	Vector of step input connectors
outPort[nOut]	Vector of step output connectors

## Modelica.StateGraph.InitialStepWithSignal

Initial step (= step that is active when simulation starts). Connector 'active' is true when the step is active



## Information

## Parameters

Name	Default	Description
nIn	1	Number of input connections
nOut	1	Number of output connections
localActive	active	= true if step is active, otherwise the step is not active

## Connectors

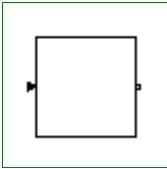
Name	Description
inPort[nIn]	Vector of step input connectors
outPort[nOut]	Vector of step output connectors
active	

## 1804 Modelica.StateGraph.Step

---

### Modelica.StateGraph.Step

Ordinary step (= step that is not active when simulation starts)



#### Information

#### Parameters

Name	Default	Description
nIn	1	Number of input connections
nOut	1	Number of output connections
localActive	active	= true if step is active, otherwise the step is not active

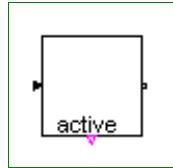
#### Connectors

Name	Description
inPort[nIn]	Vector of step input connectors
outPort[nOut]	Vector of step output connectors

---

### Modelica.StateGraph.StepWithSignal

Ordinary step (= step that is not active when simulation starts). Connector 'active' is true when the step is active



#### Information

#### Parameters

Name	Default	Description
nIn	1	Number of input connections
nOut	1	Number of output connections
localActive	active	= true if step is active, otherwise the step is not active

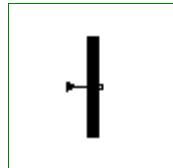
#### Connectors

Name	Description
inPort[nIn]	Vector of step input connectors
outPort[nOut]	Vector of step output connectors
active	

---

### Modelica.StateGraph.Transition

Transition where the fire condition is set by a modification of variable condition



#### Information

#### Parameters

Name	Default	Description
Fire condition		

condition	true	= true, if transition may fire (time varying expression)
<b>Timer</b>		
enableTimer	false	= true, if timer is enabled
waitTime	0	Wait time before transition fires [s]

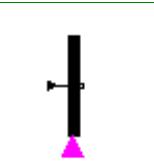
## Connectors

Name	Description
inPort	Vector of transition input connectors
outPort	Vector of transition output connectors

---

## Modelica.StateGraph.TransitionWithSignal

Transition where the fire condition is set by a Boolean input signal



## Information

## Parameters

Name	Default	Description
<b>Timer</b>		
enableTimer	false	= true, if timer is enabled
waitTime	0	Wait time before transition fires [s]

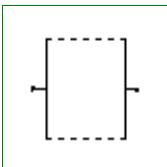
## Connectors

Name	Description
condition	
inPort	Vector of transition input connectors
outPort	Vector of transition output connectors

---

## Modelica.StateGraph.Alternative

Alternative splitting of execution path (use component between two steps)



## Information

## Parameters

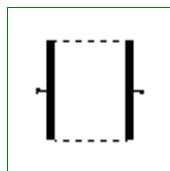
Name	Default	Description
nBranches	2	Number of alternative branches

## Connectors

Name	Description
inPort	
outPort	
join[nBranches]	
split[nBranches]	

**Modelica.StateGraph.Parallel**

Parallel splitting of execution path (use component between two transitions)

**Information****Parameters**

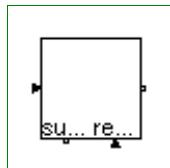
Name	Default	Description
nBranches	2	Number of parallel branches that are executed in parallel

**Connectors**

Name	Description
inPort	
outPort	
join[nBranches]	
split[nBranches]	

**Modelica.StateGraph.PartialCompositeStep**

Superclass of a subgraph, i.e., a composite step that has internally a StateGraph

**Information****Parameters**

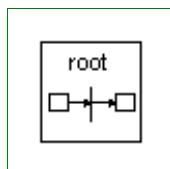
Name	Default	Description
Exception connections		
nSuspend	1	Number of suspend ports
nResume	1	Number of resume ports

**Connectors**

Name	Description
inPort	
outPort	
suspend[nSuspend]	
resume[nResume]	

**Modelica.StateGraph.StateGraphRoot**

Root of a StateGraph (has to be present on the highest level of a StateGraph)

**Information**

On the highest level of a StateGraph, an instance of StateGraphRoot has to be present. If it is not within in a model, it is automatically included by a Modelica translator due to an appropriate annotation. Practically, this means that it need not be present in a StateGraph model.

The StateGraphRoot object is needed, since all Step objects have an "outer" reference to communicate with the "nearest" CompositeStep (which inherits from PartialCompositeStep), especially to abort a CompositeStep via the "suspend" port. Even if no "CompositeStep" is present, on highest level a corresponding "inner" definition is needed and is provided by the StateGraphRoot object.

## Connectors

Name	Description
subgraphStatePort	

---

## Modelica.StateGraph.Temporary

Components that will be provided by other libraries in the future

## Information

This library is just temporarily present. The components of this library will be present in the future in the Modelica standard library (with the new block connectors) and in the UserInteraction library that is currently under development.

## Package Content

Name	Description
 SetRealParameter	Define Real parameter (GUI not yet satisfactory)
 anyTrue	Returns true, if at least one element of the Boolean input vector is true
 allTrue	Returns true, if all elements of the Boolean input vector are true
 RadioButton	Button that sets its output to true when pressed and is reset when an element of 'reset' becomes true
 NumericValue	Show value of Real input signal dynamically
 IndicatorLamp	Dynamically show Boolean input signal (false/true = white/green color)

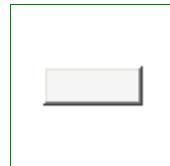
## Types and constants

```
type SetRealParameter = Real "Define Real parameter (GUI not yet satisfactory);
```

---

## Modelica.StateGraph.Temporary.SetRealParameter

Define Real parameter (GUI not yet satisfactory)



## Information

This is an **experimental component** to define a **Real parameter** in the **diagram layer**. The idea is to drag the icon from the package browser into the diagram layer. Then a window pops up in which the properties of this parameter can be defined (such as the default value). The name and default value of the parameter are displayed in the icon of this component. Whenever clicking on it, the dialog to change parameter settings pops-up.

In Dymola, the described property is not fully available. Currently, when dragging this component in the diagram layer, a dialog pops up in which the properties of the parameter can be defined. However, afterwards, the parameter is not visible in the diagram layer. Making it visible requires to go into the text layer and add an annotation with the component size, resulting for example in:

## 1808 Modelica.StateGraph.Temporary.SetRealParameter

---

```
parameter StateGraph.SetRealParameter name = 2
annotation(extent=[-10,-10; 10,10]);
```

This change makes the parameter icon visible in the diagram layer. However, clicking on this icon has no effect. Changing parameter properties, such as the default value, still requires to go in to the text layer.

---

### Modelica.StateGraph.Temporary.anyTrue

Returns true, if at least one element of the Boolean input vector is true

#### Information

#### Inputs

Name	Default	Description
b[:]		

#### Outputs

Name	Description
result	

---

### Modelica.StateGraph.Temporary.allTrue

Returns true, if all elements of the Boolean input vector are true

#### Information

#### Inputs

Name	Default	Description
b[:]		

#### Outputs

Name	Description
result	

---

### Modelica.StateGraph.Temporary.RadioButton

Button that sets its output to true when pressed and is reset when an element of 'reset' becomes true

#### Information

#### Parameters

Name	Default	Description
buttonTimeTable[:]		Time instants where button is pressend and released [s]

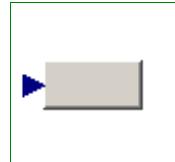
Time varying expressions		
reset[:]	{false}	Reset button to false, if an element of reset becomes true

## Connectors

Name	Description
on	

## Modelica.StateGraph.Temporary.NumericValue

Show value of Real input signal dynamically



## Information

## Parameters

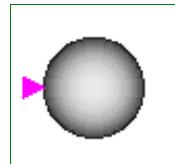
Name	Default	Description
precision	3	Number of significant digits to be shown
hideConnector	false	= true, if connector is not shown in the dynamic object diagram
Value		Real value to be shown in icon

## Connectors

Name	Description
Value	Real value to be shown in icon

## Modelica.StateGraph.Temporary.IndicatorLamp

Dynamically show Boolean input signal (false/true = white/green color)



## Information

## Connectors

Name	Description
u	

## Modelica.Thermal

Library of thermal system components to model heat transfer and simple thermo-fluid pipe flow

## Information

This package contains libraries to model heat transfer and fluid heat flow.

## Package Content

Name	Description
 FluidHeatFlow	Library of simple components for 1-dimensional incompressible thermo-fluid flow models

 HeatTransfer	Library of 1-dimensional heat transfer with lumped elements
--	---

## Modelica.Thermal.FluidHeatFlow

Library of simple components for 1-dimensional incompressible thermo-fluid flow models

### Information

This package contains very simple-to-use components to model coolant flows as needed to simulate cooling e.g. of electric machines:

- Components: components like different types of pipe models
- Examples: some test examples
- Interfaces: definition of connectors and partial models (containing the core thermodynamic equations)
- Media: definition of media properties
- Sensors: various sensors for pressure, temperature, volume and enthalpy flow
- Sources: various flow sources

### Variables used in connectors:

- Pressure p
- flow MassFlowRate m\_flow
- SpecificEnthalpy h
- flow EnthalpyFlowRate H\_flow

EnthalpyFlowRate means the  $\text{Enthalpy} = \text{cp}_{\text{constant}} * \text{m} * \text{T}$  that is carried by the medium's flow.

### Limitations and assumptions:

- Splitting and mixing of coolant flows (media with the same cp) is possible.
- Reversing the direction of flow is possible.
- The medium is considered to be incompressible.
- No mixtures of media is taken into consideration.
- The medium may not change its phase.
- Medium properties are kept constant.
- Pressure changes are only due to pressure drop and geodetic height difference  $\rho * g * h$  (if  $h > 0$ ).
- A user-defined part (0..1) of the friction losses ( $V_{\text{flow}} * dp$ ) are fed to the medium.
- **Note:** Connected flowPorts have the same temperature (mixing temperature)!  
Since mixing may occur, the outlet temperature may be different from the connector's temperature.  
Outlet temperature is defined by variable T of the corresponding component.

### Further development:

- Additional components like tanks (if needed)

### Main Authors:

Anton Haumer  
 Technical Consulting & Electrical Engineering  
 A-3423 St.Andrae-Woerdern, Austria  
 email: [a.haumer@haumer.at](mailto:a.haumer@haumer.at)

Dr.Christian Kral & Markus Plainer  
 Österreichisches Forschungs- und Prüfzentrum Arsenal Ges.m.b.H.  
 arsenal research  
 Giefinggasse 2  
 A-1210 Vienna, Austria

Copyright © 1998-2007, Modelica Association, Anton Haumer and arsenal research.

*This Modelica package is free software; it can be redistributed and/or modified under the terms of the Modelica license, see the license conditions and the accompanying disclaimer [here](#).*

## Package Content

Name	Description
 Examples	Examples that demonstrate the usage of the FluidHeatFlow components
 Components	Basic components (pipes, valves)
 Interfaces	Connectors and partial models
 Media	Medium properties
 Sensors	Ideal sensors to measure port properties
 Sources	Ideal fluid sources, e.g., ambient, volume flow

---

## Modelica.Thermal.FluidHeatFlow.Examples

### Examples that demonstrate the usage of the FluidHeatFlow components

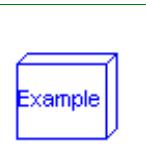
#### Information

This package contains test examples:

- 1.SimpleCooling: heat is dissipated through a media flow
- 2.ParallelCooling: two heat sources dissipate through merged media flows
- 3.IndirectCooling: heat is dissipated through two cooling cycles
- 4.PumpAndValve: demonstrates usage of an IdealPump and a Valve
- 5.PumpDropOut: demonstrates shutdown and restart of a pump
- 6.ParallelPumpDropOut: demonstrates shutdown and restart of a pump in a parallel circuit
- 7.OneMass: cooling of a mass (thermal capacity) by a coolant flow
- 8.TwoMass: cooling of two masses (thermal capacities) by two parallel coolant flows

## Package Content

Name	Description
 SimpleCooling	Example: simple cooling circuit
 ParallelCooling	Example: cooling circuit with parallel branches
 IndirectCooling	Example: indirect cooling circuit
 PumpAndValve	Example: cooling circuit with pump and valve
 PumpDropOut	Example: cooling circuit with drop out of pump
 ParallelPumpDropOut	Example: cooling circuit with parallel branches and drop out of pump
 OneMass	Example: cooling of one hot mass
 TwoMass	Example: cooling of two hot masses
 Utilities	Utility models for examples

**Modelica.Thermal.FluidHeatFlow.Examples.SimpleCooling****Example:** simple cooling circuit**Information**

1st test example: SimpleCooling

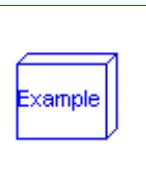
A prescribed heat source dissipates its heat through a thermal conductor to a coolant flow. The coolant flow is taken from an ambient and driven by a pump with prescribed mass flow.

**Results:**

<b>output</b>	<b>explanation</b>	<b>formula</b>	<b>actual steady-state value</b>
dTSource	Source over Ambient	$dt_{Coolant} + dt_{ToPipe}$	20 K
dTtoPipe	Source over Coolant	Losses / ThermalConductor.G	10 K
dTCoolant	Coolant's temperature increase	Losses * cp * massFlow	10 K

**Parameters**

<b>Name</b>	<b>Default</b>	<b>Description</b>
medium	FluidHeatFlow.Media.Medium()	Cooling medium
TAmb	20	Ambient temperature [degC]

**Modelica.Thermal.FluidHeatFlow.Examples.ParallelCooling****Example:** coolig circuit with parallel branches**Information**

2nd test example: ParallelCooling

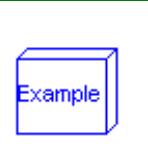
Two prescribed heat sources dissipate their heat through thermal conductors to coolant flows. The coolant flow is taken from an ambient and driven by a pump with prescribed mass flow, then splitted into two coolant flows connected to the two heat sources, and afterwards merged. Splitting of coolant flows is determined by pressure drop characteristic of the two pipes.

**Results:**

<b>output</b>	<b>explanation</b>	<b>formula</b>	<b>actual steady-state value</b>
dTSource1	Source1 over Ambient	$dt_{Coolant1} + dt_{ToPipe1}$	15 K
dTtoPipe1	Source1 over Coolant1	Losses1 / ThermalConductor1.G	5 K
dTCoolant1	Coolant's temperature increase	Losses * cp * totalMassFlow/2	10 K
dTSource2	Source2 over Ambient	$dt_{Coolant2} + dt_{ToPipe2}$	30 K
dTtoPipe2	Source2 over Coolant2	Losses2 / ThermalConductor2.G	10 K
dTCoolant2	Coolant's temperature increase	Losses * cp * totalMassFlow/2	20 K
dTmixedCoolant	mixed Coolant's temperature increase	$(dt_{Coolant1} + dt_{Coolant2})/2$	15 K

**Parameters**

<b>Name</b>	<b>Default</b>	<b>Description</b>
medium	FluidHeatFlow.Media.Medium()	Cooling medium
TAmb	20	Ambient temperature [degC]

**Modelica.Thermal.FluidHeatFlow.Examples.IndirectCooling**

**Example:** indirect cooling circuit

**Information**

3rd test example: IndirectCooling

A prescribed heat source dissipates its heat through a thermal conductor to the inner coolant cycle. It is necessary to define the pressure level of the inner coolant cycle. The inner coolant cycle is coupled to the outer coolant flow through a thermal conductor.

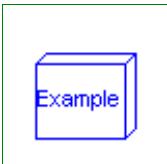
Inner coolant's temperature rise near the source is the same as temperature drop near the cooler.

**Results:**

output	explanation	formula	actual steady-state value
dTSource	Source over Ambient	$dt_{outerCoolant} + dt_{Cooler} + dt_{innerCoolant} + dt_{ToPipe}$	40 K
dTtoPipe	Source over inner Coolant	Losses / ThermalConductor.G	10 K
dTinnerColant	inner Coolant's temperature increase	Losses * cp * innerMassFlow	10 K
dTCooler	Cooler's temperature rise between inner and outer pipes	Losses * (innerGc + outerGc)	10 K
dTouterColant	outer Coolant's temperature increase	Losses * cp * outerMassFlow	10 K

**Parameters**

Name	Default	Description
outerMedium	FluidHeatFlow.Media.Medium()	Outer medium
innerMedium	FluidHeatFlow.Media.Medium()	Inner medium
TAmb	20	Ambient temperature [degC]

**Modelica.Thermal.FluidHeatFlow.Examples.PumpAndValve**

**Example:** cooling circuit with pump and valve

**Information**

4th test example: PumpAndValve

The pump is running with half speed for 0.4 s, afterwards with full speed (using a ramp of 0.1 s). The valve is half open for 0.9 s, afterwards full open (using a ramp of 0.1 s).

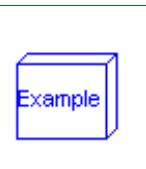
You may try to

- drive the pump with variable speed and let the valve full open to regulate the volume flow rate of coolant
- drive the pump with constant speed and throttle the valve to regulate the volume flow rate of coolant

**Parameters**

Name	Default	Description
medium	FluidHeatFlow.Media.Medium()	Cooling medium
TAmb	20	Ambient temperature [degC]

---

**Modelica.Thermal.FluidHeatFlow.Examples.PumpDropOut****Example:** cooling circuit with drop out of pump**Information**

5th test example: PumpDropOut

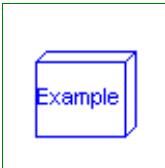
Same as 1st test example, but with a drop out of the pump:

The pump is running for 0.2 s, then shut down (using a ramp of 0.2 s) for 0.2 s, then started again (using a ramp of 0.2 s).

**Parameters**

Name	Default	Description
medium	FluidHeatFlow.Media.Medium()	Cooling medium
TAmb	20	Ambient temperature [degC]

---

**Modelica.Thermal.FluidHeatFlow.Examples.ParallelPumpDropOut****Example:** cooling circuit with parallel branches and drop out of pump**Information**

6th test example: ParallelPumpDropOut

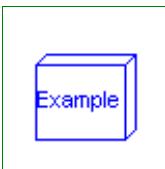
Same as 2nd test example, but with a drop out of the pump:

The pump is running for 0.2 s, then shut down (using a ramp of 0.2 s) for 0.2 s, then started again (using a ramp of 0.2 s).

**Parameters**

Name	Default	Description
medium	FluidHeatFlow.Media.Medium()	Cooling medium
TAmb	20	Ambient temperature [degC]

---

**Modelica.Thermal.FluidHeatFlow.Examples.OneMass****Example:** cooling of one hot mass**Information**

7th test example: OneMass

A thermal capacity is coupled with a coolant flow. Different initial temperatures of thermal capacity and pipe's coolant get ambient's temperature, the time behaviour depending on coolant flow.

**Parameters**

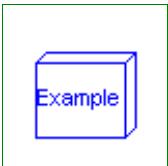
Name	Default	Description
medium	FluidHeatFlow.Media.Medium()	Cooling medium
TAmb	20	Ambient temperature [degC]

---

TMass	40	Initial temperature of mass [degC]
-------	----	------------------------------------

## Modelica.Thermal.FluidHeatFlow.Examples.TwoMass

Example: cooling of two hot masses



### Information

8th test example: TwoMass

Two thermal capacities are coupled with two parallel coolant flow. Different initial temperatures of thermal capacities and pipe's coolants get ambient's temperature, the time behaviour depending on coolant flow.

### Parameters

Name	Default	Description
medium	FluidHeatFlow.Media.Medium()	Cooling medium
TAmb	20	Ambient temperature [degC]
TMass1	40	Initial temperature of mass1 [degC]
TMass2	60	Initial temperature of mass2 [degC]

## Modelica.Thermal.FluidHeatFlow.Examples.Utilities

Utility models for examples

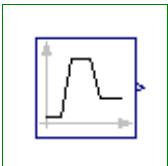
### Information

This package contains utility components used for the test examples.

### Package Content

Name	Description
DoubleRamp	Ramp going up and down

## Modelica.Thermal.FluidHeatFlow.Examples.Utilities.DoubleRamp



Ramp going up and down

### Information

Block generating the sum of two ramps.

### Parameters

Name	Default	Description
offset	1	Offset of ramps
startTime	0.2	StartTime of 1st ramp [s]
interval	0.2	Interval between end of 1st and beginning of 2nd ramp [s]
Ramp 1		
height_1	-1	Height of ramp

## 1816 Modelica.Thermal.FluidHeatFlow.Examples.Utilities.DoubleRamp

---

duration_1	0.2	Duration of ramp [s]
Ramp 2		
height_2	1	Height of ramp
duration_2	0.2	Duration of ramp [s]

### Connectors

Name	Description
y	Connector of Real output signal

---

## Modelica.Thermal.FluidHeatFlow.Components

### Basic components (pipes, valves)

### Information

This package contains components:

- pipe without heat exchange
- pipe with heat exchange
- valve (simple controlled valve)

Pressure drop is taken from partial model SimpleFriction.

Thermodynamic equations are defined in partial models (package Partials).

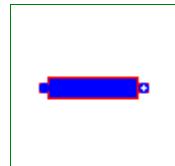
### Package Content

Name	Description
 IsolatedPipe	Pipe without heat exchange
 HeatedPipe	Pipe with heat exchange
 Valve	Simple valve

---

## Modelica.Thermal.FluidHeatFlow.Components.IsolatedPipe

### Pipe without heat exchange



### Information

Pipe without heat exchange.

Thermodynamic equations are defined by Partials.TwoPortMass( $Q_{\text{flow}} = 0$ ).

**Note:** Setting parameter m (mass of medium within pipe) to zero leads to neglection of temperature transient  $cv \cdot m \cdot \text{der}(T)$ .

### Parameters

Name	Default	Description
medium	FluidHeatFlow.Media.Medium()	Medium in the component
m	1	Mass of medium [kg]
T0	Modelica.SIunits.Conversions...	Initial temperature of medium [K]
h_g	0	Geodetic height (height difference from flowPort_a to flowPort_b) [m]

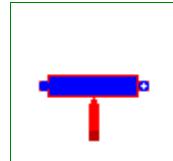
SimpleFriction		
V_flowLaminar	0.1	Laminar volume flow [m <sup>3</sup> /s]
dpLaminar	0.1	Laminar pressure drop [Pa]
V_flowNominal	1	Nominal volume flow [m <sup>3</sup> /s]
dpNominal	1	Nominal pressure drop [Pa]
frictionLoss	0	Part of friction losses fed to medium

## Connectors

Name	Description
flowPort_a	
flowPort_b	

## Modelica.Thermal.FluidHeatFlow.Components.HeatedPipe

Pipe with heat exchange



## Information

Pipe with heat exchange.

Thermodynamic equations are defined by Partials.TwoPort.

Q\_flow is defined by heatPort.Q\_flow.

**Note:** Setting parameter m (mass of medium within pipe) to zero leads to neglection of temperature transient  $cv*m*der(T)$ .

**Note:** Injecting heat into a pipe with zero massflow causes temperature rise defined by storing heat in medium's mass.

## Parameters

Name	Default	Description
medium	FluidHeatFlow.Media.Medium()	Medium in the component
m	1	Mass of medium [kg]
T0	Modelica.Slunits.Conversions...	Initial temperature of medium [K]
tapT	1	Defines temperature of heatPort between inlet and outlet temperature
h_g	0	Geodetic height (height difference from flowPort_a to flowPort_b) [m]
SimpleFriction		
V_flowLaminar	0.1	Laminar volume flow [m <sup>3</sup> /s]
dpLaminar	0.1	Laminar pressure drop [Pa]
V_flowNominal	1	Nominal volume flow [m <sup>3</sup> /s]
dpNominal	1	Nominal pressure drop [Pa]
frictionLoss	0	Part of friction losses fed to medium

## Connectors

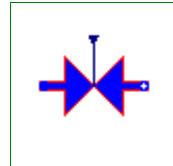
Name	Description
flowPort_a	

## 1818 Modelica.Thermal.FluidHeatFlow.Components.HeatedPipe

flowPort_b	
heatPort	

## Modelica.Thermal.FluidHeatFlow.Components.Valve

### Simple valve



### Information

Simple controlled valve.

Standard characteristic  $Kv=f(y)$  is given at standard conditions ( $dp_0$ ,  $\rho_0$ ),

- either linear :  $Kv/Kv1 = Kv0/Kv1 + (1-Kv0/Kv1) * y/Y1$
- or exponential:  $Kv/Kv1 = Kv0/Kv1 * \exp[\ln(Kv1/Kv0) * y/Y1]$

where:

- $Kv0$  ... min. flow @  $y = 0$
- $Y1$  .... max. valve opening
- $Kv1$  ... max. flow @  $y = Y1$

Flow resistance under real conditions is calculated by

$$V_{\text{flow}}^{**2} * \rho / dp = Kv(y)^{**2} * \rho_0 / dp_0$$

### Parameters

Name	Default	Description
medium	FluidHeatFlow.Media.Medium()	Medium in the component
m	0	Mass of medium [kg]
T0	Modelica.Slunits.Conversions...	Initial temperature of medium [K]
frictionLoss	0	Part of friction losses fed to medium
Standard characteristic		
LinearCharacteristic	true	Type of characteristic
y1	1	Max. valve opening
Kv1	1	Max. flow @ $y = y1$ [m <sup>3</sup> /s]
kv0	0.01	Leakage flow / max.flow @ $y = 0$
dp0	1	Standard pressure drop [Pa]
rho0	10	Standard medium's density [kg/m <sup>3</sup> ]

### Connectors

Name	Description
flowPort_a	
flowPort_b	
y	

## Modelica.Thermal.FluidHeatFlow.Interfaces

### Connectors and partial models

## Information

This package contains connectors and partial models:

- FlowPort: basic definition of the connector.
- FlowPort\_a & FlowPort\_b: same as FlowPort with different icons to differentiate direction of flow
- package Partials (defining basic thermodynamic equations)

## Package Content

Name	Description
 FlowPort	conector flow port
 FlowPort_a	Filled flow port (used upstream)
 FlowPort_b	Hollow flow port (used downstream)
 Partials	Partial models

## Modelica.Thermal.FluidHeatFlow.Interfaces.FlowPort

### conector flow port

#### Information

Basic definition of the connector.

#### Variables:

- Pressure p
- flow MassFlowRate m\_flow
- Specific Enthalpy h
- flow EnthalpyFlowRate H\_flow

If ports with different media are connected, the simulation is asserted due to the check of parameter.

## Parameters

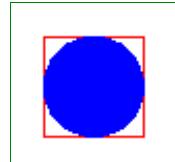
Name	Default	Description
medium		Medium in the connector

## Contents

Name	Description
medium	Medium in the connector
p	[Pa]
m_flow	[kg/s]
h	[J/kg]
H_flow	[W]

## Modelica.Thermal.FluidHeatFlow.Interfaces.FlowPort\_a

### Filled flow port (used upstream)



## 1820 Modelica.Thermal.FluidHeatFlow.Interfaces.FlowPort\_a

---

### Information

Same as FlowPort, but icon allows to differentiate direction of flow.

### Parameters

Name	Default	Description
medium		Medium in the connector

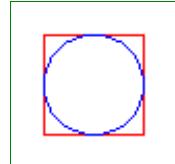
### Contents

Name	Description
medium	Medium in the connector
p	[Pa]
m_flow	[kg/s]
h	[J/kg]
H_flow	[W]

---

## Modelica.Thermal.FluidHeatFlow.Interfaces.FlowPort\_b

Hollow flow port (used downstream)



### Information

Same as FlowPort, but icon allows to differentiate direction of flow.

### Parameters

Name	Default	Description
medium		Medium in the connector

### Contents

Name	Description
medium	Medium in the connector
p	[Pa]
m_flow	[kg/s]
h	[J/kg]
H_flow	[W]

---

## Modelica.Thermal.FluidHeatFlow.Interfaces.Partial

Partial models

### Information

This package contains partial models, defining in a very compact way the basic thermodynamic equations used by the different components.

Main Authors:

Anton Haumer

Technical Consulting & Electrical Engineering  
A-3423 St.Andrae-Woerdern, Austria  
email: [a.haumer@haumer.at](mailto:a.haumer@haumer.at)

Dr.Christian Kral & Markus Plainer  
Österreichisches Forschungs- und Prüfzentrum Arsenal Ges.m.b.H.  
arsenal research  
Giefinggasse 2  
A-1210 Vienna, Austria

Copyright © 1998-2007, Modelica Association, Anton Haumer and arsenal research.

*The Modelica package is free software; it can be redistributed and/or modified under the terms of the Modelica license, see the license conditions and the accompanying disclaimer [here](#).*

## Package Content

Name	Description
SimpleFriction	Simple friction model
TwoPort	Partial model of two port
Ambient	Partial model of ambient
AbsoluteSensor	Partial model of absolute sensor
RelativeSensor	Partial model of relative sensor
FlowSensor	Partial model of flow sensor

---

## Modelica.Thermal.FluidHeatFlow.Interfaces.Partial.SimpleFriction

### Simple friction model

#### Information

Definition of relationship between pressure drop and volume flow rate:

-V\_flowLaminar < VolumeFlow < +V\_flowLaminar: laminar i.e. linear dependency of pressure drop on volume flow.

-V\_flowLaminar > VolumeFlow or VolumeFlow < +V\_flowLaminar: turbulent i.e. quadratic dependency of pressure drop on volume flow.

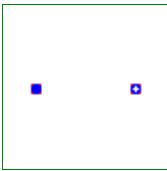
Linear and quadratic dependency are coupled smoothly at V\_flowLaminar / dpLaminar.

Quadratic dependency is defined by nominal volume flow and pressure drop (V\_flowNominal / dpNominal).

See also sketch at diagram layer.

#### Parameters

Name	Default	Description
<b>SimpleFriction</b>		
V_flowLaminar	0.1	Laminar volume flow [m <sup>3</sup> /s]
dpLaminar	0.1	Laminar pressure drop [Pa]
V_flowNominal	1	Nominal volume flow [m <sup>3</sup> /s]
dpNominal	1	Nominal pressure drop [Pa]
frictionLoss	0	Part of friction losses fed to medium

**Modelica.Thermal.FluidHeatFlow.Interfaces.Partials.TwoPort****Partial model of two port****Information**

Partial model with two flowPorts.

Possible heat exchange with the ambient is defined by  $Q_{\text{flow}}$ ; setting this = 0 means no energy exchange. Setting parameter  $m$  (mass of medium within pipe) to zero leads to neglection of temperature transient  $cv*m*\text{der}(T)$ .

Mixing rule is applied.

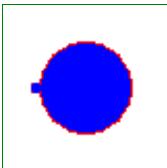
Parameter  $0 < \text{tapT} < 1$  defines temperature of heatPort between medium's inlet and outlet temperature.

**Parameters**

Name	Default	Description
medium	FluidHeatFlow.Media.Medium()	Medium in the component
$m$	1	Mass of medium [kg]
$T_0$	Modelica.SIunits.Conversions...	Initial temperature of medium [K]
$\text{tapT}$	1	Defines temperature of heatPort between inlet and outlet temperature

**Connectors**

Name	Description
flowPort_a	
flowPort_b	

**Modelica.Thermal.FluidHeatFlow.Interfaces.Partials.Ambient****Partial model of ambient****Information**

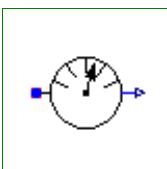
Partial model of (Infinite) ambient, defines pressure and temperature.

**Parameters**

Name	Default	Description
medium	FluidHeatFlow.Media.Medium()	Ambient's medium

**Connectors**

Name	Description
flowPort	

**Modelica.Thermal.FluidHeatFlow.Interfaces.Partials.AbsoluteSensor****Partial model of absolute sensor****Information**

Partial model for an absolute sensor (pressure/temperature).

Pressure, mass flow, temperature and enthalpy flow of medium are not affected.

## Parameters

Name	Default	Description
medium	FluidHeatFlow.Media.Medium()	Sensor's medium

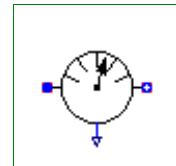
## Connectors

Name	Description
flowPort	
y	

---

## Modelica.Thermal.FluidHeatFlow.Interfaces.Partial.RelativeSensor

### Partial model of relative sensor



## Information

Partial model for a relative sensor (pressure drop/temperature difference).  
Pressure, mass flow, temperature and enthalpy flow of medium are not affected.

## Parameters

Name	Default	Description
medium	FluidHeatFlow.Media.Medium()	Sensor's medium

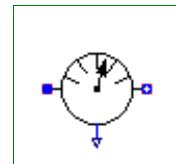
## Connectors

Name	Description
flowPort_a	
flowPort_b	
y	

---

## Modelica.Thermal.FluidHeatFlow.Interfaces.Partial.FlowSensor

### Partial model of flow sensor



## Information

Partial model for a flow sensor (mass flow/heat flow).  
Pressure, mass flow, temperature and enthalpy flow of medium are not affected, but mixing rule is applied.

## Parameters

Name	Default	Description
medium	FluidHeatFlow.Media.Medium()	Medium in the component

## Connectors

Name	Description
flowPort_a	

## 1824 Modelica.Thermal.FluidHeatFlow.Interfaces.Partials.FlowSensor

---

flowPort_b	
y	

---

## Modelica.Thermal.FluidHeatFlow.Media

### Medium properties

#### Information

This package contains definitions of medium properties.

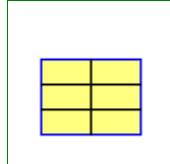
#### Package Content

Name	Description
Medium	Record containing media properties
Air_30degC	Medium: properties of air at 30 degC
Air_70degC	Medium: properties of air at 70 degC
Water	Medium: properties of water

---

## Modelica.Thermal.FluidHeatFlow.Media.Medium

### Record containing media properties



#### Information

Record containing (constant) medium properties.

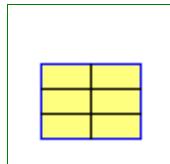
#### Parameters

Name	Default	Description
rho	1	Density [kg/m <sup>3</sup> ]
cp	1	Specific heat capacity at constant pressure [J/(kg.K)]
cv	1	Specific heat capacity at constant volume [J/(kg.K)]
lamda	1	Thermal conductivity [W/(m.K)]
nue	1	kinematic viscosity [m <sup>2</sup> /s]

---

## Modelica.Thermal.FluidHeatFlow.Media.Air\_30degC

### Medium: properties of air at 30 degC



#### Information

Medium: properties of air at 30 degC

#### Parameters

Name	Default	Description
rho	1.149	Density [kg/m <sup>3</sup> ]

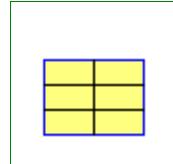
---

cp	1007	Specific heat capacity at constant pressure [J/(kg.K)]
cv	720	Specific heat capacity at constant volume [J/(kg.K)]
lamda	0.0264	Thermal conductivity [W/(m.K)]
nue	16.3E-6	kinematic viscosity [m <sup>2</sup> /s]

---

## Modelica.Thermal.FluidHeatFlow.Media.Air\_70degC

Medium: properties of air at 70 degC



### Information

Medium: properties of air at 70 degC

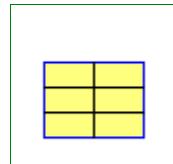
### Parameters

Name	Default	Description
rho	1.015	Density [kg/m <sup>3</sup> ]
cp	1010	Specific heat capacity at constant pressure [J/(kg.K)]
cv	723	Specific heat capacity at constant volume [J/(kg.K)]
lamda	0.0293	Thermal conductivity [W/(m.K)]
nue	20.3E-6	kinematic viscosity [m <sup>2</sup> /s]

---

## Modelica.Thermal.FluidHeatFlow.Media.Water

Medium: properties of water



### Information

Medium: properties of water

### Parameters

Name	Default	Description
rho	995.6	Density [kg/m <sup>3</sup> ]
cp	4177	Specific heat capacity at constant pressure [J/(kg.K)]
cv	4177	Specific heat capacity at constant volume [J/(kg.K)]
lamda	0.615	Thermal conductivity [W/(m.K)]
nue	0.8E-6	kinematic viscosity [m <sup>2</sup> /s]

---

## Modelica.Thermal.FluidHeatFlow.Sensors

Ideal sensors to measure port properties

### Information

This package contains sensors:

- pSensor: absolute pressure
- TSensor: absolute temperature (Kelvin)
- dpSensor: pressure drop between flowPort\_a and flowPort\_b

## 1826 Modelica.Thermal.FluidHeatFlow.Sensors

---

- dTSensor: temperature difference between flowPort\_a and flowPort\_b
- m\_flowSensor: measures mass flow rate
- V\_flowSensor: measures volume flow rate
- H\_flowSensor: measures enthalpy flow rate

Some of the sensors do not need access to medium properties for measuring, but it is necessary to define the medium in the connector (check of connections).

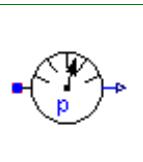
Thermodynamic equations are defined in partial models (package Interfaces.Partial). All sensors are considered massless, they do not change mass flow or enthalpy flow.

### Package Content

Name	Description
pSensor	Absolute pressure sensor
TSensor	Absolute temperature sensor
dpSensor	Pressure difference sensor
dTSensor	Temperature difference sensor
m_flowSensor	Mass flow sensor
V_flowSensor	Volume flow sensor
H_flowSensor	Enthalpy flow sensor

---

### Modelica.Thermal.FluidHeatFlow.Sensors.pSensor



#### Absolute pressure sensor

#### Information

pSensor measures the absolute pressure.

Thermodynamic equations are defined by Partials.AbsoluteSensor.

#### Parameters

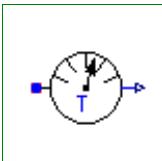
Name	Default	Description
medium	FluidHeatFlow.Media.Medium()	Sensor's medium

#### Connectors

Name	Description
flowPort	

---

### Modelica.Thermal.FluidHeatFlow.Sensors.TSensor



#### Absolute temperature sensor

#### Information

TSensor measures the absolute temperature (Kelvin).

Thermodynamic equations are defined by Partials.AbsoluteSensor.

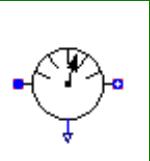
## Parameters

Name	Default	Description
medium	FluidHeatFlow.Media.Medium()	Sensor's medium

## Connectors

Name	Description
flowPort	

## Modelica.Thermal.FluidHeatFlow.Sensors.dpSensor



Pressure difference sensor

## Information

dpSensor measures the pressure drop between flowPort\_a and flowPort\_b. Thermodynamic equations are defined by Partials.RelativeSensor.

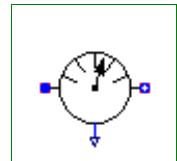
## Parameters

Name	Default	Description
medium	FluidHeatFlow.Media.Medium()	Sensor's medium

## Connectors

Name	Description
flowPort_a	
flowPort_b	

## Modelica.Thermal.FluidHeatFlow.Sensors.dTSensor



Temperature difference sensor

## Information

dTSensor measures the temperature difference between flowPort\_a and flowPort\_b. Thermodynamic equations are defined by Partials.RelativeSensor.

- Note:** Connected flowPorts have the same temperature (mixing temperature)! Since mixing may occur, the outlet temperature of a component may be different from the connector's temperature.  
Outlet temperature is defined by variable T of the corresponding component.

## Parameters

Name	Default	Description
medium	FluidHeatFlow.Media.Medium()	Sensor's medium

## Connectors

Name	Description
flowPort_a	

---

## 1828 Modelica.Thermal.FluidHeatFlow.Sensors.dTSensor

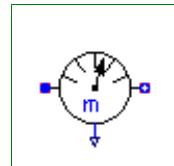
---

flowPort\_b

---

### Modelica.Thermal.FluidHeatFlow.Sensors.m\_flowSensor

Mass flow sensor



#### Information

m\_flowSensor measures the mass flow rate.

Thermodynamic equations are defined by Partials.FlowSensor.

#### Parameters

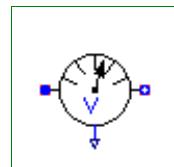
Name	Default	Description
medium	FluidHeatFlow.Media.Medium()	Medium in the component

#### Connectors

Name	Description
flowPort_a	
flowPort_b	

---

### Modelica.Thermal.FluidHeatFlow.Sensors.V\_flowSensor



Volume flow sensor

#### Information

V\_flowSensor measures the volume flow rate.

Thermodynamic equations are defined by Partials.FlowSensor.

#### Parameters

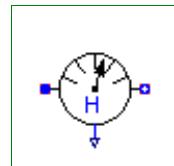
Name	Default	Description
medium	FluidHeatFlow.Media.Medium()	Medium in the component

#### Connectors

Name	Description
flowPort_a	
flowPort_b	

---

### Modelica.Thermal.FluidHeatFlow.Sensors.H\_flowSensor



Enthalpy flow sensor

#### Information

H\_flowSensor measures the enthalpy flow rate.

Thermodynamic equations are defined by Partials.FlowSensor.

## Parameters

Name	Default	Description
medium	FluidHeatFlow.Media.Medium()	Medium in the component

## Connectors

Name	Description
flowPort_a	
flowPort_b	

## Modelica.Thermal.FluidHeatFlow.Sources

Ideal fluid sources, e.g., ambient, volume flow

## Information

This package contains different types of sources:

- Ambient with constant pressure and temperature
- Ambient with prescribed pressure and temperature
- AbsolutePressure to define pressure level of a closed cooling cycle.
- Constant and prescribed volume flow
- Constant and prescribed pressure increase
- Simple pump with mechanical flange

Thermodynamic equations are defined in partial models (package Interfaces.Partial).

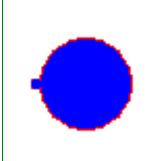
All fans / pumps are considered without losses, they do not change enthalpy flow.

## Package Content

Name	Description
Ambient	Ambient with constant properties
PrescribedAmbient	Ambient with prescirbed properties
AbsolutePressure	Defines absolute pressure level
ConstantVolumeFlow	Enforces constant volume flow
PrescribedVolumeFlow	Enforces prescribed volume flow
ConstantPressureIncrease	Enforces constant pressure increase
PrescribedPressureIncrease	Enforces prescribed pressure increase
IdealPump	Model of an ideal pump

## Modelica.Thermal.FluidHeatFlow.Sources.Ambient

Ambient with constant properties



## Information

(Infinite) ambient with constant pressure and temperature.  
Thermodynamic equations are defined by Partial.Ambient.

## 1830 Modelica.Thermal.FluidHeatFlow.Sources.Ambient

---

### Parameters

Name	Default	Description
medium	FluidHeatFlow.Media.Medium()	Ambient's medium
p_Ambient	0	Ambient's pressure [Pa]
T_Ambient	Modelica.Slunits.Conversions...	Ambient's temperature [K]

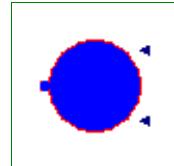
### Connectors

Name	Description
flowPort	

---

## Modelica.Thermal.FluidHeatFlow.Sources.PrescribedAmbient

Ambient with prescirbed properties



### Information

(Infinite) ambient with prescribed pressure and temperature.  
Thermodynamic equations are defined by Partials.Ambient.

### Parameters

Name	Default	Description
medium	FluidHeatFlow.Media.Medium()	Ambient's medium

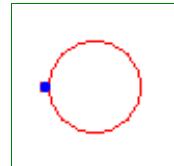
### Connectors

Name	Description
flowPort	
p_Ambient	
T_Ambient	

---

## Modelica.Thermal.FluidHeatFlow.Sources.AbsolutePressure

Defines absolute pressure level



### Information

AbsolutePressure to define pressure level of a closed cooling cycle. Coolant's mass flow, temperature and enthalpy flow are not affected.

### Parameters

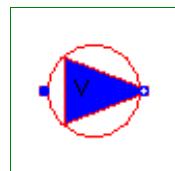
Name	Default	Description
medium	FluidHeatFlow.Media.Medium()	Medium
p	0	Pressure ground [Pa]

### Connectors

Name	Description
flowPort	

**Modelica.Thermal.FluidHeatFlow.Sources.ConstantVolumeFlow**

Enforces constant volume flow

**Information**

Fan resp. pump with constant volume flow rate. Pressure increase is the response of the whole system. Coolant's temperature and enthalpy flow are not affected.

Setting parameter m (mass of medium within fan/pump) to zero leads to neglection of temperature transient  $cv^*m^*\text{der}(T)$ .

Thermodynamic equations are defined by Partials.TwoPort.

**Parameters**

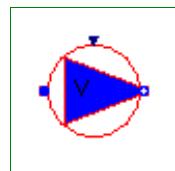
Name	Default	Description
medium	FluidHeatFlow.Media.Medium()	Medium in the component
m	1	Mass of medium [kg]
T0	Modelica.SIunits.Conversions...	Initial temperature of medium [K]
VolumeFlow	1	Volume flow rate [m <sup>3</sup> /s]

**Connectors**

Name	Description
flowPort_a	
flowPort_b	

**Modelica.Thermal.FluidHeatFlow.Sources.PrescribedVolumeFlow**

Enforces prescribed volume flow

**Information**

Fan resp. pump with prescribed volume flow rate. Pressure increase is the response of the whole system. Coolant's temperature and enthalpy flow are not affected.

Setting parameter m (mass of medium within fan/pump) to zero leads to neglection of temperature transient  $cv^*m^*\text{der}(T)$ .

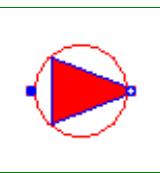
Thermodynamic equations are defined by Partials.TwoPort.

**Parameters**

Name	Default	Description
medium	FluidHeatFlow.Media.Medium()	Medium in the component
m	1	Mass of medium [kg]
T0	Modelica.SIunits.Conversions...	Initial temperature of medium [K]

**Connectors**

Name	Description
flowPort_a	
flowPort_b	
VolumeFlow	

**Modelica.Thermal.FluidHeatFlow.Sources.ConstantPressureIncrease****Enforces constant pressure increase****Information**

Fan resp. pump with constant pressure increase. Mass resp. volume flow is the response of the whole system. Coolant's temperature and enthalpy flow are not affected.

Setting parameter m (mass of medium within fan/pump) to zero leads to neglection of temperature transient  $cv^*m^*\text{der}(T)$ .

Thermodynamic equations are defined by Partials.TwoPort.

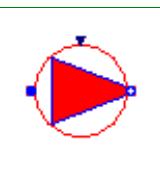
**Parameters**

Name	Default	Description
medium	FluidHeatFlow.Media.Medium()	Medium in the component
m	1	Mass of medium [kg]
T0	Modelica.SIunits.Conversions...	Initial temperature of medium [K]
PressureIncrease	1	Pressure increase [Pa]

**Connectors**

Name	Description
flowPort_a	
flowPort_b	

---

**Modelica.Thermal.FluidHeatFlow.Sources.PrescribedPressureIncrease****Enforces prescribed pressure increase****Information**

Fan resp. pump with prescribed pressure increase. Mass resp. volume flow is the response of the whole system. Coolant's temperature and enthalpy flow are not affected.

Setting parameter m (mass of medium within fan/pump) to zero leads to neglection of temperature transient  $cv^*m^*\text{der}(T)$ .

Thermodynamic equations are defined by Partials.TwoPort.

**Parameters**

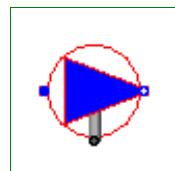
Name	Default	Description
medium	FluidHeatFlow.Media.Medium()	Medium in the component
m	1	Mass of medium [kg]
T0	Modelica.SIunits.Conversions...	Initial temperature of medium [K]

**Connectors**

Name	Description
flowPort_a	
flowPort_b	
PressureIncrease	

**Modelica.Thermal.FluidHeatFlow.Sources.IdealPump**

Model of an ideal pump

**Information**

Simple fan resp. pump where characteristic is dependent on shaft's speed,  
torque \* speed = pressure increase \* volume flow (without losses)

Pressure increase versus volume flow is defined by a linear function, from  $dp_0(V_{flow}=0)$  to  $V_{flow0}(dp=0)$ .  
The axis intersections vary with speed as follows:

- $dp \propto \text{speed}^2$
- $V_{flow} \propto \text{speed}$

Coolant's temperature and enthalpy flow are not affected.

Setting parameter  $m$  (mass of medium within fan/pump) to zero leads to neglection of temperature transient  $cv*m*\text{der}(T)$ .

Thermodynamic equations are defined by `Partials.TwoPort`.

**Parameters**

Name	Default	Description
medium	FluidHeatFlow.Media.Medium()	Medium in the component
$m$	1	Mass of medium [kg]
$T_0$	Modelica.Slunits.Conversions...	Initial temperature of medium [K]
Pump characteristic		
$w_{Nominal}$	1	Nominal speed [rad/s]
$dp_0$	2	Max. pressure increase @ $V_{flow}=0$ [Pa]
$V_{flow0}$	2	Max. volume flow rate @ $dp=0$ [m <sup>3</sup> /s]

**Connectors**

Name	Description
flowPort_a	
flowPort_b	
flange_a	

**Modelica.Thermal.HeatTransfer**

Library of 1-dimensional heat transfer with lumped elements

**Information**

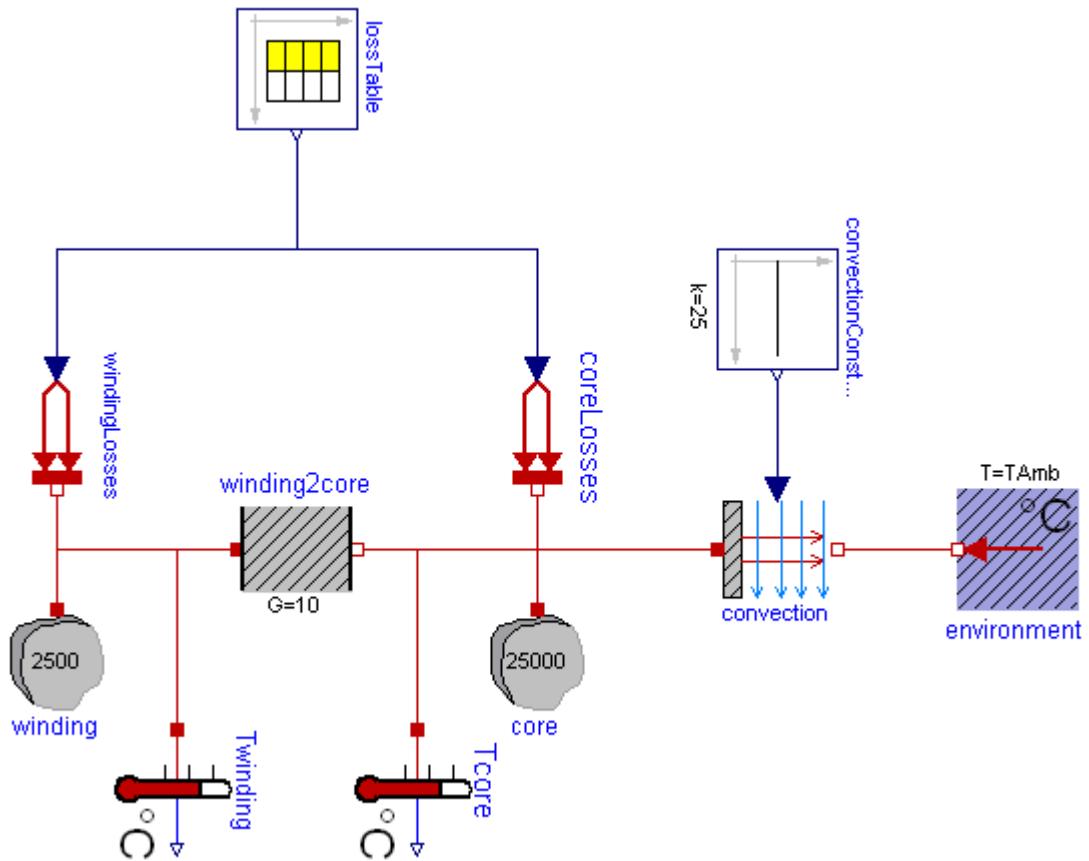
This package contains components to model **1-dimensional heat transfer** with lumped elements. This allows especially to model heat transfer in machines provided the parameters of the lumped elements, such as the heat capacity of a part, can be determined by measurements (due to the complex geometries and many materials used in machines, calculating the lumped element parameters from some basic analytic formulas is usually not possible).

Example models how to use this library are given in subpackage **Examples**.

For a first simple example, see **Examples.TwoMasses** where two masses with different initial temperatures are getting in contact to each other and arriving after some time at a common temperature.

**Examples.ControlledTemperature** shows how to hold a temperature within desired limits by switching on and off an electric resistor.

A more realistic example is provided in **Examples.Motor** where the heating of an electrical motor is modelled, see the following screen shot of this example:



The **filled and non-filled red squares** at the left and right side of a component represent **thermal ports** (connector HeatPort). Drawing a line between such squares means that they are thermally connected. The variables of a HeatPort connector are the temperature **T** at the port and the heat flow rate **Q\_flow** flowing into the component (if **Q\_flow** is positive, the heat flows into the element, otherwise it flows out of the element):

```
Modelica.SIunits.Temperature T "absolute temperature at port in Kelvin";
Modelica.SIunits.HeatFlowRate Q_flow "flow rate at the port in Watt";
```

Note, that all temperatures of this package, including initial conditions, are given in Kelvin. For convenience, in subpackages **HeatTransfer.Celsius**, **HeatTransfer.Fahrenheit** and **HeatTransfer.Rankine** components are provided such that source and sensor information is available in degree Celsius, degree Fahrenheit, or degree Rankine, respectively. Additionally, in package **SIunits.Conversions** conversion functions between the units Kelvin and Celsius, Fahrenheit, Rankine are provided. These functions may be used in the following way:

```
import SI=Modelica.SIunits;
import Modelica.SIunits.Conversions.*;
...
parameter SI.Temperature T = from_degC(25); // convert 25 degree Celsius to
Kelvin
```

There are several other components available, such as AxialConduction (discretized PDE in axial direction),

which have been temporarily removed from this library. The reason is that these components reference material properties, such as thermal conductivity, and currently the Modelica design group is discussing a general scheme to describe material properties.

For technical details in the design of this library, see the following reference:

**Michael Tiller (2001): *Introduction to Physical Modeling with Modelica*.** Kluwer Academic Publishers Boston.

#### Acknowledgements:

Several helpful remarks from the following persons are acknowledged: John Batteh, Ford Motors, Dearborn, U.S.A; **Anton Haumer**, Technical Consulting & Electrical Engineering, Austria; Ludwig Marvan, VA TECH ELIN EBG Elektronik GmbH, Wien, Austria; Hans Olsson, Dynasim AB, Sweden; Hubertus Tummescheit, Lund Institute of Technology, Lund, Sweden.

**Copyright © 2001-2007, Modelica Association, Michael Tiller and DLR.**

*This Modelica package is free software; it can be redistributed and/or modified under the terms of the Modelica license, see the license conditions and the accompanying disclaimer in the documentation of package Modelica in file "Modelica/package.mo".*

#### Package Content

Name	Description
 Examples	Example models to demonstrate the usage of package Modelica.Thermal.HeatTransfer
 Interfaces	Connectors and partial models
 HeatCapacitor	Lumped thermal element storing heat
 ThermalConductor	Lumped thermal element transporting heat without storing it
 Convection	Lumped thermal element for heat convection
 BodyRadiation	Lumped thermal element for radiation heat transfer
 FixedTemperature	Fixed temperature boundary condition in Kelvin
 PrescribedTemperature	Variable temperature boundary condition in Kelvin
 FixedHeatFlow	Fixed heat flow boundary condition
 PrescribedHeatFlow	Prescribed heat flow boundary condition
 TemperatureSensor	Absolute temperature sensor in Kelvin
 RelTemperatureSensor	Relative Temperature sensor
 HeatFlowSensor	Heat flow rate sensor
 Celsius	Components with Celsius input and/or output
 Fahrenheit	Components with Fahrenheit input and/or output
 Rankine	Components with Rankine input and/or output

#### Modelica.Thermal.HeatTransfer.Examples

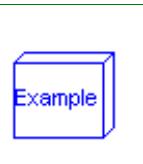
Example models to demonstrate the usage of package Modelica.Thermal.HeatTransfer

## Information

### Package Content

Name	Description
<a href="#">TwoMasses</a>	Simple conduction demo
<a href="#">ControlledTemperature</a>	Control temperature of a resistor
<a href="#">Motor</a>	Second order thermal model of a motor

### Modelica.Thermal.HeatTransfer.Examples.TwoMasses



#### Simple conduction demo

## Information

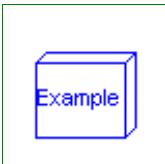
This example demonstrates the thermal response of two masses connected by a conducting element. The two masses have the same heat capacity but different initial temperatures ( $T_1=100$  [degC],  $T_2= 0$  [degC]). The mass with the higher temperature will cool off while the mass with the lower temperature heats up. They will each asymptotically approach the calculated temperature  $T_{final\_K}$  ( $T_{final\_degC}$ ) that results from dividing the total initial energy in the system by the sum of the heat capacities of each element.

Simulate for 5 s and plot the variables  
 $mass1.T$ ,  $mass2.T$ ,  $T_{final\_K}$  or  
 $Tsensor1.T$ ,  $Tsensor2.T$ ,  $T_{final\_degC}$

## Parameters

Name	Default	Description
$T_{final\_K}$		Projected final temperature [K]
$T_{final\_degC}$		Projected final temperature [degC]

### Modelica.Thermal.HeatTransfer.Examples.ControlledTemperature



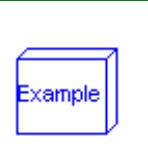
#### Control temperature of a resistor

## Information

A constant voltage of 10 V is applied to a temperature dependent resistor of  $10*(1+(T-20)/(235+20))$  Ohms, whose losses  $v^{**2}/r$  are dissipated via a thermal conductance of 0.1 W/K to ambient temperature 20 degree C. The resistor is assumed to have a thermal capacity of 1 J/K, having ambient tempartature at the beginning of the experiment. The temperature of this heating resistor is held by an OnOff-controller at reference temperature within a given bandwith +/- 1 K by switching on and off the voltage source. The reference temperature starts at 25 degree C and rises between  $t = 2$  and 8 seconds linear to 50 degree C. An appropriate simulating time would be 10 seconds.

## Parameters

Name	Default	Description
$TAmb$	20	Ambient Temperature [degC]
$TDif$	2	Error in Temperature [degC]

**Modelica.Thermal.HeatTransfer.Examples.Motor**

**Second order thermal model of a motor**

**Information**

This example contains a simple second order thermal model of a motor. The periodic power losses are described by table "lossTable":

time winding losses core losses

0	100	500
360	100	500
360	1000	500
600	1000	500

Since constant speed is assumed, the core losses keep constant whereas the winding losses are low for 6 minutes (no-load) and high for 4 minutes (over load).

The winding losses are corrected by  $(1 + \alpha * (T - T_{ref}))$  because the winding's resistance is temperature dependent whereas the core losses are kept constant ( $\alpha = 0$ ).

The power dissipation to the environment is approximated by heat flow through a thermal conductance between winding and core, partially storage of the heat in the winding's heat capacity as well as the core's heat capacity and finally by forced convection to the environment.

Since constant speed is assumed, the convective conductance keeps constant.

Using Modelica.Thermal.FluidHeatFlow it would be possible to model the coolant air flow, too (instead of simple dissipation to a constant ambient's temperature).

Simulate for 7200 s; plot Twinding.T and Tcore.T.

**Parameters**

Name	Default	Description
TAmb	20	Ambient temperature [degC]

**Modelica.Thermal.HeatTransfer.Interfaces****Connectors and partial models****Information****Package Content**

Name	Description
HeatPort	Thermal port for 1-dim. heat transfer
HeatPort_a	Thermal port for 1-dim. heat transfer (filled rectangular icon)
HeatPort_b	Thermal port for 1-dim. heat transfer (unfilled rectangular icon)
Element1D	Partial heat transfer element with two HeatPort connectors that does not store energy

**Modelica.Thermal.HeatTransfer.Interfaces.HeatPort****Thermal port for 1-dim. heat transfer**

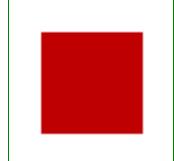
## Information

## Contents

Name	Description
T	Port temperature [K]
Q_flow	Heat flow rate (positive if flowing from outside into the component) [W]

### Modelica.Thermal.HeatTransfer.Interfaces.HeatPort\_a

Thermal port for 1-dim. heat transfer (filled rectangular icon)



## Information

This connector is used for 1-dimensional heat flow between components. The variables in the connector are:

T Temperature in [Kelvin].  
Q\_flow Heat flow rate in [Watt].

According to the Modelica sign convention, a **positive** heat flow rate **Q\_flow** is considered to flow **into a component**. This convention has to be used whenever this connector is used in a model class.

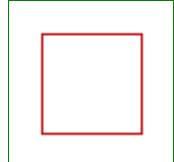
Note, that the two connector classes **HeatPort\_a** and **HeatPort\_b** are identical with the only exception of the different **icon layout**.

## Contents

Name	Description
T	Port temperature [K]
Q_flow	Heat flow rate (positive if flowing from outside into the component) [W]

### Modelica.Thermal.HeatTransfer.Interfaces.HeatPort\_b

Thermal port for 1-dim. heat transfer (unfilled rectangular icon)



## Information

This connector is used for 1-dimensional heat flow between components. The variables in the connector are:

T Temperature in [Kelvin].  
Q\_flow Heat flow rate in [Watt].

According to the Modelica sign convention, a **positive** heat flow rate **Q\_flow** is considered to flow **into a component**. This convention has to be used whenever this connector is used in a model class.

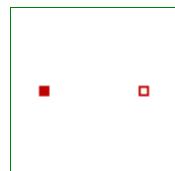
Note, that the two connector classes **HeatPort\_a** and **HeatPort\_b** are identical with the only exception of the different **icon layout**.

## Contents

Name	Description
T	Port temperature [K]
Q_flow	Heat flow rate (positive if flowing from outside into the component) [W]

**Modelica.Thermal.HeatTransfer.Interfaces.Element1D**

Partial heat transfer element with two HeatPort connectors that does not store energy

**Information**

This partial model contains the basic connectors and variables to allow heat transfer models to be created that **do not store energy**. This model defines and includes equations for the temperature drop across the element, **dT**, and the heat flow rate through the element from port\_a to port\_b, **Q\_flow**.

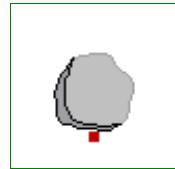
By extending this model, it is possible to write simple constitutive equations for many types of heat transfer components.

**Connectors**

Name	Description
port_a	
port_b	

**Modelica.Thermal.HeatTransfer.HeatCapacitor**

Lumped thermal element storing heat

**Information**

This is a generic model for the heat capacity of a material. No specific geometry is assumed beyond a total volume with uniform temperature for the entire volume. Furthermore, it is assumed that the heat capacity is constant (independent of temperature).

The temperature T [Kelvin] of this component is a **state**. A default of T = 25 degree Celsius (= Slunits.Conversions.from\_degC(25)) is used as start value for initialization. This usually means that at start of integration the temperature of this component is 25 degrees Celsius. You may, of course, define a different temperature as start value for initialization. Alternatively, it is possible to set parameter **steadyStateStart** to **true**. In this case the additional equation '**der(T) = 0**' is used during initialization, i.e., the temperature T is computed in such a way that the component starts in **steady state**. This is useful in cases, where one would like to start simulation in a suitable operating point without being forced to integrate for a long time to arrive at this point.

Note, that parameter **steadyStateStart** is not available in the parameter menu of the simulation window, because its value is utilized during translation to generate quite different equations depending on its setting. Therefore, the value of this parameter can only be changed before translating the model.

This component may be used for complicated geometries where the heat capacity C is determined by measurements. If the component consists mainly of one type of material, the **mass m** of the component may be measured or calculated and multiplied with the **specific heat capacity cp** of the component material to compute C:

```
C = cp*m.  
Typical values for cp at 20 degC in J/(kg.K):  
aluminium    896  
concrete      840  
copper        383  
iron          452  
silver         235  
steel          420 ... 500 (V2A)  
wood           2500
```

## Parameters

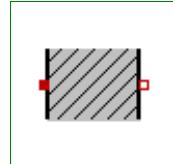
Name	Default	Description
C		Heat capacity of part (= cp*m) [J/K]
steadyStateStart	false	true, if component shall start in steady state

## Connectors

Name	Description
port	

## Modelica.Thermal.HeatTransfer.ThermalConductor

Lumped thermal element transporting heat without storing it



## Information

This is a model for transport of heat without storing it. It may be used for complicated geometries where the thermal conductance G (= inverse of thermal resistance) is determined by measurements and is assumed to be constant over the range of operations. If the component consists mainly of one type of material and a regular geometry, it may be calculated, e.g., with one of the following equations:

- Conductance for a **box** geometry under the assumption that heat flows along the box length:

```
G = k*A/L
k: Thermal conductivity (material constant)
A: Area of box
L: Length of box
```

- Conductance for a **cylindrical** geometry under the assumption that heat flows from the inside to the outside radius of the cylinder:

```
G = 2*pi*k*L/log(r_out/r_in)
pi   : Modelica.Constants.pi
k    : Thermal conductivity (material constant)
L    : Length of cylinder
log  : Modelica.Math.log;
r_out: Outer radius of cylinder
r_in : Inner radius of cylinder
```

Typical values for k at 20 degC in W/(m.K) :

aluminium	220
concrete	1
copper	384
iron	74
silver	407
steel	45 .. 15 (V2A)
wood	0.1 ... 0.2

## Parameters

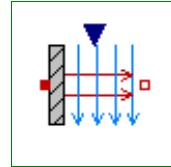
Name	Default	Description
G		Constant thermal conductance of material [W/K]

## Connectors

Name	Description
port_a	
port_b	

## Modelica.Thermal.HeatTransfer.Convection

### Lumped thermal element for heat convection



### Information

This is a model of linear heat convection, e.g., the heat transfer between a plate and the surrounding air. It may be used for complicated solid geometries and fluid flow over the solid by determining the convective thermal conductance  $G_c$  by measurements. The basic constitutive equation for convection is

```
Q_flow = Gc * (solid.T - fluid.T);
Q_flow: Heat flow rate from connector 'solid' (e.g. a plate)
        to connector 'fluid' (e.g. the surrounding air)
```

$G_c = G.signal[1]$  is an input signal to the component, since  $G_c$  is nearly never constant in practice. For example,  $G_c$  may be a function of the speed of a cooling fan. For simple situations,  $G_c$  may be *calculated* according to

```
Gc = A*h
A: Convection area (e.g. perimeter*length of a box)
h: Heat transfer coefficient
```

where the heat transfer coefficient  $h$  is calculated from properties of the fluid flowing over the solid.  
Examples:

**Machines cooled by air** (empirical, very rough approximation according to R. Fischer: Elektrische Maschinen, 10th edition, Hanser-Verlag 1999, p. 378):

```
h = 7.8*v^0.78 [W/(m².K)] (forced convection)
= 12 [W/(m².K)] (free convection)
where
v: Air velocity in [m/s]
```

**Laminar flow with constant velocity of a fluid along a flat plate** where the heat flow rate from the plate to the fluid (= solid.Q\_flow) is kept constant (according to J.P.Holman: Heat Transfer, 8th edition, McGraw-Hill, 1997, p.270):

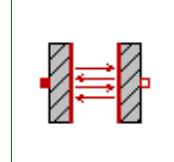
```
h = Nu*k/x;
Nu = 0.453*Re^(1/2)*Pr^(1/3);
where
h : Heat transfer coefficient
Nu : = h*x/k (Nusselt number)
Re : = v*x*rho/mue (Reynolds number)
Pr : = cp*mue/k (Prandtl number)
v : Absolute velocity of fluid
x : distance from leading edge of flat plate
rho: density of fluid (material constant)
mue: dynamic viscosity of fluid (material constant)
cp : specific heat capacity of fluid (material constant)
k : thermal conductivity of fluid (material constant)
and the equation for h holds, provided
Re < 5e5 and 0.6 < Pr < 50
```

## Connectors

Name	Description
Gc	Signal representing the convective thermal conductance in [W/K]
solid	
fluid	

## Modelica.Thermal.HeatTransfer.BodyRadiation

Lumped thermal element for radiation heat transfer



## Information

This is a model describing the thermal radiation, i.e., electromagnetic radiation emitted between two bodies as a result of their temperatures. The following constitutive equation is used:

$$Q_{\text{flow}} = Gr * \sigma * (\text{port}_a.T^4 - \text{port}_b.T^4);$$

where  $Gr$  is the radiation conductance and  $\sigma$  is the Stefan-Boltzmann constant (= Modelica.Constants.sigma).  $Gr$  may be determined by measurements and is assumed to be constant over the range of operations.

For simple cases,  $Gr$  may be analytically computed. The analytical equations use  $\epsilon$ , the emission value of a body which is in the range 0..1.  $\epsilon=1$ , if the body absorbs all radiation (= black body).  $\epsilon=0$ , if the body reflects all radiation and does not absorb any.

Typical values for  $\epsilon$ :

aluminium, polished	0.04
copper, polished	0.04
gold, polished	0.02
paper	0.09
rubber	0.95
silver, polished	0.02
wood	0.85..0.9

## Analytical Equations for $Gr$

**Small convex object in large enclosure** (e.g., a hot machine in a room):

$Gr = \epsilon * A$   
where  
 $\epsilon$ : Emission value of object (0..1)  
 $A$ : Surface area of object where radiation heat transfer takes place

**Two parallel plates:**

$Gr = A / (1/e1 + 1/e2 - 1)$   
where  
 $e1$ : Emission value of plate1 (0..1)  
 $e2$ : Emission value of plate2 (0..1)  
 $A$  : Area of plate1 (= area of plate2)

**Two long cylinders in each other**, where radiation takes place from the inner to the outer cylinder):

$Gr = 2 * \pi * r1 * L / (1/e1 + (1/e2 - 1) * (r1/r2))$   
where  
 $\pi$ : = Modelica.Constants.pi  
 $r1$ : Radius of inner cylinder

r2: Radius of outer cylinder  
 L : Length of the two cylinders  
 e1: Emission value of inner cylinder (0..1)  
 e2: Emission value of outer cylinder (0..1)

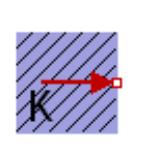
## Parameters

Name	Default	Description
Gr		Net radiation conductance between two surfaces (see docu) [m <sup>2</sup> ]

## Connectors

Name	Description
port_a	
port_b	

## Modelica.Thermal.HeatTransfer.FixedTemperature



Fixed temperature boundary condition in Kelvin

## Information

This model defines a fixed temperature T at its port in Kelvin, i.e., it defines a fixed temperature as a boundary condition.

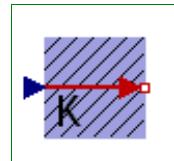
## Parameters

Name	Default	Description
T		Fixed temperature at port [K]

## Connectors

Name	Description
port	

## Modelica.Thermal.HeatTransfer.PrescribedTemperature



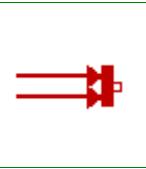
Variable temperature boundary condition in Kelvin

## Information

This model represents a variable temperature boundary condition. The temperature in [K] is given as input signal T to the model. The effect is that an instance of this model acts as an infinite reservoir able to absorb or generate as much energy as required to keep the temperature at the specified value.

## Connectors

Name	Description
port	
T	

**Modelica.Thermal.HeatTransfer.FixedHeatFlow****Fixed heat flow boundary condition****Information**

This model allows a specified amount of heat flow rate to be "injected" into a thermal system at a given port. The constant amount of heat flow rate  $Q_{\text{flow}}$  is given as a parameter. The heat flows into the component to which the component FixedHeatFlow is connected, if parameter  $Q_{\text{flow}}$  is positive.

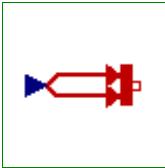
If parameter alpha is  $> 0$ , the heat flow is multiplied by  $(1 + \alpha * (\text{port.T} - \text{T}_{\text{ref}}))$  in order to simulate temperature dependent losses (which are given an reference temperature  $\text{T}_{\text{ref}}$ ).

**Parameters**

Name	Default	Description
$Q_{\text{flow}}$		Fixed heat flow rate at port [W]
$\text{T}_{\text{ref}}$	from_degC(20)	Reference temperature [K]
alpha	0	Temperature coefficient of heat flow rate [1/K]

**Connectors**

Name	Description
port	

**Modelica.Thermal.HeatTransfer.PrescribedHeatFlow****Prescribed heat flow boundary condition****Information**

This model allows a specified amount of heat flow rate to be "injected" into a thermal system at a given port. The amount of heat is given by the input signal  $Q_{\text{flow}}$  into the model. The heat flows into the component to which the component PrescribedHeatFlow is connected, if the input signal is positive.

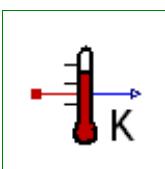
If parameter alpha is  $> 0$ , the heat flow is multiplied by  $(1 + \alpha * (\text{port.T} - \text{T}_{\text{ref}}))$  in order to simulate temperature dependent losses (which are given an reference temperature  $\text{T}_{\text{ref}}$ ).

**Parameters**

Name	Default	Description
$\text{T}_{\text{ref}}$	from_degC(20)	Reference temperature [K]
alpha	0	Temperature coefficient of heat flow rate [1/K]

**Connectors**

Name	Description
$Q_{\text{flow}}$	
port	

**Modelica.Thermal.HeatTransfer.TemperatureSensor****Absolute temperature sensor in Kelvin**

## Information

This is an ideal absolute temperature sensor which returns the temperature of the connected port in Kelvin as an output signal. The sensor itself has no thermal interaction with whatever it is connected to. Furthermore, no thermocouple-like lags are associated with this sensor model.

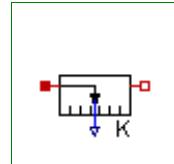
## Connectors

Name	Description
T	
port	

---

## Modelica.Thermal.HeatTransfer.RelTemperatureSensor

Relative Temperature sensor



## Information

The relative temperature "port\_a.T - port\_b.T" is determined between the two ports of this component and is provided as output signal in Kelvin.

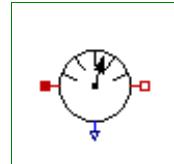
## Connectors

Name	Description
port_a	
port_b	
T_rel	

---

## Modelica.Thermal.HeatTransfer.HeatFlowSensor

Heat flow rate sensor



## Information

This model is capable of monitoring the heat flow rate flowing through this component. The sensed value of heat flow rate is the amount that passes through this sensor while keeping the temperature drop across the sensor zero. This is an ideal model so it does not absorb any energy and it has no direct effect on the thermal response of a system it is included in. The output signal is positive, if the heat flows from port\_a to port\_b.

## Connectors

Name	Description
Q_flow	Heat flow from port_a to port_b
port_a	
port_b	

---

## Modelica.Thermal.HeatTransfer.Celsius

Components with Celsius input and/or output

## Information

The components of this package are provided for the convenience of people working mostly with Celsius units, since all models in package HeatTransfer are based on Kelvin units.

Note, that in package Slunits.Conversions, functions are provided to convert between the units Kelvin, degree Celsius, degree Fahrenheit, and degree Rankine. These functions allow, e.g., a direct conversion of units at all places where Kelvin is required as parameter. Example:

```
import SIunits.Conversions.*;
Modelica.Thermal.HeatCapacitor C(T0 = from_degC(20));
```

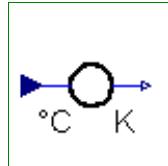
## Package Content

Name	Description
ToKelvin	Conversion block from °Celsius to Kelvin
FromKelvin	Conversion from Kelvin to °Celsius
FixedTemperature	Fixed temperature boundary condition in degree Celsius
PrescribedTemperature	Variable temperature boundary condition in °Celsius
TemperatureSensor	Absolute temperature sensor in °Celsius

---

### Modelica.Thermal.HeatTransfer.Celsius.ToKelvin

Conversion block from °Celsius to Kelvin



#### Information

This component converts an input signal from Celsius to Kelvin and provide is as output signal.

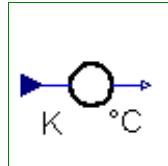
#### Connectors

Name	Description
Celsius	
Kelvin	

---

### Modelica.Thermal.HeatTransfer.Celsius.FromKelvin

Conversion from Kelvin to °Celsius

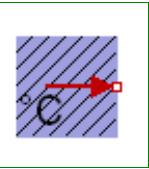


#### Information

This component converts an input signal from Kelvin to Celsius and provides is as output signal.

#### Connectors

Name	Description
Kelvin	
Celsius	

**Modelica.Thermal.HeatTransfer.Celsius.FixedTemperature**

**Fixed temperature boundary condition in degree Celsius**

**Information**

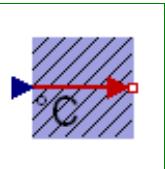
This model defines a fixed temperature  $T$  at its port in [degC], i.e., it defines a fixed temperature as a boundary condition.

**Parameters**

Name	Default	Description
$T$		Fixed Temperature at the port [degC]

**Connectors**

Name	Description
port	

**Modelica.Thermal.HeatTransfer.Celsius.PrescribedTemperature**

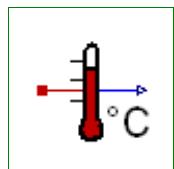
**Variable temperature boundary condition in °Celsius**

**Information**

This model represents a variable temperature boundary condition. The temperature value in [degC] is given by the input signal to the model. The effect is that an instance of this model acts as an infinite reservoir able to absorb or generate as much energy as required to keep the temperature at the specified value.

**Connectors**

Name	Description
port	
$T$	

**Modelica.Thermal.HeatTransfer.Celsius.TemperatureSensor**

**Absolute temperature sensor in °Celsius**

**Information**

This is an ideal absolute temperature sensor which returns the temperature of the connected port in Celsius as an output signal. The sensor itself has no thermal interaction with whatever it is connected to. Furthermore, no thermocouple-like lags are associated with this sensor model.

**Connectors**

Name	Description
$T$	
port	

## Modelica.Thermal.HeatTransfer.Fahrenheit

### Components with Fahrenheit input and/or output

#### Information

The components of this package are provided for the convenience of people working mostly with Fahrenheit units, since all models in package HeatTransfer are based on Kelvin units.

Note, that in package Slunits.Conversions, functions are provided to convert between the units Kelvin, degree Celsius, degree Fahrenheit and degree Rankine. These functions allow, e.g., a direct conversion of units at all places where Kelvin is required as parameter. Example:

```
import SIunits.Conversions.*;
Modelica.Thermal.HeatTransfer.HeatCapacitor C(T0 = from_degF(70));
```

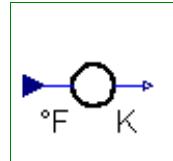
#### Package Content

Name	Description
ToKelvin	Conversion block from °Fahrenheit to Kelvin
FromKelvin	Conversion from Kelvin to °Fahrenheit
FixedTemperature	Fixed temperature boundary condition in °Fahrenheit
PrescribedTemperature	Variable temperature boundary condition in °Fahrenheit
TemperatureSensor	Absolute temperature sensor in °Fahrenheit

---

## Modelica.Thermal.HeatTransfer.Fahrenheit.ToKelvin

### Conversion block from °Fahrenheit to Kelvin



#### Information

This component converts a input signal from degree Fahrenheit to Kelvin and provides is as output signal.

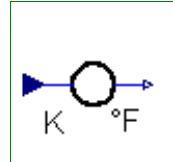
#### Connectors

Name	Description
Fahrenheit	
Kelvin	

---

## Modelica.Thermal.HeatTransfer.Fahrenheit.FromKelvin

### Conversion from Kelvin to °Fahrenheit



#### Information

This component converts all input signals from Kelvin to Fahrenheit and provides them as output signals.

#### Parameters

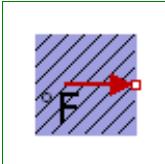
Name	Default	Description
n	1	Number of inputs (= number of outputs)

## Connectors

Name	Description
Kelvin	
Fahrenheit	

## Modelica.Thermal.HeatTransfer.Fahrenheit.FixedTemperature

Fixed temperature boundary condition in °Fahrenheit



## Information

This model defines a fixed temperature  $T$  at its port in [degF], i.e., it defines a fixed temperature as a boundary condition.

## Parameters

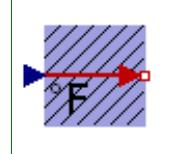
Name	Default	Description
$T$		Fixed Temperature at the port [degF]

## Connectors

Name	Description
port	

## Modelica.Thermal.HeatTransfer.Fahrenheit.PrescribedTemperature

Variable temperature boundary condition in °Fahrenheit



## Information

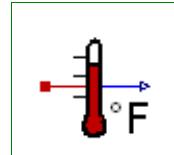
This model represents a variable temperature boundary condition. The temperature value in [degF] is given by the input signal to the model. The effect is that an instance of this model acts as an infinite reservoir able to absorb or generate as much energy as required to keep the temperature at the specified value.

## Connectors

Name	Description
port	
$T$	

## Modelica.Thermal.HeatTransfer.Fahrenheit.TemperatureSensor

Absolute temperature sensor in °Fahrenheit



## Information

This is an ideal absolute temperature sensor which returns the temperature of the connected port in Fahrenheit as an output signal. The sensor itself has no thermal interaction with whatever it is connected to. Furthermore, no thermocouple-like lags are associated with this sensor model.

## Connectors

Name	Description
T	
port	

---

## Modelica.Thermal.HeatTransfer.Rankine

Components with Rankine input and/or output

## Information

The components of this package are provided for the convenience of people working mostly with Rankine units, since all models in package HeatTransfer are based on Kelvin units.

Note, that in package Slunits.Conversions, functions are provided to convert between the units Kelvin, degree Celsius, degree Fahrenheit and degree Rankine. These functions allow, e.g., a direct conversion of units at all places where Kelvin is required as parameter. Example:

```
import SIunits.Conversions.*;
Modelica.Thermal.HeatTransfer.HeatCapacitor C(T0 = from_degRk(500));
```

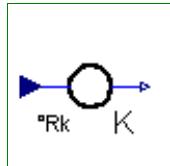
## Package Content

Name	Description
ToKelvin	Conversion block from °Rankine to Kelvin
FromKelvin	Conversion from Kelvin to °Rankine
FixedTemperature	Fixed temperature boundary condition in °Rankine
PrescribedTemperature	Variable temperature boundary condition in °Rankine
TemperatureSensor	Absolute temperature sensor in °Rankine

---

## Modelica.Thermal.HeatTransfer.Rankine.ToKelvin

Conversion block from °Rankine to Kelvin



## Information

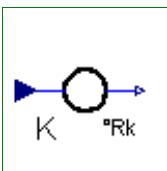
This component converts all input signals from degree Rankine to Kelvin and provides them as output signals.

## Parameters

Name	Default	Description
n	1	Number of inputs (= number of outputs)

## Connectors

Name	Description
Rankine	
Kelvin	

**Modelica.Thermal.HeatTransfer.Rankine.FromKelvin****Conversion from Kelvin to °Rankine****Information**

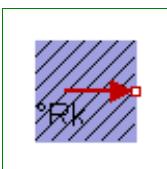
This component converts all input signals from Kelvin to Rankine and provides them as output signals.

**Parameters**

Name	Default	Description
n	1	Number of inputs (= number of outputs)

**Connectors**

Name	Description
Kelvin	
Rankine	

**Modelica.Thermal.HeatTransfer.Rankine.FixedTemperature****Fixed temperature boundary condition in °Rankine****Information**

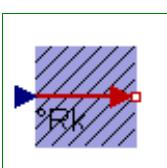
This model defines a fixed temperature T at its port in degree Rankine, [degRk], i.e., it defines a fixed temperature as a boundary condition.

**Parameters**

Name	Default	Description
T		Fixed Temperature at the port [degRk]

**Connectors**

Name	Description
port	

**Modelica.Thermal.HeatTransfer.Rankine.PrescribedTemperature****Variable temperature boundary condition in °Rankine****Information**

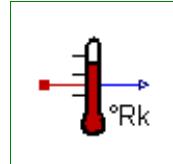
This model represents a variable temperature boundary condition. The temperature value in degree Rankine, [degRk] is given by the input signal to the model. The effect is that an instance of this model acts as an infinite reservoir able to absorb or generate as much energy as required to keep the temperature at the specified value.

## Connectors

Name	Description
port	
T	

## Modelica.Thermal.HeatTransfer.Rankine.TemperatureSensor

Absolute temperature sensor in °Rankine



## Information

This is an ideal absolute temperature sensor which returns the temperature of the connected port in Rankine as an output signal. The sensor itself has no thermal interaction with whatever it is connected to. Furthermore, no thermocouple-like lags are associated with this sensor model.

## Connectors

Name	Description
T	
port	

## Modelica.Utilities

Library of utility functions dedicated to scripting (operating on files, streams, strings, system)

## Information

This package contains Modelica **functions** that are especially suited for **scripting**. The functions might be used to work with strings, read data from file, write data to file or copy, move and remove files.

For an introduction, have especially a look at:

- [Modelica.Utilities.User's Guide](#) discusses the most important aspects of this library.
- [Modelica.Utilities.Examples](#) contains examples that demonstrate the usage of this library.

The following main sublibraries are available:

- [Files](#) provides functions to operate on files and directories, e.g., to copy, move, remove files.
- [Streams](#) provides functions to read from files and write to files.
- [Strings](#) provides functions to operate on strings. E.g. substring, find, replace, sort, scanToken.
- [System](#) provides functions to interact with the environment. E.g., get or set the working directory or environment variables and to send a command to the default shell.

Copyright © 1998-2007, Modelica Association, DLR and Dynasim.

*This Modelica package is free software; it can be redistributed and/or modified under the terms of the Modelica license, see the license conditions and the accompanying disclaimer [here](#).*

## Package Content

Name	Description
<a href="#">UsersGuide</a>	User's Guide of Utilities Library
<a href="#">Examples</a>	Examples to demonstrate the usage of package Modelica.Utilities

 Files	Functions to work with files and directories
 Streams	Read from files and write to files
 Strings	Operations on strings
 System	Interaction with environment
 Types	Type definitions used in package Modelica.Utilities

## Modelica.Utilities.UsersGuide

Library **Modelica.Utilities** contains Modelica **functions** that are especially suited for **scripting**. Currently, only a rudimentary user's Guide is present. This will be improved in the next releases. The user's Guide has currently the following chapters:



1. [Release Notes](#) summarizes the differences between different versions of this library.
2. [ImplementationNotes](#) describes design decisions for this library especially for Modelica tool vendors.
3. [Contact](#) provides information about the authors of the library as well as acknowledgments.

### Error handling

In case of error, all functions in this library use a Modelica "assert(..)" to provide an error message and to cancel all actions. This means that functions do not return, if an error is triggered inside the function. In the near future, an exception handling mechanism will be introduced in Modelica that will allow to catch errors at a defined place.

## Modelica.Utilities.Examples

### Examples to demonstrate the usage of package Modelica.Utilities

#### Information

This package contains quite involved examples that demonstrate how to use the functions of package Modelica.Utilities. In particular the following examples are present.

- Function [calculator](#) is an interpreter to evaluate expressions consisting of +,-,\*,/(),sin(), cos(), tan(), sqrt(), pi. For example: calculator("1.5\*sin(pi/6)");
- Function [expression](#) is the basic function used in "calculator" to evaluate an expression. It is useful if the expression interpreter is used in a larger scan operation (such as [readRealParameter](#) below).
- Function [readRealParameter](#) reads the value of a parameter from file, given the file and the parameter name. The value on file is interpreted with the Examples.expression function and can therefore be an expression.
- Model [readRealParameterModel](#) is a test model to demonstrate the usage of "readRealParameter". The model contains 3 parameters that are read from file "Modelica.Utilities/data/Examples\_readRealParameters.txt".

#### Package Content

Name	Description
 <a href="#">calculator</a>	Interpreter to evaluate simple expressions consisting of +,-,*,/(),sin(), cos(), tan(), sqrt(), pi
 <a href="#">expression</a>	Expression interpreter that returns with the position after the expression (expression may consist of +,-,*,/(),sin(), cos(), tan(), sqrt(), pi)

## 1854 Modelica.Utilities.Examples

 <a href="#">readRealParameter</a>	Read the value of a Real parameter from file
 <a href="#">readRealParameterModel</a>	Demonstrate usage of Examples.readRealParameter/.expression

### Modelica.Utilities.Examples.calculator

Interpreter to evaluate simple expressions consisting of +,-,\*,/,(),sin(), cos(), tan(), sqrt(), pi



#### Information

##### Syntax

```
result = calculator(expression);
```

##### Description

This function demonstrates how a simple expression calculator can be implemented in form of a recursive decent parser using basically the Strings.scanToken(..) and Strings.scanDelimiter(..) function.

The following operations are supported (pi=3.14.. is a predefined constant):

```
+,-
*,/
(expression)
sin(expression)
cos(expression)
tan(expression)
sqrt(expression)
pi
```

##### Example

```
calculator("2+3*(4-1)"); // returns 11
calculator("sin(pi/6)"); // returns 0.5
```

#### Inputs

Name	Default	Description
string		Expression that is evaluated

#### Outputs

Name	Description
result	Value of expression

### Modelica.Utilities.Examples.expression

Expression interpreter that returns with the position after the expression (expression may consist of +,-,\*,/,(),sin(), cos(), tan(), sqrt(), pi)



## Information

### Syntax

```
result = expression(string);
(result, nextIndex) = expression(string, startIndex=1, message="");
```

### Description

This function is nearly the same as Examples.calculator. The essential difference is that function "expression" might be used in other parsing operations: After the expression is parsed and evaluated, the function returns with the value of the expression as well as the position of the character directly after the expression.

This function demonstrates how a simple expression calculator can be implemented in form of a recursive decent parser using basically the Strings.scanToken(..) and scanDelimiters(..) function. There are 2 local functions (term, primary) that implement the corresponding part of the grammar.

The following operations are supported (pi=3.14.. is a predefined constant):

```
+,
-,
*,
/
(expression)
sin(expression)
cos(expression)
tan(expression)
sqrt(expression)
pi
```

The optional argument "startIndex" defines at which position scanning of the expression starts.

In case of error, the optional argument "message" is appended to the error message, in order to give additional information where the error occurred.

This function parses the following grammar

```
expression: [ add_op ] term { add_op term }
add_op    : "+" | "-"
term      : primary { mul_op primary }
mul_op    : "*" | "/"
primary   : UNSIGNED_NUMBER
           | pi
           | ( expression )
           | functionName( expression )
function  : sin
           | cos
           | tan
           | sqrt
```

Note, in Examples.readRealParameter it is shown, how the expression function can be used as part of another scan operation.

### Example

```
expression("2+3*(4-1)"); // returns 11
expression("sin(pi/6)"); // returns 0.5
```

## Inputs

Name	Default	Description
string		Expression that is evaluated
startIndex	1	Start scanning of expression at character startIndex
message	""	Message used in error message if scan is not successful

## Outputs

Name	Description
result	Value of expression
nextIndex	Index after the scanned expression

## Modelica.Utilities.Examples.readRealParameter

Read the value of a Real parameter from file



## Information

### Syntax

```
result = readRealParameter(fileName, name);
```

### Description

This function demonstrates how a function can be implemented that reads the value of a parameter from file. The function performs the following actions:

1. It opens file "fileName" and reads the lines of the file.
2. In every line, Modelica line comments ("// ... end-of-line") are skipped
3. If a line consists of "name = expression" and the "name" in this line is identical to the second argument "name" of the function call, the expression calculator Examples.expression is used to evaluate the expression after the "=" character. The expression can optionally be terminated with a ";".
4. The result of the expression evaluation is returned as the value of the parameter "name".

### Example

On file "test.txt" the following lines might be present:

```
// Motor data
J      = 2.3      // inertia
w_rel0 = 1.5*2; // relative angular velocity
phi_rel0 = pi/3
```

The function returns the value "3.0" when called as:

```
readRealParameter("test.txt", "w_rel0")
```

## Inputs

Name	Default	Description
fileName		Name of file
name		Name of parameter

## Outputs

Name	Description
result	Actual value of parameter on file

---

## Modelica.Utilities.Examples.readRealParameterModel

Demonstrate usage of Examples.readRealParameter/.expression



## Information

Model that shows the usage of Examples.readRealParameter and Examples.expression. The model has 3 parameters and the values of these parameters are read from a file.

## Parameters

Name	Default	Description
file	classDirectory() + "data/Exa..."	File on which data is present
J	readRealParameter(file, "J")	[kg.m2]
phi_rel0	readRealParameter(file, "phi...")	[rad]
w_rel0	readRealParameter(file, "w_r...")	[rad/s]

---

## Modelica.Utilities.Files

### Functions to work with files and directories

## Information

This package contains functions to work with files and directories. As a general convention of this package, '/' is used as directory separator both for input and output arguments of all functions. For example:

```
exist("Modelica/Mechanics/Rotational.mo");
```

The functions provide the mapping to the directory separator of the underlying operating system. Note, that on Windows system the usage of '\' as directory separator would be inconvenient, because this character is also the escape character in Modelica and C Strings.

In the table below an example call to every function is given:

Function/type	Description
list(name)	List content of file or of directory.
copy(oldName, newName) copy(oldName, newName, replace=false)	Generate a copy of a file or of a directory.
move(oldName, newName) move(oldName, newName, replace=false)	Move a file or a directory to another place.
remove(name)	Remove file or directory (ignore call, if it does not exist).
removeFile(name)	Remove file (ignore call, if it does not exist)
createDirectory(name)	Create directory (if directory already exists, ignore call).
result = exist(name)	Inquire whether file or directory exists.
assertNew(name,message)	Trigger an assert, if a file or directory exists.
fullName = fullPathName(name)	Get full path name of file or directory name.
(directory, name, extension) =	Split path name in directory, file name kernel, file name

## 1858 Modelica.Utilities.Files

<code>splitPathName(name)</code>	extension.
<code>fileName = temporaryFileName()</code>	Return arbitrary name of a file that does not exist and is in a directory where access rights allow to write to this file (useful for temporary output of files).

### Package Content

Name	Description
 <code>list</code>	List content of file or directory
 <code>copy</code>	Generate a copy of a file or of a directory
 <code>move</code>	Move a file or a directory to another place
 <code>remove</code>	Remove file or directory (ignore call, if it does not exist)
 <code>removeFile</code>	Remove file (ignore call, if it does not exist)
 <code>createDirectory</code>	Create directory (if directory already exists, ignore call)
 <code>exist</code>	Inquire whether file or directory exists
 <code>assertNew</code>	Trigger an assert, if a file or directory exists
 <code>fullPathName</code>	Get full path name of file or directory name
 <code>splitPathName</code>	Split path name in directory, file name kernel, file name extension
 <code>temporaryFileName</code>	Return arbitrary name of a file that does not exist and is in a directory where access rights allow to write to this file (useful for temporary output of files)

### Modelica.Utilities.Files.list

List content of file or directory



#### Information

#### Syntax

```
Files.list(name);
```

#### Description

If "name" is a regular file, the content of the file is printed.

If "name" is a directory, the directory and file names in the "name" directory are printed in sorted order.

#### Inputs

Name	Default	Description
<code>name</code>		If name is a directory, list directory content. If it is a file, list the file content

### Modelica.Utilities.Files.copy

Generate a copy of a file or of a directory



## Information

### Syntax

```
Files.copy(oldName, newName);
Files.copy(oldName, newName, replace = true);
```

### Description

Function **copy(..)** copies a file or a directory to a new location. Via the optional argument **replace** it can be defined whether an already existing file may be replaced by the required copy.

If oldName/newName are directories, then the newName directory may exist. In such a case the content of oldName is copied into directory newName. If replace = **false** it is required that the existing files in newName are different from the existing files in oldName.

### Example

```
copy("C:/test1/directory1", "C:/test2/directory2");
    -> the content of directory1 is copied into directory2
        if "C:/test2/directory2" does not exist, it is newly
            created. If "replace=true", files in directory2
                may be overwritten by their copy
copy("test1.txt", "test2.txt")
    -> make a copy of file "test1.txt" with the name "test2.txt"
        in the current directory
```

## Inputs

Name	Default	Description
oldName		Name of file or directory to be copied
newName		Name of copy of the file or of the directory
replace	false	= true, if an existing file may be replaced by the required copy

---

## Modelica.Utilities.Files.move

Move a file or a directory to another place



## Information

### Syntax

```
Files.move(oldName, newName);
Files.move(oldName, newName, replace = true);
```

### Description

Function **move(..)** moves a file or a directory to a new location. Via the optional argument **replace** it can be defined whether an already existing file may be replaced.

If oldName/newName are directories, then the newName directory may exist. In such a case the content of oldName is moved into directory newName. If replace = **false** it is required that the existing files in newName are different from the existing files in oldName.

## 1860 Modelica.Utilities.Files.move

---

### Example

```
move("C:/test1/directory1", "C:/test2/directory2");
-> the content of directory1 is moved into directory2.
Afterwards directory1 is deleted.
if "C:/test2/directory2" does not exist, it is newly
created. If "replace=true", files in directory2
may be overwritten
move("test1.txt", "test2.txt")
-> rename file "test1.txt" into "test2.txt"
within the current directory
```

### Inputs

Name	Default	Description
oldName		Name of file or directory to be moved
newName		New name of the moved file or directory
replace	false	= true, if an existing file or directory may be replaced

---

## Modelica.Utilities.Files.remove

Remove file or directory (ignore call, if it does not exist)



### Information

### Syntax

```
Files.remove(name);
```

### Description

Removes the file or directory "name". If "name" does not exist, the function call is ignored. If "name" is a directory, first the content of the directory is removed and afterwards the directory itself.

This function is silent, i.e., it does not print a message.

### Inputs

Name	Default	Description
name		Name of file or directory to be removed

---

## Modelica.Utilities.Files.removeFile

Remove file (ignore call, if it does not exist)



### Information

### Syntax

```
Files.removeFile(fileName);
```

## Description

Removes the file "fileName". If "fileName" does not exist, the function call is ignored. If "fileName" exists but is no regular file (e.g., directory, pipe, device, etc.) an error is triggered.

This function is silent, i.e., it does not print a message.

## Inputs

Name	Default	Description
fileName		Name of file that should be removed

## Modelica.Utilities.Files.createDirectory

Create directory (if directory already exists, ignore call)



## Information

### Syntax

```
Files.createDirectory(directoryName);
```

## Description

Creates directory "directoryName". If this directory already exists, the function call is ignored. If several directories in "directoryName" do not exist, all of them are created. For example, assume that directory "E:/test1" exists and that directory "E:/test1/test2/test3" shall be created. In this case the directories "test2" in "test1" and "test3" in "test2" are created.

This function is silent, i.e., it does not print a message. In case of error (e.g., "directoryName" is an existing regular file), an assert is triggered.

## Inputs

Name	Default	Description
directoryName		Name of directory to be created (if present, ignore call)

## Modelica.Utilities.Files.exist

Inquire whether file or directory exists



## Information

### Syntax

```
result = Files.exist(name);
```

## Description

Returns true, if "name" is an existing file or directory. If this is not the case, the function returns false.

## 1862 Modelica.Utilities.Files.exist

---

### Inputs

Name	Default	Description
name		Name of file or directory

### Outputs

Name	Description
result	= true, if file or directory exists

---

## Modelica.Utilities.Files.assertNew

Trigger an assert, if a file or directory exists



### Information

### Syntax

```
Files.assertNew(name);  
Files.assertNew(name, message="This is not allowed");
```

### Description

Triggers an assert, if "name" is an existing file or directory. The error message has the following structure:

File "<name>" already exists.  
<message>

### Inputs

Name	Default	Description
name		Name of file or directory
message	"This is not allowed."	Message that should be printed after the default message in a new line

---

## Modelica.Utilities.Files fullPathName

Get full path name of file or directory name



### Information

### Syntax

```
fullName = Files.fullPathName(name);
```

### Description

Returns the full path name of a file or directory "name".

### Inputs

Name	Default	Description

name	Absolute or relative file or directory name
------	---

## Outputs

Name	Description
fullName	Full path of 'name'

## Modelica.Utilities.Files.splitPathName

Split path name in directory, file name kernel, file name extension



## Information

### Syntax

```
(directory, name, extension) = Files.splitPathName(pathName);
```

### Description

Function **splitPathName(..)** splits a path name into its parts.

### Example

```
(directory, name, extension) = Files.splitPathName("C:/user/test/input.txt")
-> directory = "C:/user/test/"
    name      = "input"
    extension = ".txt"
```

## Inputs

Name	Default	Description
pathName		Absolute or relative file or directory name

## Outputs

Name	Description
directory	Name of the directory including a trailing '/'
name	Name of the file without the extension
extension	Extension of the file name. Starts with '!'.

## Modelica.Utilities.Files.temporaryFileName

Return arbitrary name of a file that does not exist and is in a directory where access rights allow to write to this file (useful for temporary output of files)



## Information

### Syntax

```
fileName = Files.temporaryFileName();
```

## Description

Return arbitrary name of a file that does not exist and is in a directory where access rights allow to write to this file (useful for temporary output of files).

## Outputs

Name	Description
fileName	Full path name of temporary file

## Modelica.Utilities.Streams

### Read from files and write to files

## Information

### Library content

Package **Streams** contains functions to input and output strings to a message window or on files. Note that a string is interpreted and displayed as html text (e.g., with `print(..)` or `error(..)`) if it is enclosed with the Modelica html quotation, e.g.,

```
string = "<html> first line <br> second line </html>".
```

It is a quality of implementation, whether (a) all tags of html are supported or only a subset, (b) how html tags are interpreted if the output device does not allow to display formatted text.

In the table below an example call to every function is given:

Function/type	Description
<code>print(string)</code> <code>print(string,fileName)</code>	Print string "string" or vector of strings to message window or on file "fileName".
<code>stringVector = readFile(fileName)</code>	Read complete text file and return it as a vector of strings.
<code>(string, endOfFile) = readLine(fileName, lineNumber)</code>	Returns from the file the content of line lineNumber.
<code>lines = countLines(fileName)</code>	Returns the number of lines in a file.
<code>error(string)</code>	Print error message "string" to message window and cancel all actions
<code>close(fileName)</code>	Close file if it is still open. Ignore call if file is already closed or does not exist.

Use functions `scanXXX` from package `Strings` to parse a string.

If Real, Integer or Boolean values shall be printed or used in an error message, they have to be first converted to strings with the builtin operator `String(..)`. Example:

```
if x < 0 or x > 1 then
  Streams.error("x (= " + String(x) + ") has to be in the range 0 .. 1");
end if;
```

## Package Content

Name	Description
 <code>print</code>	Print string to terminal or file
 <code>readFile</code>	Read content of a file and return it in a vector of strings

 <b>readLine</b>	Reads a line of text from a file and returns it in a string
 <b>countLines</b>	Returns the number of lines in a file
 <b>error</b>	Print error message and cancel all actions
 <b>close</b>	Close file

## Modelica.Utilities.Streams.print

Print string to terminal or file



### Information

#### Syntax

```
Streams.print(string);
Streams.print(string,fileName);
```

#### Description

Function **print(..)** opens automatically the given file, if it is not yet open. If the file does not exist, it is created. If the file does exist, the given string is appended to the file. If this is not desired, call "File.remove(fileName)" before calling print ("remove(..)" is silent, if the file does not exist). The Modelica environment may close the file whenever appropriate. This can be enforced by calling **Streams.close(fileName)**. After every call of "print(..)" a "new line" is printed automatically.

#### Example

```
Streams.print("x = " + String(x));
Streams.print("y = " + String(y));
Streams.print("x = " + String(y), "mytestfile.txt");
```

#### See also

[Streams](#), [Streams.error](#), [String](#)

### Inputs

Name	Default	Description
string	""	String to be printed
fileName	""	File where to print (empty string is the terminal)

## Modelica.Utilities.Streams.readFile

Read content of a file and return it in a vector of strings



### Information

#### Syntax

```
stringVector = Streams.readFile(fileName)
```

## 1866 Modelica.Utilities.Streams.readFile

---

### Description

Function **readFile(..)** opens the given file, reads the complete content, closes the file and returns the content as a vector of strings. Lines are separated by LF or CR-LF; the returned strings do not contain the line separators.

### Inputs

Name	Default	Description
fileName		Name of the file that shall be read

### Outputs

Name	Description
stringVector[countLines(fileName)]	Content of file

---

## Modelica.Utilities.Streams.readLine

Reads a line of text from a file and returns it in a string



### Information

### Syntax

```
(string, endOfFile) = Streams.readLine(fileName, lineNumber)
```

### Description

Function **readLine(..)** opens the given file, reads enough of the content to get the requested line, and returns the line as a string. Lines are separated by LF or CR-LF; the returned string does not contain the line separator. The file might remain open after the call.

If `lineNumber > countLines(fileName)`, an empty string is returned and `endOfFile=true`. Otherwise `endOfFile=false`.

### Inputs

Name	Default	Description
fileName		Name of the file that shall be read
lineNumber		Number of line to read

### Outputs

Name	Description
string	Line of text
endOfFile	If true, end-of-file was reached when trying to read line

---

## Modelica.Utilities.Streams.countLines

Returns the number of lines in a file



## Information

### Syntax

```
numberOfLines = Streams.countLines(fileName)
```

### Description

Function **countLines(..)** opens the given file, reads the complete content, closes the file and returns the number of lines. Lines are separated by LF or CR-LF.

### Inputs

Name	Default	Description
fileName		Name of the file that shall be read

### Outputs

Name	Description
numberOfLines	Number of lines in file

## Modelica.Utilities.Streams.error

Print error message and cancel all actions



### Information

### Syntax

```
Streams.error(string);
```

### Description

Print the string "string" as error message and cancel all actions. Line breaks are characterized by "\n" in the string.

### Example

```
Streams.error("x (= " + String(x) + ") \nhas to be in the range 0 .. 1");
```

### See also

[Streams](#), [Streams.print](#), [String](#)

### Inputs

Name	Default	Description
string		String to be printed to error message window

**Modelica.Utilities.Streams.close****Close file****Information****Syntax**

```
Streams.close(fileName)
```

**Description**

Close file if it is open. Ignore call if file is already closed or does not exist.

**Inputs**

Name	Default	Description
fileName		Name of the file that shall be closed

---

**Modelica.Utilities.Strings****Operations on strings****Information****Library content**

Package **Strings** contains functions to manipulate strings.

In the table below an example call to every function is given using the **default** options.

Function	Description
len = length(string)	Returns length of string
string2 = substring(string1,startIndex,endIndex)	Returns a substring defined by start and end index
result = repeat(n) result = repeat(n,string)	Repeat a blank or a string n times.
result = compare(string1, string2)	Compares two substrings with regards to alphabetical order
identical = isEqual(string1,string2)	Determine whether two strings are identical
result = count(string,searchString)	Count the number of occurrences of a string
index = find(string,searchString)	Find first occurrence of a string in another string
index = findLast(string,searchString)	Find last occurrence of a string in another string
string2 = replace(string,searchString,replaceString)	Replace one or all occurrences of a string
stringVector2 = sort(stringVector1)	Sort vector of strings in alphabetic order
(token, index) = scanToken(string,startIndex)	Scan for a token (Real/Integer/Boolean/String/Identifier/Delimiter/NoToken)
(number, index) = scanReal(string,startIndex)	Scan for a Real constant
(number, index) = scanInteger(string,startIndex)	Scan for an Integer constant

<code>(boolean, index) = scanBoolean(string,startIndex)</code>	Scan for a Boolean constant
<code>(string2, index) = scanString(string,startIndex)</code>	Scan for a String constant
<code>(identifier, index) = scanIdentifier(string,startIndex)</code>	Scan for an identifier
<code>(delimiter, index) = scanDelimiter(string,startIndex)</code>	Scan for delimiters
<code>scanNoToken(string,startIndex)</code>	Check that remaining part of string consists solely of white space or line comments ("// ...\\n").
<code>syntaxError(string,index,message)</code>	Print a "syntax error message" as well as a string and the index at which scanning detected an error

The functions "compare", "isEqual", "count", "find", "findLast", "replace", "sort" have the optional input argument **caseSensitive** with default **true**. If **false**, the operation is carried out without taking into account whether a character is upper or lower case.

## Package Content

Name	Description
 <code>length</code>	Returns length of string
 <code>substring</code>	Returns a substring defined by start and end index
 <code>repeat</code>	Repeat a string n times
 <code>compare</code>	Compare two strings lexicographically
 <code>isEqual</code>	Determine whether two strings are identical
 <code>count</code>	Count the number of non-overlapping occurrences of a string
 <code>find</code>	Find first occurrence of a string within another string
 <code>findLast</code>	Find last occurrence of a string within another string
 <code>replace</code>	Replace non-overlapping occurrences of a string from left to right
 <code>sort</code>	Sort vector of strings in alphabetic order
 <code>scanToken</code>	Scan for the next token and return it
 <code>scanReal</code>	Scan for the next Real number and trigger an assert if not present
 <code>scanInteger</code>	Scan for the next Integer number and trigger an assert if not present
 <code>scanBoolean</code>	Scan for the next Boolean number and trigger an assert if not present
 <code>scanString</code>	Scan for the next Modelica string and trigger an assert if not present
 <code>scanIdentifier</code>	Scan for the next Identifier and trigger an assert if not present
 <code>scanDelimiter</code>	Scan for the next delimiter and trigger an assert if not present
 <code>scanNoToken</code>	Scan string and check that it contains no more token
 <code>syntaxError</code>	Print an error message, a string and the index at which scanning detected an error
 <code>Advanced</code>	Advanced scanning functions

### Modelica.Utilities.Strings.length

Returns length of string



## 1870 Modelica.Utilities.Strings.length

---

### Information

#### Syntax

```
Strings.length(string);
```

#### Description

Returns the number of characters of "string".

#### Inputs

Name	Default	Description
string		

#### Outputs

Name	Description
result	Number of characters of string

---

## Modelica.Utilities.Strings.substring

Returns a substring defined by start and end index



### Information

#### Syntax

```
string2 = Strings.substring(string, startIndex, endIndex);
```

#### Description

This function returns the substring from position startIndex up to and including position endIndex of "string". If index, startIndex, or endIndex are not correct, e.g., if endIndex > length(string), an assert is triggered.

#### Example

```
string1 := "This is line 111";
string2 := Strings.substring(string1, 9, 12); // string2 = "line"
```

#### Inputs

Name	Default	Description
string		String from which a substring is inquired
startIndex		Character position of substring begin (index=1 is first character in string)
endIndex		Character position of substring end

#### Outputs

Name	Description
result	String containing substring string[startIndex:endIndex]

**Modelica.Utilities.Strings.repeat**

Repeat a string n times

**Information****Syntax**

```
string2 = Strings.repeat(n);
string2 = Strings.repeat(n, string=" ");
```

**Description**

The first form returns a string consisting of n blanks.

The second form returns a string consisting of n substrings defined by the optional argument "string".

**Inputs**

Name	Default	Description
n	1	Number of occurrences
string	" "	String that is repeated

**Outputs**

Name	Description
repeatedString	String containing n concatenated strings

**Modelica.Utilities.Strings.compare**

Compare two strings lexicographically

**Information****Syntax**

```
result = Strings.compare(string1, string2);
result = Strings.compare(string1, string2, caseSensitive=true);
```

**Description**

Compares two strings. If the optional argument caseSensitive=false, upper case letters are treated as if they would be lower case letters. The result of the comparison is returned as:

```
result = Modelica.Utilities.Types.Compare.Less      // string1 < string2
       = Modelica.Utilities.Types.Compare.Equal    // string1 = string2
       = Modelica.Utilities.Types.Compare.Greater   // string1 > string2
```

Comparison is with regards to lexicographical order, e.g., "a" < "b";

## 1872 Modelica.Utilities.Strings.compare

---

### Inputs

Name	Default	Description
string1		
string2		
caseSensitive	true	= false, if case of letters is ignored

### Outputs

Name	Description
result	Result of comparison

---

## Modelica.Utilities.Strings isEqual

Determine whether two strings are identical



### Information

### Syntax

```
Strings.isEqual(string1, string2);
Strings.isEqual(string1, string2, caseSensitive=true);
```

### Description

Compare whether two strings are identical, optionally ignoring case.

### Inputs

Name	Default	Description
string1		
string2		
caseSensitive	true	= false, if lower and upper case are ignored for the comparison

### Outputs

Name	Description
identical	True, if string1 is identical to string2

---

## Modelica.Utilities.Strings.count

Count the number of non-overlapping occurrences of a string



### Information

### Syntax

```
Strings.count(string, searchString)
Strings.count(string, searchString, startIndex=1,
              caseSensitive=true)
```

## Description

Returns the number of non-overlapping occurrences of string "searchString" in "string". The search is started at index "startIndex" (default = 1). If the optional argument "caseSensitive" is false, for the counting it does not matter whether a letter is upper or lower case. /p>

## Inputs

Name	Default	Description
string		String that is analyzed
searchString		String that is searched for in string
startIndex	1	Start search at index startIndex
caseSensitive	true	= false, if lower and upper case are ignored for count

## Outputs

Name	Description
result	Number of occurrences of 'searchString' in 'string'

---

## Modelica.Utilities.Strings.find

Find first occurrence of a string within another string



## Information

### Syntax

```
index = Strings.find(string, searchString);
index = Strings.find(string, searchString, startIndex=1,
                      caseSensitive=true);
```

## Description

Finds first occurrence of "searchString" within "string" and return the corresponding index. Start search at index "startIndex" (default = 1). If the optional argument "caseSensitive" is false, lower and upper case are ignored for the search. If "searchString" is not found, a value of "0" is returned.

## Inputs

Name	Default	Description
string		String that is analyzed
searchString		String that is searched for in string
startIndex	1	Start search at index startIndex
caseSensitive	true	= false, if lower and upper case are ignored for the search

## Outputs

Name	Description
index	Index of the beginning of the first occurrence of 'searchString' within 'string', or zero if not present

**Modelica.Utilities.Strings.findLast**

Find last occurrence of a string within another string

**Information****Syntax**

```
index = Strings.findLast(string, searchString);
index = Strings.findLast(string, searchString,
                      startIndex=length(string),
                      caseSensitive=true,
```

**Description**

Finds first occurrence of "searchString" within "string" when searching from the last character of "string" backwards, and return the corresponding index. Start search at index "startIndex" (default = length(string)). If the optional argument "caseSensitive" is false, lower and upper case are ignored for the search. If "searchString" is not found, a value of "0" is returned.

**Inputs**

Name	Default	Description
string		String that is analyzed
searchString		String that is searched for in string
startIndex	0	Start search at index startIndex. If startIndex = 0, start at length(string)
caseSensitive	true	= false, if lower and upper case are ignored for the search

**Outputs**

Name	Description
index	Index of the beginning of the last occurrence of 'searchString' within 'string', or zero if not present

**Modelica.Utilities.Strings.replace**

Replace non-overlapping occurrences of a string from left to right

**Information****Syntax**

```
Strings.replace(string, searchString, replaceString);
Strings.replace(string, searchString, replaceString,
               startIndex=1, replaceAll=true, caseSensitive=true);
```

**Description**

Search in "string" for "searchString" and replace the found substring by "replaceString".

- The search starts at the first character of "string", or at character position "startIndex", if this optional argument is provided.
- If the optional argument "replaceAll" is **true** (default), all occurrences of "searchString" are replaced. If the argument is **false**, only the first occurrence is replaced.

- The search for "searchString" distinguishes upper and lower case letters. If the optional argument "caseSensitive" is **false**, the search ignores whether letters are upper or lower case.

The function returns the "string" with the performed replacements.

## Inputs

Name	Default	Description
string		String to be modified
searchString		Replace non-overlapping occurrences of 'searchString' in 'string' with 'replaceString'
replaceString		String that replaces 'searchString' in 'string'
startIndex	1	Start search at index startIndex
replaceAll	true	if false, replace only the first occurrence, otherwise all occurrences
caseSensitive	true	= false, if lower and upper case are ignored when searching for searchString

## Outputs

Name	Description
result	Resultant string of replacement operation

## Modelica.Utilities.Strings.sort

Sort vector of strings in alphabetic order



## Information

### Syntax

```
stringVector2 = Streams.sort(stringVector1);
stringVector2 = Streams.sort(stringVector1, caseSensitive=true);
```

### Description

Function **sort(..)** sorts a string vector `stringVector1` in lexicographical order and returns the result in `stringVector2`. If the optional argument "caseSensitive" is **false**, lower and upper case letters are not distinguished.

### Example

```
s1 = {"force", "angle", "pressure"};
s2 = Strings.sort(s1);
-> s2 = {"angle", "force", "pressure"};
```

## Inputs

Name	Default	Description
stringVector1[:]		vector of strings
caseSensitive	true	= false, if lower and upper case are ignored when comparing elements of stringVector1

## Outputs

Name	Description
stringVector2[size(stringVector1, 1)]	string1 sorted in alphabetical order

## Modelica.Utilities.Strings.scanToken

Scan for the next token and return it



## Information

### Syntax

```
(token, nextIndex) = Strings.scanToken(string, startIndex,  
unsigned=false);
```

### Description

Function **scanToken** scans the string starting at index "startIndex" and returns the next token, as well as the index directly after the token. The returned token is a record that holds the type of the token and the value of the token:

token.tokenType	Type of the token, see below
token.real	Real value if tokenType == TokenType.RealToken
token.integer	Integer value if tokenType == TokenType.IntegerToken
token.boolean	Boolean value if tokenType == TokenType.BooleanToken
token.string	String value if tokenType == TokenType.StringToken/IdentifierToken/DelimiterToken

Variable `token.tokenType` is an enumeration (emulated as a package with constants) that can have the following values:

```
import T = Modelica.Utilities.Types.TokenType;
```

T.RealToken	Modelica Real literal (e.g., 1.23e-4)
T.IntegerToken	Modelica Integer literal (e.g., 123)
T.BooleanToken	Modelica Boolean literal (e.g., false)
T.StringToken	Modelica String literal (e.g., "string 123")
T.IdentifierToken	Modelica identifier (e.g., "force_a")
T.DelimiterToken	any character without white space that does not appear as first character in the tokens above (e.g., "&")
T.NoToken	White space, line comments and no other token until the end of the string

Modelica line comments ("// ... end-of-line/end-of-string") as well as white space is ignored. If "unsigned=true", a Real or Integer literal is not allowed to start with a "+" or "-" sign.

### Example

```
import Modelica.Utilities.Strings.*;
import T = Modelica.Utilities.Types.TokenType;
(token, index) := scanToken(string);
if token.tokenType == T.RealToken then
    realValue := token.real;
elseif token.tokenType == T.IntegerToken then
    integerValue := token.integer;
```

```

elseif token.tokenType == T.BooleanToken then
    booleanValue := token.boolean;
elseif token.tokenType == T.Identifier then
    name := token.string;
else
    syntaxError(string,index,"Expected Real, Integer, Boolean or
identifier token");
end if;

```

## Inputs

Name	Default	Description
string		String to be scanned
startIndex	1	Start scanning of string at character startIndex
unsigned	false	= true, if Real and Integer tokens shall not start with a sign

## Outputs

Name	Description
token	Scanned token
nextIndex	Index of character after the found token; = 0, if NoToken

---

## Modelica.Utilities.Strings.scanReal



Scan for the next Real number and trigger an assert if not present

## Information

### Syntax

```

number = Strings.scanReal(string);
(number, nextIndex) = Strings.scanReal(string, startIndex=1,
                                         unsigned=false,
                                         message="" );

```

### Description

The first form, "scanReal(string)", scans "string" for a Real number with leading white space and returns the value.

The second form, "scanReal(string,startIndex,unsigned)", scans the string starting at index "startIndex", checks whether the next token is a Real literal and returns its value as a Real number, as well as the index directly after the Real number. If the optional argument "unsigned" is **true**, the real number shall not have a leading "+" or "-" sign.

If the required Real number with leading white space is not present in "string", an assert is triggered.

## Inputs

Name	Default	Description
string		String to be scanned
startIndex	1	Start scanning of string at character startIndex
unsigned	false	= true, if Real token shall not start with a sign

## 1878 Modelica.Utilities.Strings.scanReal

---

message	""	Message used in error message if scan is not successful
---------	----	---

### Outputs

Name	Description
number	Value of real number
nextIndex	index of character after the found number

---

## Modelica.Utilities.Strings.scanInteger

Scan for the next Integer number and trigger an assert if not present



### Information

### Syntax

```
number = Strings.scanInteger(string);
(number, nextIndex) = Strings.scanInteger(string, startIndex=1,
                                         unsigned=false,
                                         message="");
```

### Description

Function **scanInteger** scans the string starting at index "startIndex", checks whether the next token is an Integer literal and returns its value as an Integer number, as well as the index directly after the Integer number. An assert is triggered, if the scanned string does not contain an Integer literal with optional leading white space.

### Inputs

Name	Default	Description
string		String to be scanned
startIndex	1	Start scanning of string at character startIndex
unsigned	false	= true, if Integer token shall not start with a sign
message	""	Message used in error message if scan is not successful

### Outputs

Name	Description
number	Value of Integer number
nextIndex	Index of character after the found number

---

## Modelica.Utilities.Strings.scanBoolean

Scan for the next Boolean number and trigger an assert if not present



### Information

### Syntax

```
number = Strings.scanBoolean(string);
```

```
(number, nextIndex) = Strings.scanBoolean(string, startIndex=1,
message="");
```

## Description

Function **scanBoolean** scans the string starting at index "startIndex", checks whether the next token is a Boolean literal (i.e., is either the string "false" or "true", if converted to lower case letters) and returns its value as a Boolean number, as well as the index directly after the Boolean number. An assert is triggered, if the scanned string does not contain a Boolean literal with optional leading white space.

## Inputs

Name	Default	Description
string		String to be scanned
startIndex	1	Start scanning of string at character startIndex
message	""	Message used in error message if scan is not successful

## Outputs

Name	Description
number	Value of Boolean
nextIndex	Index of character after the found number

## Modelica.Utilities.Strings.scanString



Scan for the next Modelica string and trigger an assert if not present

## Information

### Syntax

```
string2 = Strings.scanString(string);
(string2, nextIndex) = Strings.scanString(string, startIndex=1,
message="");
```

## Description

Function **scanString** scans the string starting at index "startIndex", checks whether the next token is a String literal and returns its value as a String, as well as the index directly after the String. An assert is triggered, if the scanned string does not contain a String literal with optional leading white space.

## Inputs

Name	Default	Description
string		String to be scanned
startIndex	1	Start scanning of string at character startIndex
message	""	Message used in error message if scan is not successful

## Outputs

Name	Description

## 1880 Modelica.Utilities.Strings.scanString

result	Value of string
nextIndex	Index of character after the found string

### Modelica.Utilities.Strings.scanIdentifier

Scan for the next Identifier and trigger an assert if not present



#### Information

#### Syntax

```
identifier = Strings.scanIdentifier(string);
(identifier, nextIndex) = Strings.scanIdentifier(string, startIndex=1,
message="");
```

#### Description

Function **scanIdentifier** scans the string starting at index "startIndex", checks whether the next token is an Identifier and returns its value as a string, as well as the index directly after the Identifier. An assert is triggered, if the scanned string does not contain an Identifier with optional leading white space.

#### Inputs

Name	Default	Description
string		String to be scanned
startIndex	1	Start scanning of identifier at character startIndex
message	""	Message used in error message if scan is not successful

#### Outputs

Name	Description
identifier	Value of Identifier
nextIndex	Index of character after the found identifier

### Modelica.Utilities.Strings.scanDelimiter

Scan for the next delimiter and trigger an assert if not present



#### Information

#### Syntax

```
delimiter = Strings.scanDelimiter(string);
(delimiter, nextIndex) = Strings.scanDelimiter(string, startIndex=1,
requiredDelimiters={","},
message="");
```

#### Description

Function **scanDelimiter** scans the string starting at index "startIndex", checks whether the next token is a delimiter string and returns its value as a string, as well as the index directly after the delimiter. An assert is

triggered, if the scanned string does not contain a delimiter out of the list of requiredDelimiters. Input argument requiredDelimiters is a vector of strings. The elements may have any length, including length 0. If an element of the requiredDelimiters is zero, white space is treated as delimiter. The function returns delimiter="" and nextIndex is the index of the first non white space character.

## Inputs

Name	Default	Description
string		String to be scanned
startIndex	1	Start scanning of delimiters at character startIndex
requiredDelimiters[:]	{" , "}	Delimiters that are searched
message	""	Message used in error message if scan is not successful

## Outputs

Name	Description
delimiter	Found delimiter
nextIndex	Index of character after the found delimiter

## Modelica.Utilities.Strings.scanNoToken

Scan string and check that it contains no more token



## Information

### Syntax

```
Strings.scanNoToken(string, startIndex=1, message="");
```

### Description

Function **scanNoToken** scans the string starting at index "startIndex" and checks whether there is no more token in the string. An assert is triggered if this is not the case, using the "message" argument as additional explanation in the error text.

## Inputs

Name	Default	Description
string		String to be scanned
startIndex	1	Start scanning of string at character startIndex
message	""	Message used in error message if scan is not successful

## Modelica.Utilities.Strings.syntaxError

Print an error message, a string and the index at which scanning detected an error



## Information

### Syntax

```
Strings.syntaxError(string, index, message);
```

### Description

Function **syntaxError** prints an error message in the following form:

```
Syntax error at column <index> of
<string>
  ^          // shows character that is wrong
<message>
```

where the strings withing <..> are the actual values of the input arguments of the function.

If the given string is too long, only a relevant part of the string is printed.

## Inputs

Name	Default	Description
string		String that has an error at position index
index		Index of string at which scanning detected an error
message	""	String printed at end of error message

---

## Modelica.Utilities.Strings.Advanced

### Advanced scanning functions

## Information

### Library content

Package **Strings.Advanced** contains basic scanning functions. These functions should be **not called** directly, because it is much simpler to utilize the higher level functions "Strings.scanXXX". The functions of the "Strings.Advanced" library provide the basic interface in order to implement the higher level functions in package "Strings".

Library "Advanced" provides the following functions:

```
(nextIndex, realNumber)      = scanReal           (string, startIndex,
unsigned=false);
(nextIndex, integerNumber)   = scanInteger        (string, startIndex,
unsigned=false);
(nextIndex, string2)         = scanString         (string, startIndex);
(nextIndex, identifier)     = scanIdentifier    (string, startIndex);
nextIndex                   = skipWhiteSpace   (string, startIndex);
nextIndex                   = skipLineComments (string, startIndex);
```

All functions perform the following actions:

1. Scanning starts at character position "startIndex" of "string" (startIndex has a default of 1).
2. First, white space is skipped, such as blanks (" "), tabs ("\t"), or newline ("\n")
3. Afterwards, the required token is scanned.

4. If successful, on return nextIndex = index of character directly after the found token and the token value is returned as second output argument.  
If not successful, on return nextIndex = startIndex.

The following additional rules apply for the scanning:

- Function **scanReal**:  
Scans a full number including one optional leading "+" or "-" (if unsigned=false) according to the Modelica grammar. For example, "+1.23e-5", "0.123" are Real numbers, but ".1" is not. Note, an Integer number, such as "123" is also treated as a Real number.
- Function **scanInteger**:  
Scans an Integer number including one optional leading "+" or "-" (if unsigned=false) according to the Modelica (and C/C++) grammar. For example, "+123", "20" are Integer numbers. Note, a Real number, such as "123.4" is not an Integer and scanInteger returns nextIndex = startIndex.
- Function **scanString**:  
Scans a String according to the Modelica (and C/C++) grammar, e.g., "This is a "string"" is a valid string token.
- Function **scanIdentifier**:  
Scans a Modelica identifier, i.e., the identifier starts either with a letter, followed by letters, digits or "\_". For example, "w\_rel", "T12".
- Function **skipWhiteSpace**  
Skips white space and Modelica (C/C++) line comments iteratively. A line comment starts with "//" and ends either with an end-of-line ("\n") or the end of the "string".

## Package Content

Name	Description
(f) <b>scanReal</b>	Scans a signed real number
(f) <b>scanInteger</b>	Scans signed integer number
(f) <b>scanString</b>	
(f) <b>scanIdentifier</b>	Scans simple identifiers
(f) <b>skipWhiteSpace</b>	Scans white space
(f) <b>skipLineComments</b>	Scans comments and white space

---

## Modelica.Utilities.Strings.Advanced.scanReal

Scans a signed real number



### Information

#### Syntax

```
(nextIndex, realNumber) = scanReal(string, startIndex=1,  
unsigned=false);
```

#### Description

Starts scanning of "string" at position "startIndex". First skips white space and scans afterwards a number of

## 1884 Modelica.Utilities.Strings.Advanced.scanReal

---

type Real with an optional sign according to the Modelica grammar:

```
real      ::= [sign] unsigned [fraction] [exponent]
sign     ::= '+' | '-'
unsigned ::= digit [unsigned]
fraction ::= '.' [unsigned]
exponent ::= ('e' | 'E') [sign] unsigned
digit   ::= '0' | '1' | '2' | '3' | '4' | '5' | '6' | '7' | '8' | '9'
```

If successful, the function returns nextIndex = index of character directly after the found real number, as well as the value in the second output argument.

If not successful, on return nextIndex = startIndex and the second output argument is zero.

If the optional argument "unsigned" is **true**, the number shall not start with '+' or '-'. The default of "unsigned" is **false**.

### See also

[Strings.Advanced](#).

### Inputs

Name	Default	Description
string		
startIndex	1	Index where scanning starts
unsigned	false	= true, if number shall not start with '+' or '-'

### Outputs

Name	Description
nextIndex	Index after the found token (success=true) or index at which scanning failed (success=false)
number	Value of Real number

---

## Modelica.Utilities.Strings.Advanced.scanInteger

Scans signed integer number



### Information

### Syntax

```
(nextIndex, integerNumber) = scanInteger(string, startIndex=1,  
unsigned=false);
```

### Description

Starts scanning of "string" at position "startIndex". First skips white space and scans afterwards a signed number of type Integer. An Integer starts with an optional '+' or '-', immediately followed by a non-empty sequence of digits.

If successful, the function returns nextIndex = index of character directly after the found Integer number, as well as the Integer value in the second output argument.

If not successful, on return nextIndex = startIndex and the second output argument is zero.

Note, a Real number, such as "123.4", is not treated as an Integer number and scanInteger will return nextIndex = startIndex in this case.

If the optional argument "unsigned" is **true**, the number shall not start with '+' or '-'. The default of "unsigned" is **false**.

### See also

[Strings.Advanced](#).

### Inputs

Name	Default	Description
string		
startIndex	1	
unsigned	false	= true, if number shall not start with '+' or '-'

### Outputs

Name	Description
nextIndex	Index after the found token (success=true) or index at which scanning failed (success=false)
number	Value of Integer number

## Modelica.Utilities.Strings.Advanced.scanString



### Information

### Syntax

```
(nextIndex, string2) = scanString(string, startIndex=1);
```

### Description

Starts scanning of "string" at position "startIndex". First skips white space and scans afterwards a string according to the Modelica grammar, i.e., a string enclosed in double quotes.

If successful, the function returns nextIndex = index of character directly after the found string, as well as the string value in the second output argument.

If not successful, on return nextIndex = startIndex and the second output argument is an empty string.

### See also

[Strings.Advanced](#).

### Inputs

Name	Default	Description
string		
startIndex	1	Index where scanning starts

## 1886 Modelica.Utilities.Strings.Advanced.scanString

### Outputs

Name	Description
nextIndex	Index after the found token (success=true) or index at which scanning failed (success=false)
string2	Value of String token

## Modelica.Utilities.Strings.Advanced.scanIdentifier

Scans simple identifiers



### Information

### Syntax

```
(nextIndex, identifier) = scanIdentifier(string, startIndex=1);
```

### Description

Starts scanning of "string" at position "startIndex". First skips white space and scans afterwards a Modelica identifier, i.e., a sequence of characters starting with a letter ("a".."z" or "A".."Z") followed by letters, digits or underscores ("\_").

If successful, the function returns nextIndex = index of character directly after the found identifier, as well as the identifier as string in the second output argument.

If not successful, on return nextIndex = startIndex and the second output argument is an empty string.

### See also

[Strings.Advanced](#).

### Inputs

Name	Default	Description
string		
startIndex	1	Index where scanning starts

### Outputs

Name	Description
nextIndex	Index after the found token (success=true) or index at which scanning failed (success=false)
identifier	Value of identifier token

## Modelica.Utilities.Strings.Advanced.skipWhiteSpace

Scans white space



### Information

### Syntax

```
nextIndex = skipWhiteSpace(string, startIndex);
```

## Description

Starts scanning of "string" at position "startIndex" and skips white space. The function returns nextIndex = index of character of the first non white space character.

## See also

[Strings.Advanced.](#)

## Inputs

Name	Default	Description
string		
startIndex	1	

## Outputs

Name	Description
nextIndex	

## Modelica.Utilities.Strings.Advanced.skipLineComments

Scans comments and white space



## Information

### Syntax

```
nextIndex = skipLineComments(string, startIndex);
```

## Description

Starts scanning of "string" at position "startIndex". First skips white space and scans afterwards a Modelica (C/C++) line comment, i.e., a sequence of characters that starts with "//" and ends with an end-of-line "\n" or with the end of the string. If end-of-line is reached, the function continues to skip white space and scanning of line comments until end-of-string is reached, or another token is detected.

If successful, the function returns nextIndex = index of character directly after the found line comment.

If not successful, on return nextIndex = startIndex.

## See also

[Strings.Advanced.](#)

## Inputs

Name	Default	Description
string		
startIndex	1	

## Outputs

Name	Description

## 1888 Modelica.Utilities.Strings.Advanced.skipLineComments

---

nextIndex

---

## Modelica.Utilities.System

### Interaction with environment

#### Information

This package contains functions to interact with the environment.

#### Package Content

Name	Description
(f) <a href="#">getWorkDirectory</a>	Get full path name of work directory
(f) <a href="#">setWorkDirectory</a>	Set work directory
(f) <a href="#">getEnvironmentVariable</a>	Get content of environment variable
(f) <a href="#">setEnvironmentVariable</a>	Set content of local environment variable
(f) <a href="#">command</a>	Execute command in default shell
(f) <a href="#">exit</a>	Terminate execution of Modelica environment

---

## Modelica.Utilities.System.getWorkDirectory

Get full path name of work directory



#### Information

#### Outputs

Name	Description
directory	Full path name of work directory

---

## Modelica.Utilities.System.setWorkDirectory

Set work directory



#### Information

#### Inputs

Name	Default	Description
directory		New work directory

---

## Modelica.Utilities.System.getEnvironmentVariable

Get content of environment variable



## Information

### Inputs

Name	Default	Description
name		Name of environment variable
convertToSlash	false	True, if native directory separators in 'result' shall be changed to '/'

### Outputs

Name	Description
content	Content of environment variable (empty, if not existent)
exist	= true, if environment variable exists; = false, if it does not exist

## Modelica.Utilities.System.setEnvironmentVariable



Set content of local environment variable

## Information

### Inputs

Name	Default	Description
name		Name of environment variable
content		Value of the environment variable
convertFromSlash	false	True, if '/' in content shall be changed to the native directory separator

## Modelica.Utilities.System.command



Execute command in default shell

## Information

### Inputs

Name	Default	Description
string		String to be passed to shell

### Outputs

Name	Description
result	Return value from command (depends on environment)

## Modelica.Utilities.System.exit



Terminate execution of Modelica environment

## 1890 Modelica.Utilities.System.exit

---

### Inputs

Name	Default	Description
status	0	Result to be returned by environment (0 means success)

---

## Modelica.Utilities.Types

Type definitions used in package Modelica.Utilities

### Information

This package contains type definitions used in Modelica.Utilities.

### Package Content

Name	Description
 Compare	Type and constants to compare two strings, as temporary solution until enumerations are available
 FileType	Type and constants to describe the type of a file, as temporary solution until enumerations are available
 TokenType	Type and constants for token types, as temporary solution until enumerations are available
 TokenValue	Value of token

---

## Modelica.Utilities.Types.Compare

Type and constants to compare two strings, as temporary solution until enumerations are available

### Information

#### Syntax

```
TokenType.Temp tokenType = TokenType.RealToken;
```

#### Description

Package TokenType is an emulation of the following enumeration

```
enumeration TokenType = {RealToken, IntegerToken, BooleanToken,  
StringToken, IdentifierToken, DelimiterToken,  
NoToken};
```

since enumerations are not yet supported in available Modelica tools.

### Package Content

Name	Description
Less=1	
Equal=2	
Greater=3	
Type	

## Types and constants

```
constant Integer Less=1;
constant Integer Equal=2;
constant Integer Greater=3;
type Type = Integer;
```

## Modelica.Utilities.Types.Compare.Type

### Modelica.Utilities.Types.FileType

Type and constants to describe the type of a file, as temporary solution until enumerations are available

#### Information

#### Syntax

```
FileType.Type fileType = FileType.RegularFile;
```

#### Description

Package **FileType** is an emulation of the following enumeration

```
enumeration FileType = {NoFile, RegularFile, Directory, SpecialFile};
```

since enumerations are not yet supported in available Modelica tools.

#### Package Content

Name	Description
NoFile=1	no file exists
RegularFile=2	regular file
Directory=3	directory
SpecialFile=4	pipe, FIFO, device, etc.
Type	

## Types and constants

```
constant Integer NoFile=1 "no file exists";
constant Integer RegularFile=2 "regular file";
constant Integer Directory=3 "directory";
constant Integer SpecialFile=4 "pipe, FIFO, device, etc.;"
```

## 1892 Modelica.Utilities.Types.FileType

---

```
type Type = Integer;
```

---

## Modelica.Utilities.Types.FileType.Type

---

### Modelica.Utilities.Types.TokenType

Type and constants for token types, as temporary solution until enumerations are available

#### Information

#### Syntax

```
TokenType.Temp tokenType = TokenType.RealToken;
```

#### Description

Package **TokenType** is an emulation of the following enumeration

```
enumeration TokenType = {RealToken, IntegerToken, BooleanToken,
StringToken, IdentifierToken, DelimiterToken,
NoToken};
```

since enumerations are not yet supported in available Modelica tools.

#### Package Content

Name	Description
RealToken=1	
IntegerToken=2	
BooleanToken=3	
StringToken=4	
IdentifierToken=5	
DelimiterToken=6	
NoToken=7	
Type	

#### Types and constants

```
constant Integer RealToken=1;

constant Integer IntegerToken=2;

constant Integer BooleanToken=3;

constant Integer StringToken=4;

constant Integer IdentifierToken=5;

constant Integer DelimiterToken=6;
```

```
constant Integer NoToken=7;  
type Type = Integer;
```

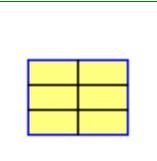
---

## **Modelica.Utilities.Types.TokenType.Type**

---

### **Modelica.Utilities.Types.TokenValue**

**Value of token**



### **Information**

---

*HTML-documentation generated by Dymola Fri Aug 17 21:37:04 2007.*