Semester Project - Spring 2011

Modeling and Simulation of Hydraulic Systems in Dymola/Modelica by Means of Bond Graphs

Author:
Raphael Sebastian Theurillat

Supervisor:
Prof. Dr. François E. Cellier

Modeling and Simulation Research Group
Departement Informatik
ETH Zurich

July 25, 2011

Abstract

In this semester project a library for hydraulic systems based on bond graphs in dymola/-modelica is presented. The models in the library are based on the bond graph library [6]. The models contain a hydraulic part with all basic hydraulic models. Furthermore they contain thermal ports that give out the produced heat. Three examples are provided that illustrate the use of the library. Fluid properties which are decoupled from the models are provided, the viscosity and bulk modulus of the fluid are calculated dependent on pressure and temperature in the model. The individual models are documented and a users guide is provided.

Acknowledgements

I would like to thank my supervisor Prof. François E. Cellier for his encouragement, guidance and support during this project.

I am grateful for my parents who rendered it possible for me to attend my studies.

Raphael Theurillat

Contents

Chapter 1

Introduction

1.1 Situation

Dymola is an object oriented modeling and simulation software. Through its graphical interface it permits modeling highly complex systems in a way that is easily maintainable [3].
A suitable way for keeping the graphical interface as long as possible is modeling with bond graphs. Bond graphs allow to represent power flows in physical systems, independently of the actual domain [6].
Hydraulic systems transmit energy via a fluid, the supplied power is transformed to the specific energy of the mass flow and can be retransformed to mechanical energy [7].

1.2 Goal

The goal of this semester project is to offer a hydraulic library in Dymola based on bondgraphs. This library should be at least as powerfull as the HyLibLight package offered in Dymola. Additionally the hydraulic library should contain thermal ports which handle the generation of heat. It should be possible to choose whether these thermal ports are externally connected or not.
Test models shall be included to show how to use the new library.
The models should be documented in html files and a users guide shall be generated.

Chapter 2

Hydraulic Modeling

2.1 Modeling in Thermodynamics

Hydraulics is a part of Thermodynamics. With general thermodynamic modeling it is possible to represent hydraulic flows. The way to model thermodynamics with bondgraphs was shown by Cellier and Greifeneder 2008[5]. This paper describes the internal energy of matter U as:

$$U = T \cdot S - p \cdot V + g \cdot M \tag{2.1}$$

With T temperature, S the entropy, p the pressure, V the volume, g the Gibbs potential and M the mass.
The internal energy is not the only energy to be considered in hydraulics. For moving masses also the kinetic and the potential gravitational energy play an important role:

$$E_{kin} = \frac{1}{2} \cdot M \cdot v^2 \tag{2.2}$$

$$E_{pot} = M \cdot g_m \cdot h \tag{2.3}$$

where m denotes the mass, v the velocity of the particle, g_m the gravitational acceleration and h the height of the particle over a reference. The total energy of a control volume can be calculated by [10].

$$E_{tot} = U + E_{kin} + E_{pot} \tag{2.4}$$

Hydraulic systems have a quasi-onedimensional flow. The direction of flow is into the direction of the fluidic line. As a consequence the velocity is a scalar quantity.
For representation in Bondgraphs the flow of energy, the power, is needed [5]:

$$\frac{dU}{dt} = T \cdot \dot{S} - p \cdot q + g \cdot \dot{M} \tag{2.5}$$

where q is the volumetric flow of matter.
The flow of kinetic and potential energy can be derived simply by differentiation:

$$\frac{dE_{kin}}{dt} = \frac{1}{2} v^2 \cdot \dot{M} + M\dot{v} \cdot v \tag{2.6}$$

$$\frac{dE_{pot}}{dt} = M g_m \cdot \dot{h} + g_m h \cdot \dot{M} \tag{2.7}$$

Now it is possible to write down the total energy flow of matter:

$$\frac{dE_{tot}}{dt} = T \cdot \dot{S} - p \cdot q + (g + \frac{1}{2}v^2 + g_m h) \cdot \dot{M} + M g_m \cdot \dot{h} + M\dot{v} \cdot v \tag{2.8}$$

In a bondgraphic representation this would lead to a multi-bond library with 5 sub-bonds. However, since only hydraulic systems are modeled, some simplifications can be made, as shown in the next section.

2.2 Modeling of hydraulic systems

In hydraulic systems it is possible to reduce the complexity of the energy flows drastically. Gravitational potential energy is considered to change very little in common hydraulic systems, so the terms with h and $\dot{h}$ can be neglected (after setting reference height to h).

$$\frac{dE_{tot}}{dt} = T \cdot \dot{S} - p \cdot q + (g + \frac{1}{2}v^2) \cdot \dot{M} + M\dot{v} \cdot v \tag{2.9}$$

Furthermore hydraulic systems are considered incompressible which means that the density ρ stays constant [9]. This leads to the possibility to calculate the mass flow rate out of the volumetric flow rate:

$$d\dot{M} = \rho \cdot q \tag{2.10}$$

which leads to a simplified representation of the total energy. Another simplification can be done by assuming a onedimensional flow in a tube. If we know the cross-section A of that tube, the velocity and the volumetric flow rate can be coupled:

$$v = \frac{q}{A} \tag{2.11}$$

A further simplification concerns the chemical part. Since there is only one fluid in the system, the gibbs potential is not needed in hydraulic systems, and can therefore be neglected.

$$\frac{dE_{tot}}{dt} = T \cdot \dot{S} + (-p + \frac{1}{2}\rho v^2 + \frac{V\rho}{A}\dot{v}) \cdot q \tag{2.12}$$

It can be seen that hydraulic systems can be modeled with two pairs of variables: The temperature T and the entropy flow $\dot{S}$ plus the dynamic pressure and volumetric flow q.

Chapter 3

Implementation

3.1 Bonds and Connectors

As shown in the previous section we consider four variables to model bondgraphs for hydraulic systems.

Figure 3.1: The main bond-types used. The pressure bond contains as effort variable the dynamic pressure. The flow variable is the volumetric flow. The thermo bond contains as effort variable the temperature and as flow variable the entropy flow

For the thermal part the connectors from the *Bondlib* for thermal systems are taken, since they contain exactly the values we need. For the hydraulic part a connector was developed which contains two variables: one with units of pressure and one with units of a volumetric flow:

```
connector Port "Model of Hydraulic Port"
  Modelica.SIunits.AbsolutePressure p "pressure at port";
  flow Modelica.SIunits.VolumeFlowRate q "flow rate through port";

end Port;
```

3.2 Hydraulic Models

The basic part of the *Hydraulic* library are the hydraulic models. There are four different types of models:

1. Pumps - models with sources or energy conversions from mechanics
2. Components - passive parts of hydraulic systems, such as resistances and chambers
3. Sensors - sensor elements
4. Controllers - models of controllers that control the flow

These models are made with bond graphs in an object oriented way.

3.2.1 Pumps

In this part different sorts of sources are presented. The models *Tank*, *PressureSource*, *FlowSource* and *ForcedFlow* describe sources of flow or pressure with one or two ports. The non-ideal models have additional resistances and leaks as they occur in reality. The models *Turbine* and *TransTurbine* are models that contain rotational and tranlational mechanical connectors. They are models of turbines or pumps with a constant relation between the volumetric flow and the velocity. The model *Piston* and *TwoChamberPiston* are models of pistons with mechanical connectors. The hydraulic pressure moves the piston in or out.

3.2.2 Components

There are three different kinds of passive elements. Resistive elements couple the pressure and the volumetric flow. Inductive elements couple the derivative of the volume flow with the pressure and conductive elements couple the derivative of the pressure with the volumetric flow [4].

LamRes Model of a laminar resistance. The conductance is constant.

$$q = G \cdot \Delta p \tag{3.1}$$

With q the volumetric flow, Δp the pressure drop over the element and G the conductance

TurbRes Model of a turbulent resistance. The resistance has a linear and a quadratic term. The pressure difference is calculated by the following equation:

$$\Delta p = a \cdot q + b \cdot sgn(q) \cdot q^2 \tag{3.2}$$

The parameters a and b can be calculated in three different ways. The parameters k_1 and k_2 describe the relation between the pressure difference and the velocity w of the fluid [8].

$$\Delta p = k_1 w + k_2 \cdot sgn(w) \cdot w^2 \tag{3.3}$$

The resistance coefficient ζ is a coefficient between the specific kinetic energy of the flow and the pressure difference [8].

$$\zeta = \frac{\Delta p}{\rho w^2/2} \tag{3.4}$$

If the parameter λ is used it defines a friction coefficient[8].

$$\lambda = \frac{\zeta}{L/D} = \frac{\Delta p D}{L \rho w^2/2} \tag{3.5}$$

Line Model of a hydraulic line. It contains a laminar and a turbulent part. The laminar part calculates the flow by the Hagen-Poiseuille formula for a laminar flow[8]:

$$\lambda = \frac{64}{Re} \tag{3.6}$$

If a critical Re-number is reached, it switches to turbulent flow which is calculated by the Altshul approximate formula[1]:

$$\lambda = 0.11(\bar{\Delta} + \frac{68}{Re})^{0.25} \tag{3.7}$$

Where $\bar{\Delta}$ denotes the ratio of the average height of asperities to the tube diameter.

Chamber Chamber is a model which describes the capacitive part of hydraulics. It is used to calculate compressibility effects. The formula used is:

$$\frac{V}{\beta} \cdot \frac{dp}{dt} = q \tag{3.8}$$

where V denotes the Volume and β the bulk modulus of the hydraulic oil.

Inertia This model describes inertial effects in hydraulics. The element behaves after the formula:

$$\frac{L\rho}{A} \cdot \frac{dp}{dt} = q \tag{3.9}$$

with L the lenth of the tube, ρ the density of the fluid and A the cross section.

LineWithKinetics This model includes a line, a chamber and a inertia. That means it has a resistive, a conductive and an inertial part. It can be used for short lines.

LongLine In longline, multiple linewithkinetics are put together to describe a long line. This model calculates the effects that occur in long hydraulic lines, such as time delays in pressure jumps.

3.2.3 Sensors

There are two kind of sensors: the pressure sensor which senses the absolute pressure in the fluid and the flow sensor which senses the volumetric flow of oil. They are ideal sensors and do not have an influence on the flow and present measure in an exact way.

3.2.4 Controllers

In the package *Controllers* different controllers are provided. Model *SimpleValve* describes an ideal valve that can either be completely closed or completely opened. *ThreeTwoValve* is a model for the often used 3-2-Valve. *CheckValve* is a model for a turbulent controller. The diameter of the valve can be set. *Checkvalve* provides a controlleable laminar resistance.

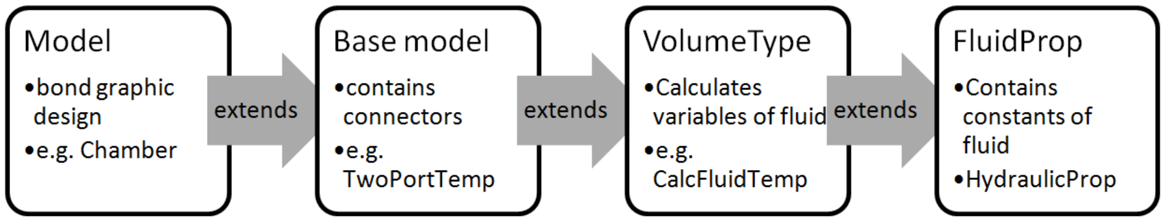


Figure 3.2: The way fluid properties are included in the models

3.3 Heat Properties

The models in the *Hydraulic* library have a *heatPort*. In resistive elements, the produced heat by the resistance is given out. In capacitive elements, there is a thermal model included, that contains the energy saved by the fluid. All models use the temperature in the heat port to calculate fluid properties.
It is important to note that convective flows are not directly modeled in the *Hydraulic* library. The produced heat is given out through the heatport and capacitive elements store heat. The heat flows must be added seperately if wished.

3.4 Fluid Properties

Fluid properties are very important for hydraulic systems. The properties have to be interchangeable, so that different fluids can be simulated easily. To achieve this task a similar solution to the one in *HyLibLight* was taken[2]. The constant fluid properties are saved in the model *FluidProp*. These Properties are used by the model *VolumeType* which computes the variable fluid properties.
The basic interface models extend *VolumeType* to have access to the variables and constants of the fluid.
The flowchart of the extensions can be seen in figure 3.2.

3.4.1 Volumetype

The models in *VolumeType* calculate the kinematic viscosity and the bulk modulus. The user can choose between three different methods:

CalcFluidConst The values of the bulk modulus and of the viscosity are constant

CalcFluidPress The bulk modulus and the viscosity are calculated dependent on pressure

CalcFluidTemp With this model the values are dependent on temperature and on pressure.

3.4.2 FluidProp

In the FluidProp File the constants of the fluids are saved. Then they can be accessed by the VolumeType and the models. The constants saved in the file can be seen in the appendix.

Chapter 4

Examples

In this chapter, different examples for the *Hydraulic* library are presented.

4.1 Long Line Relief

In this example a tank with high pressure is connected to a long line and a chamber with environment conditions at the beginning. Due to the pressure compensation, fluid flows from the tank through the long line into the chamber.
The heat is modeled as a lumped volume, that means the temperature in the long line and in the chamber is the same. All heat produced stays in the fluid. The three models of the example have the following parameters: The pressure in the *Tank* is $1 \cdot 10^7 Pa$, the Conductivity of the outflow of the Tank is $2 \cdot 10^{-8} \frac{m^3}{sPa}$. The *LongLine* consists of 10 *LineWithKinetics*. The total length of LongLine is $100m$ the Diameter is $30 \cdot 10^{-3} m$ and the roughness of the pipe is $1e-6m$. The *Chamber* has a volume of $1m^3$ and a start pressure of $1e5Pa$.
In the figure 4.2 the simulation results can be seen. The pressure rises from $1 \cdot 10^5 Pa$ to $1e7Pa$. There are overshooting effects that occur because of the kinetic effects in *LongLine*. Also it can be seen that the pressure does not rise from the beginning on but with a lag of about $0.066s$. This delay comes from the time lag in the line (the pressure shock advances with the speed of sound of the fluid).

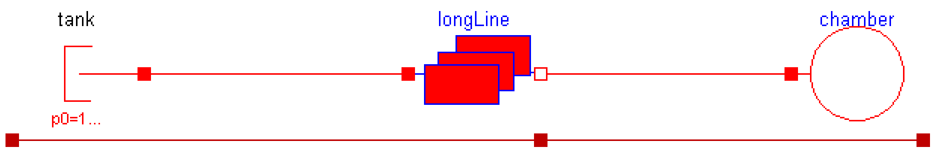


Figure 4.1: The model LongLineRelief. It contains three models, a Tank, a LongLine and a Chamber.

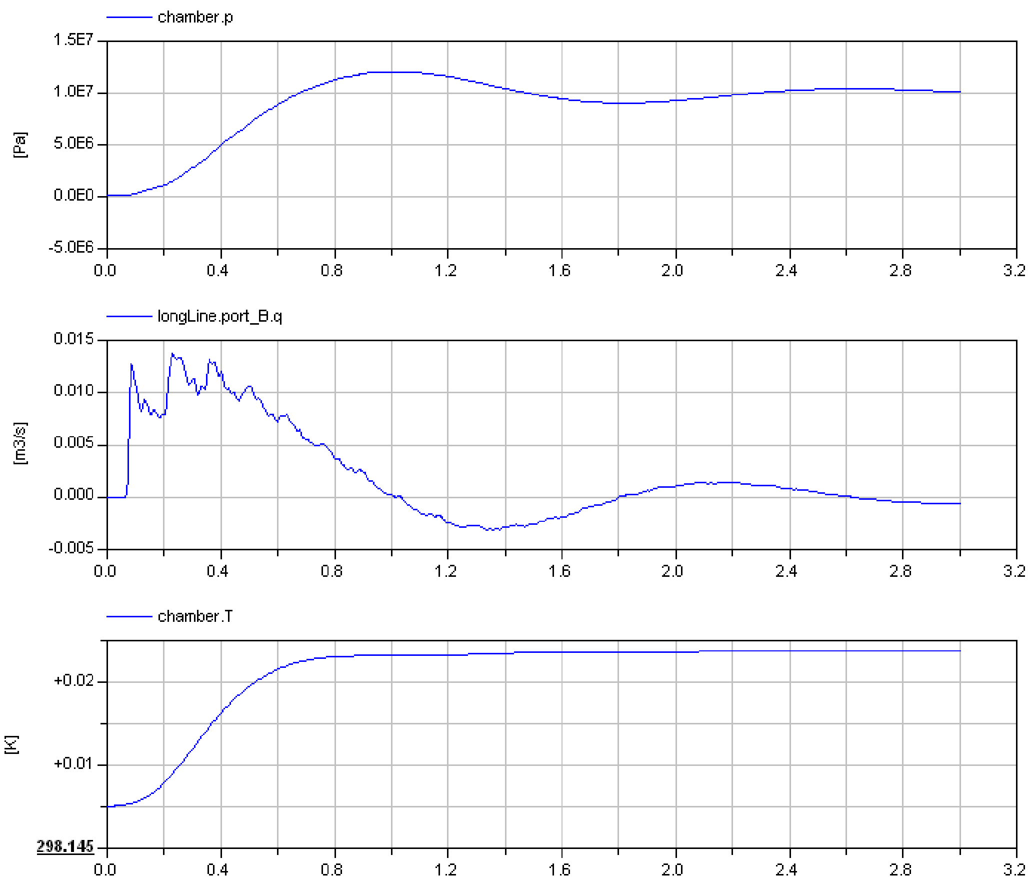


Figure 4.2: Simulation results of *LongLineRelief*. In the first graph the pressure in the chamber over the time is visible. The second graph shows the volumetric flow into the chamber. In the leftmost graph the temperature in the chamber can be seen.

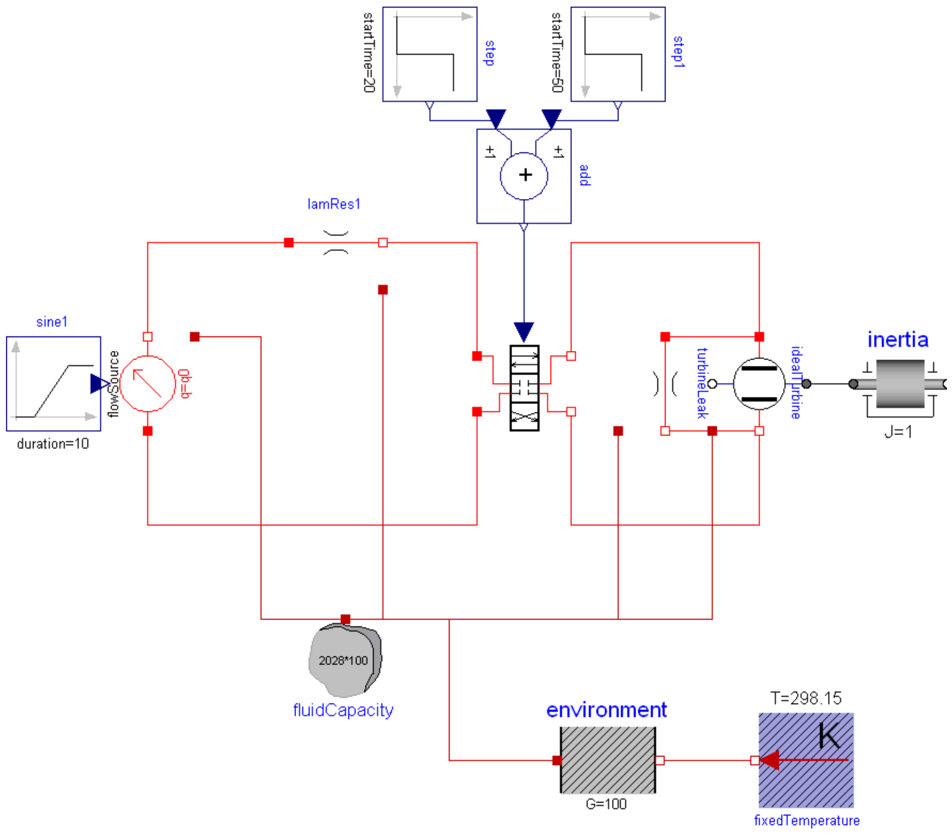


Figure 4.3: The model *TurbineControl*.

In the second part of figure 4.2 the volumetric flow into the chamber can be seen. Due to the pressure overshoot it changes from inflow to outflow at $1.03s$. The fluctuations have mainly two reasons: the first one is that pressure shocks run through the long line and cause a part of the fluctuations. The second part are fluctuations caused from the discretisation of the pipe. The third graph of figure 4.2 shows the Temperature in the fluid of the chamber. It rises due to the losses in the outflow of the *Tank* and in the *Longline*. The total temperature rise is $0.019K$. This temperature rise has very little influence on viscosity and bulk modulus. However if longer time intervalls were considered, the influence of temperature effect is no longer negligeable.

4.2 TurbineControl

The example *TurbineControl* shows how turbines and controllers can be modeled. It contains a flow source which pumps fluid. Then a 3-2-valve controlls the way the fluid going through the turbine. The turbine then moves. It could be connected to further mechanical parts.

This example contains no hydraulic capacitors or inductors. With this simplification it is easily possible to model controllers that can be completely shut or completely opened. As a

consequence hydraulic shocks are not visible in simulation.
In figure 4.3 the configuration of the example can be seen. The flow source starts at zero flow and then increases to $0.01\frac{m^3}{s}$. The internal leakage of the flowsource is set to $1 \cdot 10^{-6}\frac{m^3}{sPa}$. The laminar resistance is $1 \cdot 10^{-8}$. The controller is opened in the beginning at $T = 20s$ it closes and at $T = 50s$ it opens again but with the lines crossed.
The thermal capacity is modeled as a lumped volume of $100kg$ of mass. There is a thermal environment of $T_env = 298.15K$. The thermal conductance between the fluid and the environment is set to $1000\frac{W}{K}$. All the produced heat is first brought to the thermal capacity. The thermal capacity loses energy through the conductance to the environment.

In figure 4.4 The results of the simulation are plotted. The first graph shows the control of the flowsource. It is a ramp from zero to $0.01\frac{m^3}{s}$. The second graph shows the control of the 3-2-valve, wich shuts at $20s$ and opens crossed at $50s$. The third graph shows the position of the turbine. What can be seen is that the turbine stays still in the beginning. Then it accelerates in negative direction. As soon as the valve is closed, it stops to accelerate and slowly decceerates. When the valve is opened in the crossed configuration at $50s$, the turbine starts to accelerate and reaches a maximal velocity. The last graph shows the moment between the inertia and the hydraulic part of the turbine. It first rises proportional to the ascending flow in the flow source. When the valve shuts, there is very little torque. As soon as the valve opens again, the torque from the flow source acts in the opposite direction.

In figure 4.5 the temperature in the fluid is plotted. As expected the temperature first rises. The rising slows down, and a equilibrium state at about $308K$ is reached. In this state the absorbed heat of the environment is equal to the produced heat in the elements.

4.3 Hydraulic Gear

This example shows the possibility for models with pistons. Two pistons with different areas are connected via a turbulent resistance. If a force acts on one piston it is transferred via the pressure line to the second piston. The areas act as amplificators of the force.

In figure 4.6 the set up of the example can be seen. The position of the upper piston is controlled. A tank represents the environment pressure of $1e5Pa$. The piston is connected to a turbulent resistance, it has a area of $0.01m^2$. The other connector of the turbulent resistance is connected to a second piston with an area of $0.02m^2$. The second piston is on one side fixed, on the other side there is a mass of $1kg$ and a spring-damper element. The gear ratio can be calculated by the division of the two areas and is 2.
In figure 4.7 the simulation results are shown. The position of the driving piston moves much more because of the gear ratio. The force graph shows the inverse effect. The pressure drop in the turbulent resistance goes up to $1500Pa$. The maximal flow through the turbulent resistance is $0.004\frac{m^3}{s}$.

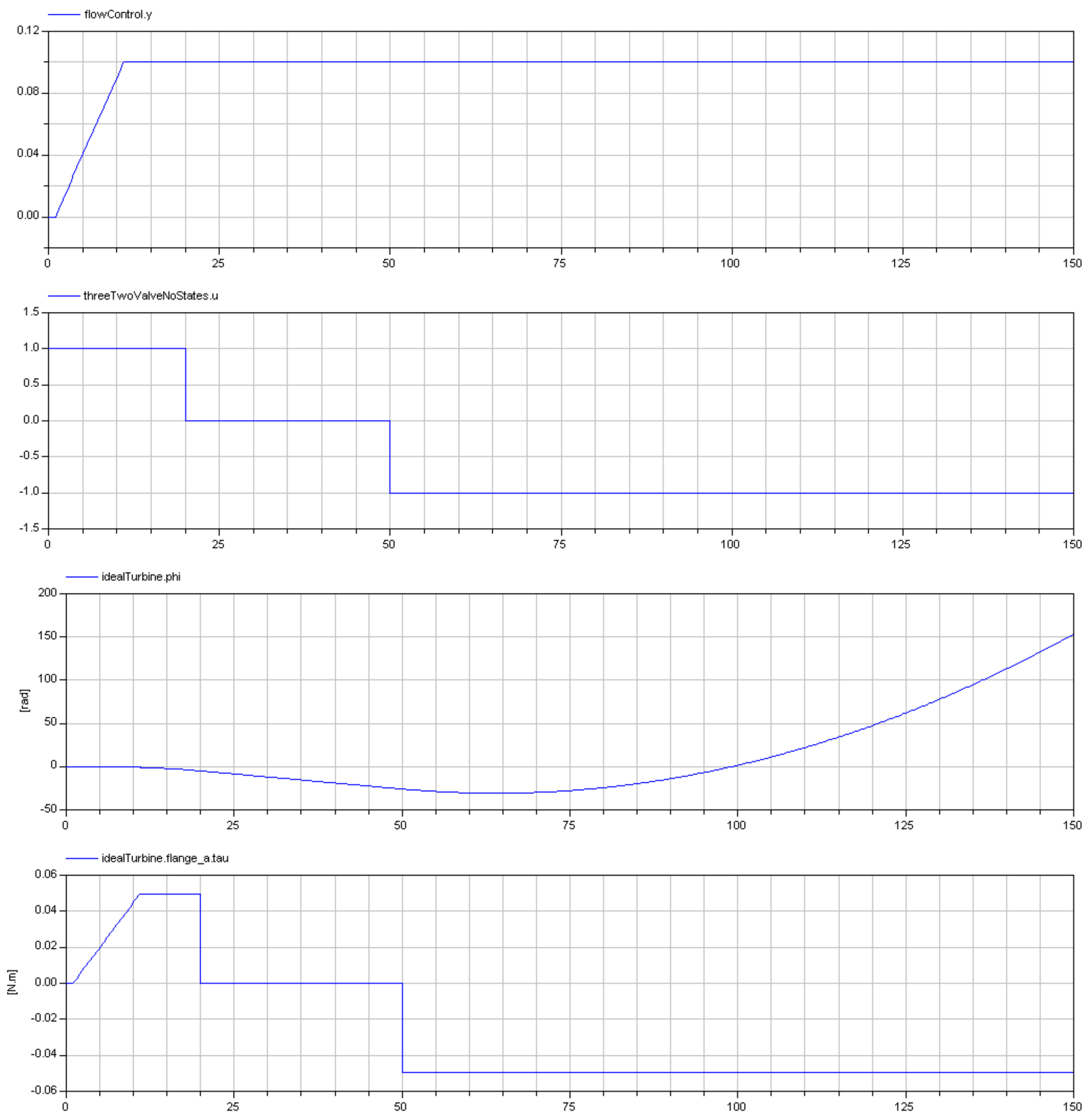


Figure 4.4: Simulation results of *TurbineControl*. The first graph shows the volumetric flowrate of the flowsource that is commanded. The second graph shows the control of the 3-2-valve, first direct flow, then blocked and in after that crossed flow. The third graph shows the resulting angle of the turbine. The last graph shows the moment that is transferred between the turbine and the inertia of the turbine. These results are all for simulation time $150s$

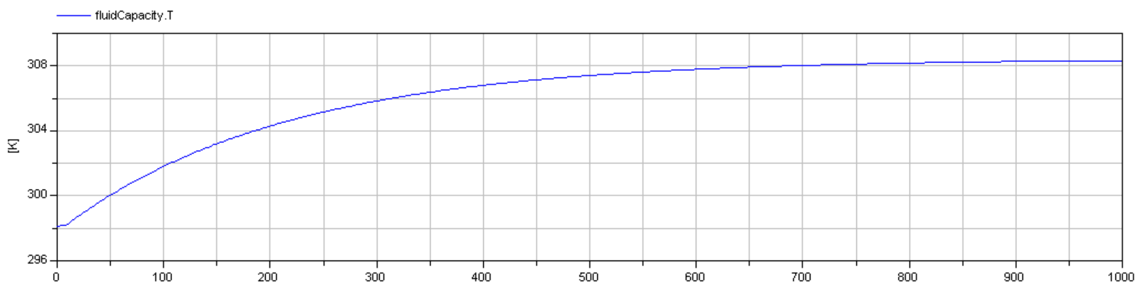


Figure 4.5: Simulation results of *TurbineControl* The graph shows the Temperature of the lumped volume for a timespan of 150s

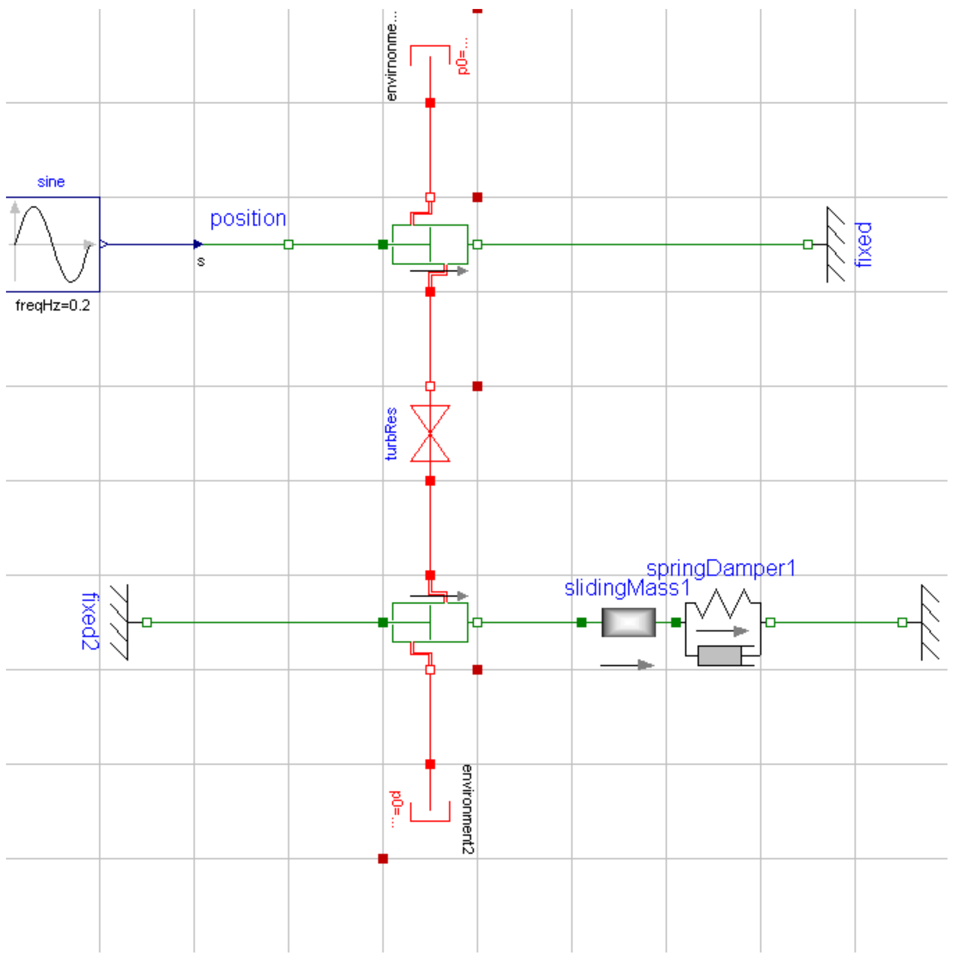


Figure 4.6: The example *hydraulicGear*

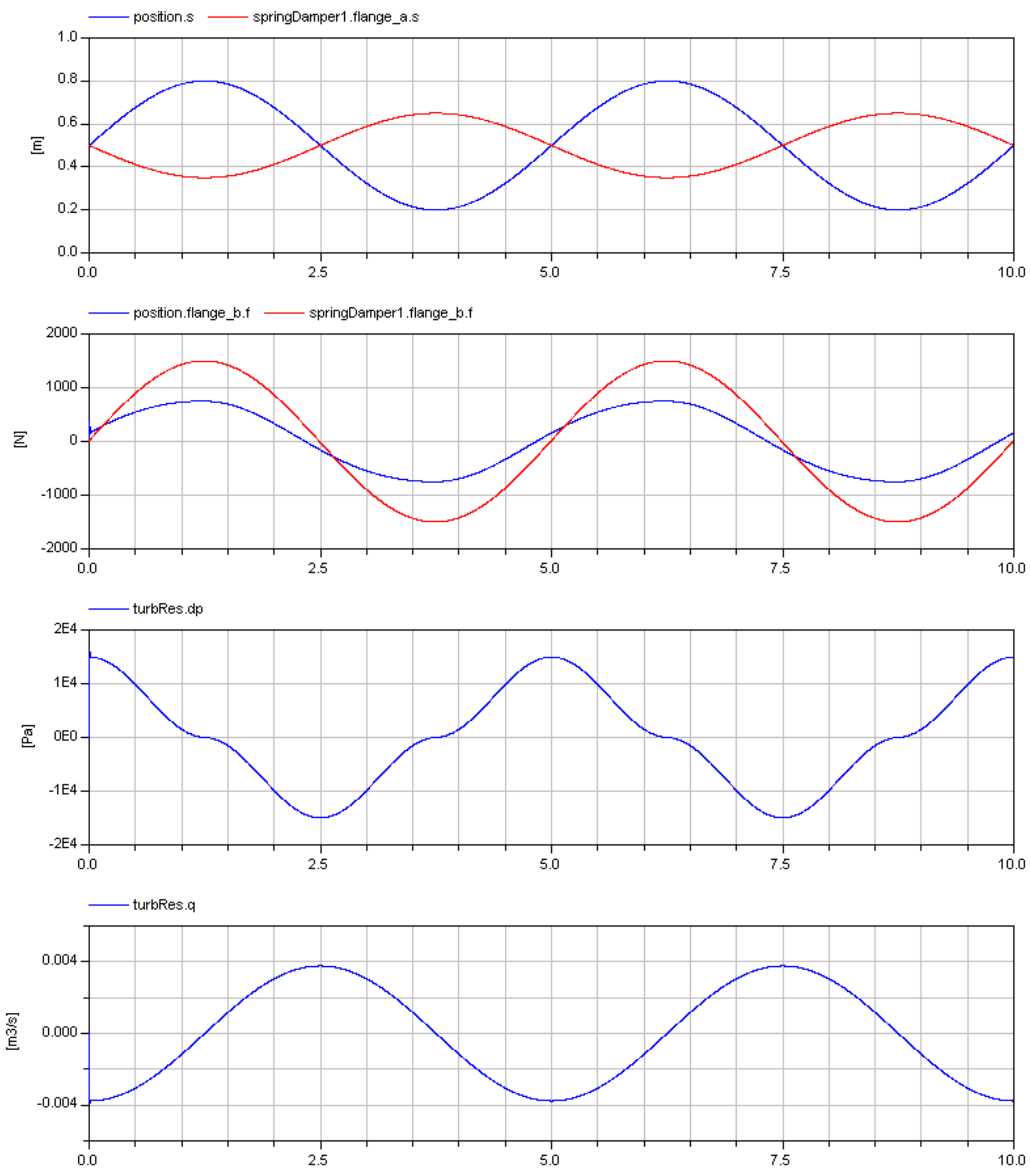


Figure 4.7: Simulation result of the example *hydraulicGear*. The first graph shows in blue the position of the upper piston and in red the position of the lower one. In the second graph the force is plotted, in blue the force in the upper piston, in red the force in the lower piston. The third graph shows the pressure drop in the turbulent resistance and the fourth graph shows the volumetric flow through the turbulent resistance

Bibliography

[1] A.D. Altshul. *Hydraulic Resistance.* Nedra Press, Moscow, 1982.

[2] P. Beater. Modeling and digital simulation of hydraulic systems in design and engineering education using modelica and hylib. 2000.

[3] D. Brück, H. Elmqvist, H. Olsson, and S.E. Mattsson. Dymola for multi-engineering modeling and simulation. 2002.

[4] F. E. Cellier. *Continuous system modeling.* Springer-Verlag, New York [etc.], 1991. 90-25286 François E. Cellier.

[5] F. E. Cellier and Greifeneder J. Thermobondlib - a new modelica library for modeling convective flows. 2008.

[6] F. E. Cellier and A. Nebot. The modelica bond graph library. 2005.

[7] H. Dubbel and K. H. Grote. *Dubbel Taschenbuch für den Maschinenbau.* Springer, Berlin, 22., neu bearb. und erw. aufl. edition, 2007. Hrsg. von Karl-Heinrich Grote ... [et al.].

[8] I. E. Idelcik and M. O. Steinberg. *Handbook of hydraulic resistance.* Jaico Pub. House, Mumbai, 3rd ; 6th print. edition, 2008. I.E. Idelchik ; editor, M.O. Steinberg ; translated by Greta R. Malyaskaya ; translation editor, Oleg G. Martynenko "Jaico book", JH-429 – T.p. verso Includes bibliographical references and index.

[9] D. C. Karnopp, D. L. Margolis, and R. C. Rosenberg. *System dynamics modeling and simulation of mechatronic systems.* John Wiley & Sons, Hoboken, N.J., 4th edition, 2006. Dean C. Karnopp, Donald L. Margolis, Ronald C. Rosenberg.

[10] M. J. Moran and H. N. Shapiro. *Fundamentals of engineering thermodynamics.* Wiley, Hoboken, 6th edition, 2008. Michael J. Moran, Howard N. Shapiro 1 CD-ROM ; 1 Begleitband CD-ROM includes example problems from "Fundamentals of engineering thermodynamics".

Hydraulic

Package Content

Name	Description
UsersGuide	Users Guide
Basic	Package of Basic Models
Interfaces	Hydraulic interfaces
Pumps	Pump models
Components	Hydraulic components
Sensors	Package sensors
Controllers	Hydraulic Controllers
Examples	Examples for the Hydraulic Library

HTML-documentation generated by Dymola Thu Jul 21 09:16:19 2011.

Hydraulic.UsersGuide

Users Guide of Package Hydraulic

Library **Hydraulic** is a **free** Modelica package providing components to model hydraulic systems in a convenient domain-independent format. This package contains the **users guide** to the library, and has the following content:

1. Overview of Library gives an overview of the library.
2. Release Notes offers a historic perspective on the development of this library.

Package Content

Name	Description
Overview	Overview of Library
Versions	Realease Notes

HTML-documentation generated by Dymola Thu Jul 21 09:16:20 2011.

Hydraulic.UsersGuide.Overview

This overview contains the following sections:

1. An introduction to hydraulic modeling
2. Hydraulic connectors
3. Fluid properties
4. How to use the hydraulic library
5. How to use thermal connectors
6. Pump elements
7. Component elements
8. Controll elements
9. Examples

Package Content

Name	Description
Introduction	An introduction to hydraulic modeling
Connectors	Connectors in the Hydraulic library
Fluid	Fluid properties in the hydraulic Library
HydraulicHowTo	How to use the hydraulic library
ThermalHowTo	How to use the thermal connectors in the hydraulic library
Pumps	Pump Elements
Components	Component Elements
Controllers	Controll Elements
Examples	Examples for the hydraulic Library

Hydraulic.UsersGuide.Overview.Introduction

The **Hydraulic** package offers a graphical approach to Model hydraulic Systems.

The library contains models for common hydraulic elements. The elements contain hydraulic connectors which contain pressure **p** and volumeflow **q**. In each model also a heat port is provided which contains the temperature **T** and the entropy flow **Sdot**. The heat port provides information on energy produced in the element as well as information on the temperature to calculate the fluid properties.

Hydraulic.UsersGuide.Overview.Connectors

The **Hydraulic** library uses its own hydraulic ports. In the ports there are two variables: the effort variable pressure **p** and the flow variable volumetric flow **q**. The flow **q** is positive if the flow is out of the element and negative if it flows into the element.

Hydraulic.UsersGuide.Overview.Fluid

Properties of the fluid can be set independently of the models. The different models (e.g. chamber) extend one of the three files **CalcFluid -Const**, **-Press** or **-Temp**. In these files the bulkmodulus and the viscosity of the fluid are calculated, either as constant values, pressure dependent, or pressure and temperature dependent. The files **CalcFluid** extend the file **FluidProp** where the constants of the used fluid are saved. To change the fluid only this file has to be changed and there is no need to change the models.

Hydraulic.UsersGuide.Overview.HydraulicHowTo

The **Hydraulic** library can be used just by draging and dropping of the models. Then the parameters have to be set and all the elements have to be connected to each other.

The following points have to be considered:

- The hydraulic connectors used in this library are **not** compatible to the connectors in the HyLibLight library.
- If the initial values of pressure of adjoining chambers are very different numerical problems during simulation can occur.

Hydraulic systems are often very stiff in a numerical way. In those cases the simulation can take a long time or it can even completely fail. The following points help to reduce these problems to a certain extend:

- Reduce tolerance in simulation, e.g. to 1e-10
- Use small **chambers** between the models. These chambers have very small volume and help to smooth out peaks in the beginning of the simulation
- Use of **Inertia** can help to reduce the variability of flows.

Hydraulic.UsersGuide.Overview.ThermalHowTo

Most of the models in **Hydraulic** have a heat port attached. They can be used for maynly three things:

- Computation of the produced heat in resistive elements
- Computation of the temperature in the fluid
- Use of the temperature information for fluid properties

The heat ports do **not** have to be connected! If they are not connected the Temperature **TNom** is used as temperature in the element.

If the heat port is connected it is used in the following way:

- In the purely resistive elements, such as **LamRes**, **TurbRes** and **Line** the produced heat is calculated and given out through the heat port
- Each **Chamber** has also a **heatCapacity**, so it stores the heat in the fluid
- Note that convective flows are **not** included in this library. However convective flows can be modeled as shown in the following **EXAMPLE**

Hydraulic.UsersGuide.Overview.Pumps

The models in **Pumps** describe hydraulic elements which are either sources of hydraulic power or power converter between mechanical and hydraulic systems.

Hydraulic.UsersGuide.Overview.Components

In the package **Components** models of passive hydraulic elements are presented. These models represent the different standard elements that are normally used.

Hydraulic.UsersGuide.Overview.Controllers

In this part of the library **Hydraulic** contains different controllers to controll the fluid flow. The models are of much different complexity. If in doubt, use the simpler models because the need much less computational power and usually are simulated with less numerical problems.

Hydraulic.UsersGuide.Overview.Examples

There are three examplex presented in the hydraulic library. The example LongLineRelief shows a tank that pumps oil via a long line into a chamber. The second example TurbineControl represents a turbine that is controlled by a flow source and a 3-2-valve. The third example HydraulicGear is an example for a hydraulic gear. A piston is driven mechanically, a second piston with a different area drives then mechanical components

---

HTML-documentation generated by Dymola Thu Jul 21 09:16:20 2011.

Hydraulic.UsersGuide.Versions

This is Version 1.0 of the Hydraulic Library

HTML-documentation generated by Dymola Thu Jul 21 09:16:20 2011.

Hydraulic.Basic

Package of Basic Models

Package Content

Name	Description
GSswitch	Conductance with switch between laminar and turbulent
mGSswitch	Modulated conductance element with switch between laminar and turbulent
RSquad	Resistance with linear and quadratic term
mRSquad	modulated resistance with linear and quadratic term

Hydraulic.Basic.GSswitch

Conductance with switch between laminar and turbulent

Information

The bondgraphic conductive hydraulic switching source is a directed TwoPort element. It inherits the effort and flow variables from the directed TwoPort.

The element is made for hydraulic flows. If the critical Re-number is reached, the flow switches from laminar to turbulent. The flows behave after the Moody-Diagram. The primary side of the linear conductive source element is assumed to not be in the thermal domain.

The conductive hydraulic switching source element has free causality on the primary side, and fixed causality on the secondary side. The causality stroke is at the element on the secondary side (a source of entropy, rather than a source of temperature).

```
Potential variables:

 e1:  Bondgraphic effort variable of inflow

 f1:  Bondgraphic flow variable of inflow, normalized positive for flows into the model

 e2:  Bondgraphic effort variable of outflow

 f2:  Bondgraphic flow variable of outflow, normalized positive for flows out of the model
```

Parameters

Type	Name	Default	Description
DynamicViscosity	nu		Kinematic viscosity [Pa.s]
Length	D		Diameter [m]
Length	L		Length [m]
Length	e_r		Roughness [m]
Density	rho		Density [kg/m3]
Real	ReCrit	2000	Critical Reynolds number

Connectors

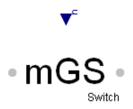
Type	Name	Description
BondCon	BondCon1	Left bond graph connector
BondCon	BondCon2	Right bond graph connector

Modelica definition

```
model GSswitch
  "Conductance with switch between laminar and turbulent"
  extends BondLib.Interfaces.TwoPort;
  parameter Modelica.SIunits.DynamicViscosity nu "Kinematic viscosity";
  parameter Modelica.SIunits.Length D "Diameter";
  parameter Modelica.SIunits.Length L "Length";
  parameter Modelica.SIunits.Length e_r "Roughness";
  parameter Modelica.SIunits.Density rho "Density";
  parameter Real ReCrit=2000 "Critical Reynolds number";
  Real Re "Reyolds Number";

  Boolean turbulent(start=false);


equation
   turbulent = (Re>ReCrit);

   e1=if turbulent then 0.44*L*rho/(D^5*Modelica.Constants.pi^2)*(e_r/D + 17*
    Modelica.Constants.pi*nu*D/abs(f1))^.25*abs(f1)*f1 else 128*L*rho*nu*f1/(D^
    4*Modelica.Constants.pi);

  Re =abs(4*f1/(D*Modelica.Constants.pi*nu));
  e1*f1 = e2*f2;

end GSswitch;
```

Hydraulic.Basic.mGSswitch

Modulated conductance element with switch between laminar and turbulent



Information

The modulated bondgraphic conductive hydraulic switching source is a directed TwoPort element. It inherits the effort and flow variables from the directed TwoPort. The input variable controlls the diameter of the pipe, if u=0 then Diameter is minimal, if u=1 then D=D0

The element is made for hydraulic flows. If the critical Re-number is reached, the flow switches from laminar to turbulent. The flows behave after the Moody-Diagram. The primary side of the linear conductive source element is assumed to not be in the thermal domain.

The modulated conductive hydraulic switching source element has free causality on the primary side, and fixed causality on the secondary side. The causality stroke is at the element on the secondary side (a source of entropy, rather than a source of temperature).

Potential variables:

e1: Bondgraphic effort variable of inflow

f1: Bondgraphic flow variable of inflow, normalized positive for flows into the model

e2: Bondgraphic effort variable of outflow

f2: Bondgraphic flow variable of outflow, normalized positive for flows out of the model

Parameters

Type	Name	Default	Description
DynamicViscosity	nu		Dynamic Viscosity [Pa.s]
Length	D0		Maximal Diameter [m]
Length	L		Length [m]
Length	e_r		Roughness [m]
Density	rho		Density [kg/m3]
Real	ReCrit	3000	Critical Reynolds number

Connectors

Type	Name	Description
BondCon	BondCon1	Left bond graph connector
BondCon	BondCon2	Right bond graph connector
input RealInput	u	

Modelica definition

```
model mGSswitch
  "Modulated conductance element with switch between laminar and turbulent"

  extends BondLib.Interfaces.TwoPort;

  parameter Modelica.SIunits.DynamicViscosity nu "Dynamic Viscosity";
  parameter Modelica.SIunits.Length D0 "Maximal Diameter";
  Modelica.SIunits.Length D "Diameter";
  parameter Modelica.SIunits.Length L "Length";
  parameter Modelica.SIunits.Length e_r "Roughness";
  parameter Modelica.SIunits.Density rho "Density";
  parameter Real ReCrit=3000 "Critical Reynolds number";
  Real Re "Reyolds Number";
  Boolean turbulent;

  Modelica.Blocks.Interfaces.RealInput u;
equation
  u=D/D0;
  turbulent =  (Re>ReCrit);
  Re =abs(4*f1/(D*Modelica.Constants.pi*nu));

     e1=if turbulent then 0.44*L*rho/(D^5*Modelica.Constants.pi^2)*(e_r/D + 17*
    Modelica.Constants.pi*nu*D/abs(f1))^.25*abs(f1)*f1 else 128*L*rho*nu*f1/(D^
    4*Modelica.Constants.pi);
  e1*f1 = e2*f2;
end mGSswitch;
```

Hydraulic.Basic.RSquad

Resistance with linear and quadratic term

Information

The bondgraphic linear and quadratic resistive source is a directed TwoPort element. It inherits the effort and flow variables from the directed TwoPort. The linear

The primary side of the linear resistive source element is assumed to not be in the thermal domain.

The resistive source element has free causality on the primary side, and fixed causality on the secondary side. The causality stroke is at the element on the seconda

Potential variables:

e1: Bondgraphic effort variable of inflow

f1: Bondgraphic flow variable of inflow, normalized positive for flows into the model

e2: Bondgraphic effort variable of outflow

f2: Bondgraphic flow variable of outflow, normalized positive for flows out of the model

Equations:

e1 = a*f1 + b*abs(f1)*f1

e1*f1 = e2*f2

Parameters

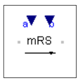
Type	Name	Default	Description
Real	a		Linear Parameter
Real	b		Quadratic Parameter

Connectors

Type	Name	Description
BondCon	BondCon1	Left bond graph connector
BondCon	BondCon2	Right bond graph connector

Modelica definition

```
model RSquad "Resistance with linear and quadratic term"
  extends BondLib.Interfaces.TwoPort;

  parameter Real a "Linear Parameter";
  parameter Real b "Quadratic Parameter";

equation
  e1=a*f1+b*abs(f1)*f1;
  e1*f1=e2*f2;
end RSquad;
```



Hydraulic.Basic.mRSquad

modulated resistance with linear and quadratic term

Information

The modulated bondgraphic linear and quadratic resistive source is a directed TwoPort element. It inherits the effort and flow variables from the directed TwoPort.

The primary side of the linear resistive source element is assumed to not be in the thermal domain.

The modulated resistive source element has free causality on the primary side, and fixed causality on the secondary side. The causality stroke is at the element on t

---

Potential variables:

e1: Bondgraphic effort variable of inflow

f1: Bondgraphic flow variable of inflow, normalized positive for flows into the model

e2: Bondgraphic effort variable of outflow

f2: Bondgraphic flow variable of outflow, normalized positive for flows out of the model

---

Equations:

e1 = a*f1 + b*abs(f1)*f1

e1*f1 = e2*f2

Connectors

Type	Name	Description
BondCon	BondCon1	Left bond graph connector
BondCon	BondCon2	Right bond graph connector
input RealInput	a	
input RealInput	b	

Modelica definition

```
model mRSquad "modulated resistance with linear and quadratic term"
  extends BondLib.Interfaces.TwoPort;

  Modelica.Blocks.Interfaces.RealInput a;
  Modelica.Blocks.Interfaces.RealInput b;
equation
  e1=a*f1+b*abs(f1)*f1;
  e1*f1=e2*f2;
end mRSquad;
```

HTML-documentation generated by Dymola Thu Jul 21 09:16:20 2011.

Hydraulic.Interfaces

Hydraulic interfaces

Package Content

Name	Description
HydraulicUnits	package with hydraulic units
FluidProps	Package of fluid properties
Port	Model of Hydraulic Port
🟥 Port_A	Hydraulic port A
□ Port_B	Hydraulic port B
Hy2BG	Conversion from hydraulic to bond graph
BG2Hy	Conversion from bond graph to hydraulic
OnePort	Hydraulic one port model
TwoPortComponent	Hydraulic two port model
TwoPortSystem	Two Port System
OnePortTemp	Hydraulic one port model with heatport
TwoPortComponentTemp	Hydraulic two port model with heat port
TwoPortSystemTemp	Hydraulic two port system
TranslationSystemTemp	System with mechanical and hydraulic parts
RotationSystemTemp	System with rotational mechanic and hydraulic parts

Hydraulic.Interfaces.Port

Model of Hydraulic Port

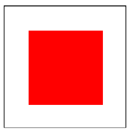
Contents

Type	Name	Description
AbsolutePressure	p	pressure at port [Pa]
flow VolumeFlowRate	q	flow rate through port [m3/s]

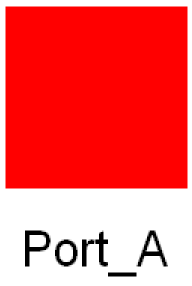
Modelica definition

```
connector Port "Model of Hydraulic Port"
  Modelica.SIunits.AbsolutePressure p "pressure at port";
  flow Modelica.SIunits.VolumeFlowRate q "flow rate through port";
end Port;
```

Hydraulic.Interfaces.Port_A



Hydraulic port A



Information

This connector is incompatible with the connectors from HyLibLight.

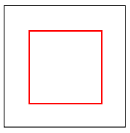
Contents

Type	Name	Description
AbsolutePressure	p	pressure at port [Pa]
flow VolumeFlowRate	q	flow rate through port [m3/s]

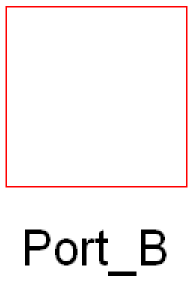
Modelica definition

```
connector Port_A "Hydraulic port A"
  extends Port;

end Port_A;
```

Hydraulic.Interfaces.Port_B



Hydraulic port B



Information

This connector is incompatible with the connectors from HyLibLight.

Contents

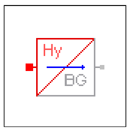
Type	Name	Description
AbsolutePressure	p	pressure at port [Pa]
flow VolumeFlowRate	q	flow rate through port [m3/s]

Modelica definition

```
connector Port_B "Hydraulic port B"
  extends Port;
```

```
end Port_B;
```

Hydraulic.Interfaces.Hy2BG



Conversion from hydraulic to bond graph

Information

Wrapper model translating hydraulic (external) connectors to bond graph (internal) connectors.

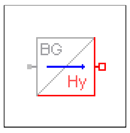
Connectors

Type	Name	Description
BondCon	bondCon	
Oil		
Properties		
Port_A	port_A	

Modelica definition

```
model Hy2BG "Conversion from hydraulic to bond graph"

  Hydraulic.Interfaces.Port_A port_A;
  BondLib.Interfaces.BondCon bondCon;
equation
  bondCon.e=port_A.p;
  bondCon.f*bondCon.d=port_A.q;
end Hy2BG;
```

Hydraulic.Interfaces.BG2Hy



Conversion from bond graph to hydraulic

Information

Wrapper model translating hydraulic (external) connectors to bond graph (internal) connectors.

Connectors

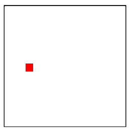
Type	Name	Description
BondCon	bondCon	
Oil		
Properties		
Port_B	port_B	

Modelica definition

```
model BG2Hy "Conversion from bond graph to hydraulic"

  BondLib.Interfaces.BondCon bondCon;
  Port_B port_B;
```

```
equation
  bondCon.e=port_B.p;
  bondCon.f*bondCon.d=port_B.q;

end BG2Hy;
```

Hydraulic.Interfaces.OnePort



Hydraulic one port model

Information

Hydraulic one port model.

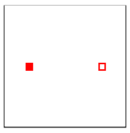
Connectors

Type	Name	Description
Oil		
Properties		
Port_A	port_A	Hydraulic Connector

Modelica definition

```
partial model OnePort "Hydraulic one port model"

  Modelica.SIunits.AbsolutePressure p "Pressure at Port";
  Modelica.SIunits.VolumeFlowRate q "Volumeflow at Port";
  Interfaces.Port_A port_A "Hydraulic Connector";
equation
  port_A.p = p;
  port_A.q = q;
end OnePort;
```

Hydraulic.Interfaces.TwoPortComponent



Hydraulic two port model

Information

Hydraulic two port model.

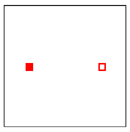
Connectors

Type	Name	Description
Oil		
Properties		
Port_A	port_A	
Port_B	port_B	Hydraulic Connector B

Modelica definition

```
partial model TwoPortComponent "Hydraulic two port model"
  Interfaces.Port_A port_A;
  Interfaces.Port_B port_B "Hydraulic Connector B";
  Modelica.SIunits.AbsolutePressure p "Pressure at port_A";
  Modelica.SIunits.Pressure dp "Pressuredrop over Ports";
  Modelica.SIunits.VolumeFlowRate q "Volumeflow through Component";

equation
  p=port_A.p;
  dp = port_A.p - port_B.p;
  q = port_A.q;
end TwoPortComponent;
```

Hydraulic.Interfaces.TwoPortSystem



Two Port System

Information

Hydraulic two port system. It extends VolumeType to calculate bulk modulus and kinematic viscosity.

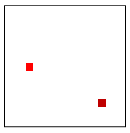
Connectors

Type	Name	Description
Oil		
Properties		
Port_A	port_A	Hydraulic Connector A
Port_B	port_B	Hydraulic Connector B

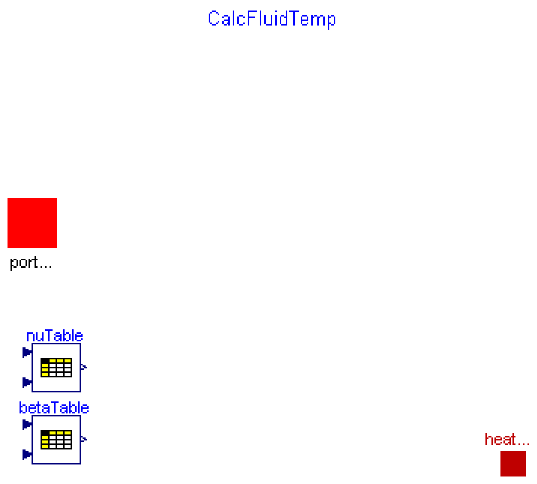
Modelica definition

```
partial model TwoPortSystem "Two Port System"
  Interfaces.Port_A port_A "Hydraulic Connector A";
  Interfaces.Port_B port_B "Hydraulic Connector B";
  Modelica.SIunits.AbsolutePressure p "Pressure at port_A";
equation
  p=port_A.p;

end TwoPortSystem;
```

Hydraulic.Interfaces.OnePortTemp



Hydraulic one port model with heatport



Information

Hydraulic one port model with additional heatport. It extends VolumeType to calculate bulk modulus and kinematic viscosity.

Parameters

Type	Name	Default	Description
Oil			
Properties			
replaceable model Prop	FluidProp		constants of fluid models

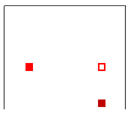
Connectors

Type	Name	Description
HeatPort_a	heatPort	
Oil		
Properties		
Port_A	port_A	

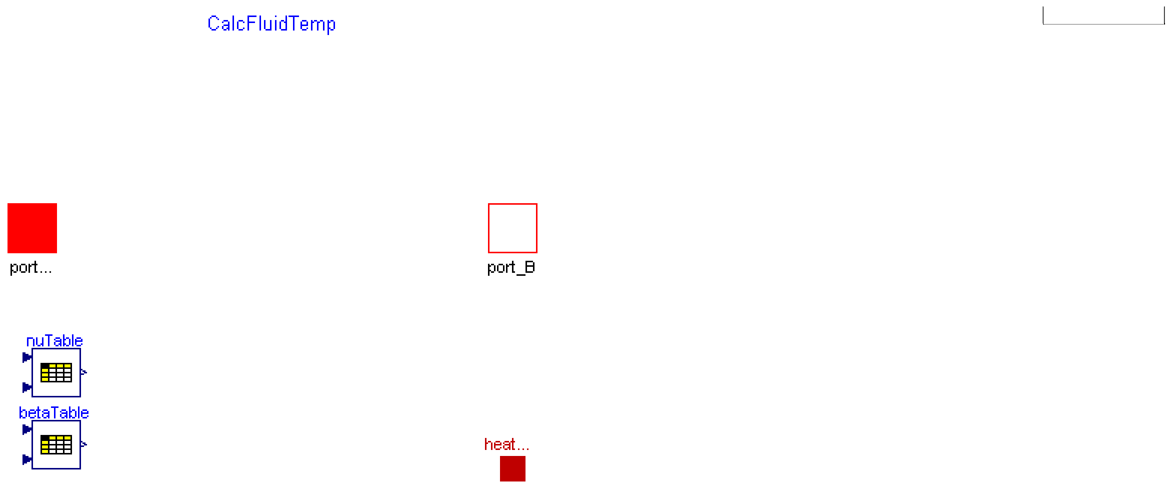
Modelica definition

```
partial model OnePortTemp "Hydraulic one port model with heatport"
  Modelica.SIunits.AbsolutePressure p "Start pressure at Port";
  Modelica.SIunits.VolumeFlowRate q "Volumeflow at Port";
  Interfaces.Port_A port_A;
  replaceable model VolumeType =
      Hydraulic.Interfaces.FluidProps.CalcFluidTemp
  extends VolumeType "model of oil bulk modulus and kinematic viscosity";
  extends VolumeType;
  Modelica.Thermal.HeatTransfer.Interfaces.HeatPort_a heatPort;

equation
  port_A.p = p;
  port_A.q = q;
end OnePortTemp;
```

Hydraulic.Interfaces.TwoPortComponentTemp



Hydraulic two port model with heat port



Information

Hydraulic two port model with additional heatport. It extends VolumeType to calculate bulk modulus and kinematic viscosity.

Parameters

Type	Name	Default	Description
Oil			
Properties			
replaceable model Prop	FluidProp	constants of fluid models	

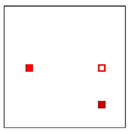
Connectors

Type	Name	Description
HeatPort_a	heatPort	
Oil		
Properties		
Port_A	port_A	
Port_B	port_B	Hydraulic Connector B

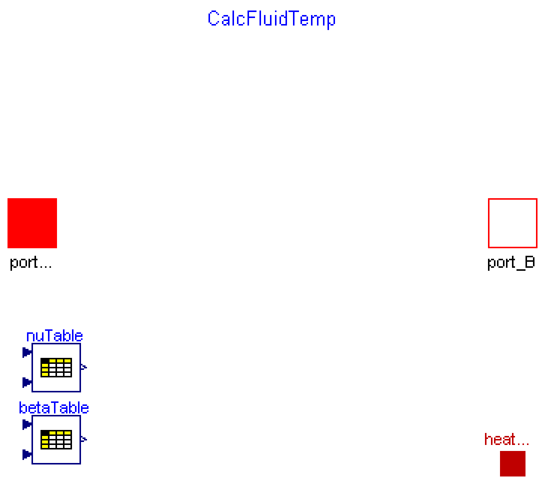
Modelica definition

```
partial model TwoPortComponentTemp
  "Hydraulic two port model with heat port"
  Modelica.SIunits.AbsolutePressure p "Pressure at port_A";
  Modelica.SIunits.Pressure dp "Pressuredrop over Ports";
  Modelica.SIunits.VolumeFlowRate q "Volumeflow through Component";
  Modelica.SIunits.Temperature T "Temperature";
  replaceable model VolumeType =
      Hydraulic.Interfaces.FluidProps.CalcFluidTemp
  extends VolumeType "model of oil bulk modulus and kinematic viscosity";

  extends VolumeType;
  Interfaces.Port_A port_A;
  Interfaces.Port_B port_B "Hydraulic Connector B";

  Modelica.Thermal.HeatTransfer.Interfaces.HeatPort_a heatPort;

equation
  p=port_A.p;
  dp = port_A.p - port_B.p;
  q = port_A.q;
end TwoPortComponentTemp;
```

Hydraulic.Interfaces.TwoPortSystemTemp



Hydraulic two port system



Information

Hydraulic two port system with additional heatport. It extends VolumeType to calculate bulk modulus and kinematic viscosity.

Parameters

Type	Name	Default	Description
Oil			
Properties			
replaceable model Prop	FluidProp	constants of fluid models	

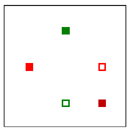
Connectors

Type	Name	Description
HeatPort_a	heatPort	
Oil		
Properties		
Port_A	port_A	
Port_B	port_B	Hydraulic Connector B

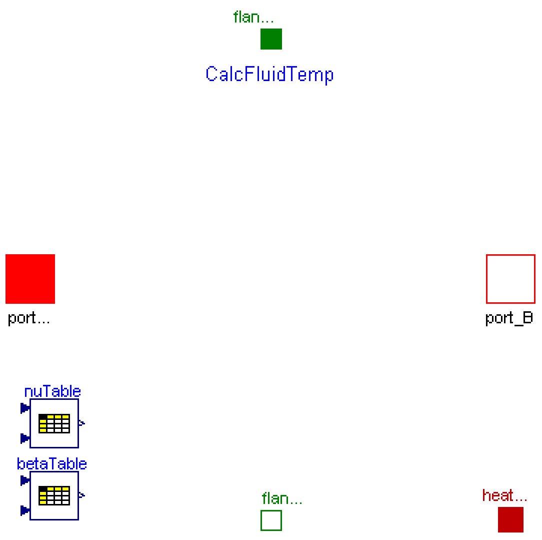
Modelica definition

```
partial model TwoPortSystemTemp "Hydraulic two port system"

  replaceable model VolumeType =
      Hydraulic.Interfaces.FluidProps.CalcFluidTemp
  extends VolumeType "model of oil bulk modulus and kinematic viscosity";
  extends VolumeType;
  Interfaces.Port_A port_A;
  Interfaces.Port_B port_B "Hydraulic Connector B";
  Modelica.Thermal.HeatTransfer.Interfaces.HeatPort_a heatPort;
  Modelica.SIunits.AbsolutePressure p "Pressure at port_A";
equation
  p=port_A.p;
end TwoPortSystemTemp;
```

Hydraulic.Interfaces.TranslationSystemTemp



System with mechanical and hydraulic parts



Information

Hydraulic and mechanical system with 2 hydraulic ports, two translational mechanical ports and one heatport. It extends VolumeType to calculate bulk modulus and kinematic viscosity.

Parameters

Type	Name	Default	Description
Oil			
Properties			
replaceable model Prop	FluidProp	constants of fluid models	

Connectors

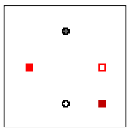
Type	Name	Description
Flange_a	flange_a	Translational Connector
Flange_b	flange_b	Translational Connector
HeatPort_a	heatPort	
Oil		
Properties		
Port_A	port_A	
Port_B	port_B	Hydraulic Connector B

Modelica definition

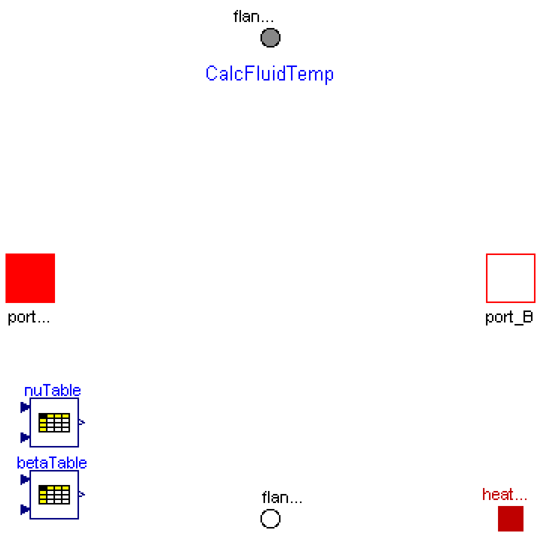
```
partial model TranslationSystemTemp
```

```
  "System with mechanical and hydraulic parts"

  replaceable model VolumeType =
      Hydraulic.Interfaces.FluidProps.CalcFluidTemp
  extends VolumeType "model of oil bulk modulus and kinematic viscosity";
  extends VolumeType;

  Modelica.SIunits.Position s "Absolute position of flange";
  Modelica.SIunits.Velocity v "Absolute velocity of flange";
  Interfaces.Port_A port_A;
  Interfaces.Port_B port_B "Hydraulic Connector B";
  BondLib.Mechanical.Translational.Interfaces.Flange_a flange_a
    "Translational Connector";
  BondLib.Mechanical.Translational.Interfaces.Flange_b flange_b
    "Translational Connector";
  Modelica.Thermal.HeatTransfer.Interfaces.HeatPort_a heatPort;
  Modelica.SIunits.AbsolutePressure p "Pressure at port_A";
equation
  p=port_A.p;
  flange_a.s = s;
  flange_a.v = v;
end TranslationSystemTemp;
```

Hydraulic.Interfaces.RotationSystemTemp



System with rotational mechanic and hydraulic parts



Information

Hydraulic and mechanical system with 2 hydraulic ports, two rotational mechanical ports and one heatport. It extends VolumeType to calculate bulk modulus and kinematic viscosity.

Parameters

Type	Name	Default	Description
Oil			
Properties			
replaceable model Prop	FluidProp	constants of fluid models	

Connectors

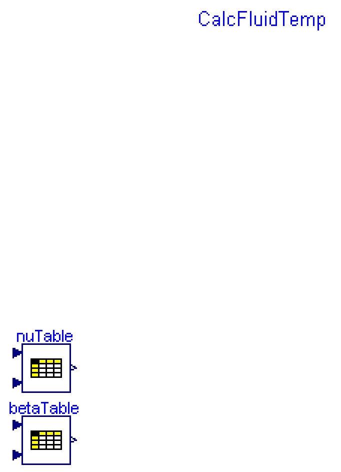
Type	Name	Description
Flange_a	flange_a	Rotational Connector
Flange_b	flange_b	Rotational Connector
HeatPort_a	heatPort	
Oil		
Properties		
Port_A	port_A	
Port_B	port_B	Hydraulic Connector B

Modelica definition

```
partial model RotationSystemTemp
  "System with rotational mechanic and hydraulic parts"

  replaceable model VolumeType =
      Hydraulic.Interfaces.FluidProps.CalcFluidTemp
  extends VolumeType "model of oil bulk modulus and kinematic viscosity";
  extends VolumeType;

  Modelica.SIunits.Angle phi "Absolute rotation angle of flange";
  Modelica.SIunits.AngularVelocity w "Absolute angular velocity of flange";
  Interfaces.Port_A port_A;
  Interfaces.Port_B port_B "Hydraulic Connector B";

  BondLib.Mechanical.Rotational.Interfaces.Flange_a flange_a
    "Rotational Connector";
  BondLib.Mechanical.Rotational.Interfaces.Flange_b flange_b
    "Rotational Connector";
  Modelica.Thermal.HeatTransfer.Interfaces.HeatPort_a heatPort;
  Modelica.SIunits.AbsolutePressure p "Pressure at port_A";
equation
  p=port_A.p;
  flange_a.phi = phi;
  flange_a.w = w;
end RotationSystemTemp;
```

Hydraulic.Interfaces.OnePortTemp.VolumeType

model of oil bulk modulus and kinematic viscosity



Parameters

Type	Name	Default	Description

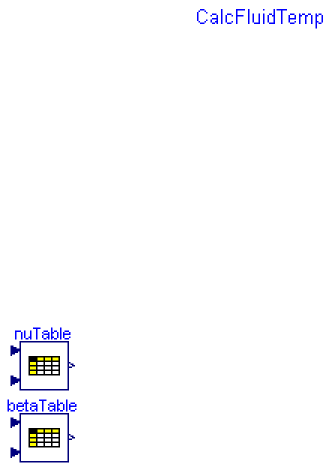
Oil		
Properties		
replaceable model Prop	FluidProp	constants of fluid models

Modelica definition

```
replaceable model VolumeType =
    Hydraulic.Interfaces.FluidProps.CalcFluidTemp
extends VolumeType "model of oil bulk modulus and kinematic viscosity";
```

Hydraulic.Interfaces.TwoPortComponentTemp.VolumeType

model of oil bulk modulus and kinematic viscosity



Parameters

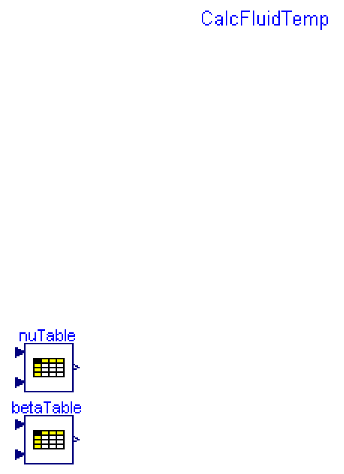
Type	Name	Default	Description
Oil			
Properties			
replaceable model Prop		FluidProp	constants of fluid models

Modelica definition

```
replaceable model VolumeType =
    Hydraulic.Interfaces.FluidProps.CalcFluidTemp
extends VolumeType "model of oil bulk modulus and kinematic viscosity";
```

Hydraulic.Interfaces.TwoPortSystemTemp.VolumeType

model of oil bulk modulus and kinematic viscosity



Parameters

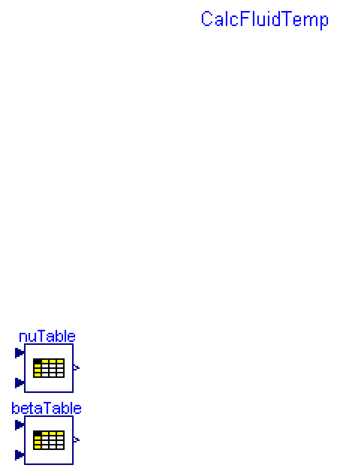
Type	Name	Default	Description
Oil			
Properties			
replaceable model	Prop	FluidProp	constants of fluid models

Modelica definition

```
replaceable model VolumeType =
    Hydraulic.Interfaces.FluidProps.CalcFluidTemp
extends VolumeType "model of oil bulk modulus and kinematic viscosity";
```

Hydraulic.Interfaces.TranslationSystemTemp.VolumeType

model of oil bulk modulus and kinematic viscosity



Parameters

Type	Name	Default	Description
Oil			

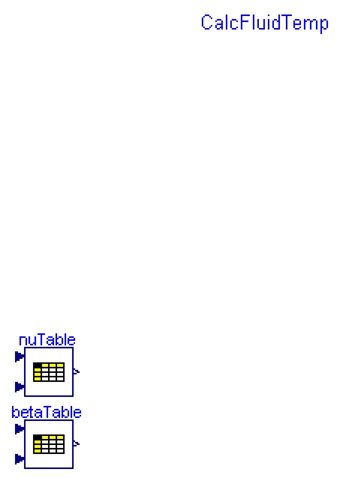
Properties		
replaceable model Prop	FluidProp	constants of fluid models

Modelica definition

```
replaceable model VolumeType =
    Hydraulic.Interfaces.FluidProps.CalcFluidTemp
extends VolumeType "model of oil bulk modulus and kinematic viscosity";
```

Hydraulic.Interfaces.RotationSystemTemp.VolumeType

model of oil bulk modulus and kinematic viscosity



Parameters

Type	Name	Default	Description
Oil			
Properties			
replaceable model Prop	FluidProp	constants of fluid models	

Modelica definition

```
replaceable model VolumeType =
    Hydraulic.Interfaces.FluidProps.CalcFluidTemp
extends VolumeType "model of oil bulk modulus and kinematic viscosity";
```

HTML-documentation generated by Dymola Thu Jul 21 09:16:22 2011.

Hydraulic.Interfaces.HydraulicUnits

package with hydraulic units

Package Content

Name	Description
Conductance	
RotPerFlow	
DispPerFlow	

Hydraulic.Interfaces.HydraulicUnits.Conductance

Parameters

Type	Name	Default	Description
	displayUnit	"m3/(s.Pa)"	

Modelica definition

```
type Conductance = Real (
    final quantity="Conductance",
    final min=0.0,
    final unit="m3/(s.Pa)",
    displayUnit="m3/(s.Pa)");
```

Hydraulic.Interfaces.HydraulicUnits.RotPerFlow

Parameters

Type	Name	Default	Description
	displayUnit	"rad/m3"	

Modelica definition

```
type RotPerFlow = Real (
    final quantity="RotPerFlow",
    final unit="rad/m3",
    displayUnit="rad/m3");
```

Hydraulic.Interfaces.HydraulicUnits.DispPerFlow

Parameters

Type	Name	Default	Description
	displayUnit	"m/m3"	

Modelica definition

```
type DispPerFlow = Real (
    final quantity="DispPerFlow",
    final unit="1/m2",
    displayUnit="m/m3");
```

HTML-documentation generated by Dymola Thu Jul 21 09:16:22 2011.

Hydraulic.Interfaces.FluidProps

Package of fluid properties

Package Content

Name	Description
CalcFluidPress	Bulk modulus and kinematic viscosity depending on pressure.
CalcFluidConst	Constant bulk modulus and viscosity.
CalcFluidTemp	Bulk modulus and viscosity depending on pressure and temperature.
HydraulicProps	Definition of constants describing the fluid.
FluidProp	Definition of default class that contains the constants which describe the fluid.

Hydraulic.Interfaces.FluidProps.CalcFluidPress

Bulk modulus and kinematic viscosity depending on pressure.

CalcFluidPress

Information

To compute the pressure in the model OilVolume an equation for the bulk modulus is needed. Model *CalcFluidPress* describes a pressure dependent bulk modulus according to Hoffmann. The pressure in the volume is calculated by:

d port_A.p / d time = beta(port_A.p) * port_A.q(time) / volume

The bulk modulus is given by:

beta = beta(port_A.p).

Variables used in above equations

port_A.p	pressure in the volume, [Pa]
beta	bulk modulus, pressure dependent, [Pa]
volume	geometric volume of oil under pressure (parameter), [m^3]
port_A.q	port_A.q, [m^3/s]

The kinematic viscosity in the volume is calculated by:

nu = nuConst

The required parameters for betapmax, pbeta1, pbeta2 and nuConst are given in Hydraulic.Interfaces.FluidProps.HydraulicProps. By using another class these parameters can easily be changed.

Release Notes

Design taken from HyLibLight

Modelica definition

```
partial model CalcFluidPress
  "Bulk modulus and kinematic viscosity depending on pressure."

  replaceable model Prop = Hydraulic.Interfaces.FluidProps.FluidProp
    "constants of fluid models";

  Modelica.SIunits.AbsolutePressure p;
  Modelica.SIunits.BulkModulus beta;
  Modelica.SIunits.Temperature T;
  Modelica.SIunits.KinematicViscosity nu;

equation
  beta = max(Prop.betapmax*(1 - Modelica.Math.exp(Prop.pbeta1 + Prop.pbeta2*
    (p+1e5))), 1e5);
  nu=Prop.nuConst;
end CalcFluidPress;
```

Hydraulic.Interfaces.FluidProps.CalcFluidConst

Constant bulk modulus and viscosity.

CalcFluidConst

Information

To compute the pressure in the model OilVolume an equation for the bulk modulus is needed. Model *CalcFluidConst* describes a constant bulk modulus. The pressure in the volume is calculated by:

d port_A.p / d time = beta(port_A.p) * port_A.q(time) / volume

The bulk modulus is given by:

beta = Prop.betaconst.

The kinematic viscositi is given by:

nu = Prop.nuConst.

Variables used in above equations

port_A.p	pressure in the volume, [Pa]
beta	bulk modulus, pressure dependent, [Pa]
volume	geometric volume of oil under pressure (parameter), [m^3]
port_A.q	port_A.q, [m^3/s]

The required parameters betaconst and nuConst are given in Hydraulic.Interfaces.FluidProps.HydraulicProps. By using another class these parameters can easily be changed.

Release Notes

Design taken from HyLibLight

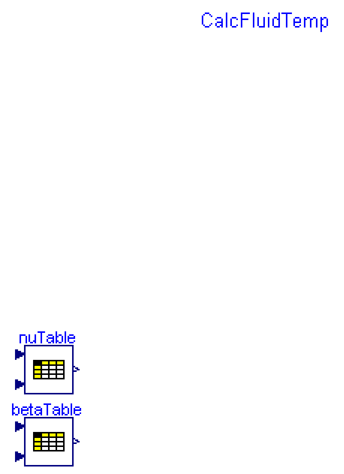
Modelica definition

```
partial model CalcFluidConst "Constant bulk modulus and viscosity."
  replaceable model Prop = Hydraulic.Interfaces.FluidProps.FluidProp
```

```
    "constants of fluid models";

  Modelica.SIunits.AbsolutePressure p;
  Modelica.SIunits.BulkModulus beta;
  Modelica.SIunits.Temperature T;
  Modelica.SIunits.KinematicViscosity nu;

equation
  beta = Prop.betaconst;
  nu=Prop.nuConst;
end CalcFluidConst;
```

Hydraulic.Interfaces.FluidProps.CalcFluidTemp

Bulk modulus and viscosity depending on pressure and temperature.



Information

To compute the pressure in the model OilVolume an equation for the bulk modulus is needed. Model *CalcFluidTemp* gives this equation of a *pressure and temperature dependent* bulk modulus according to data given in Dubbel. The pressure in the volume is calculated by:

d port_A.p / d time = beta * port_A.q(time) / volume

The bulk modulus is given by:

beta = beta(port_A.p, temperature).

The kinematic viscosity is given by:

nu = nu(port_A.p, temperature).

Variables used in above equations

port_A.p	pressure in the volume, [Pa]
beta	bulk modulus, [Pa]
temperature	temperature of oil (parameter Prop.OilTemperature), [° C]
volume	geometric volume of oil under pressure (parameter), [m^3]
port_A.q	port_A.q, [m^3/s]

The required parameters to calculate the bulk modulus and the kinematic viscosity are given in FluidProp

Release Notes

Design taken from HyLibLight

Modelica definition

```
partial model CalcFluidTemp
  "Bulk modulus and viscosity depending on pressure and temperature."
  replaceable model Prop = Hydraulic.Interfaces.FluidProps.FluidProp
    "constants of fluid models";

  Modelica.SIunits.AbsolutePressure p;
  Modelica.SIunits.Temperature T;
  Modelica.SIunits.BulkModulus beta;
  Modelica.SIunits.KinematicViscosity nu;
protected
  Modelica.Blocks.Tables.CombiTable2D betaTable(table=Prop.bulkModulus,
      smoothness=Modelica.Blocks.Types.Smoothness.ContinuousDerivative);
  Modelica.Blocks.Tables.CombiTable2D nuTable(table=Prop.kinVisc, smoothness=
        Modelica.Blocks.Types.Smoothness.ContinuousDerivative);
equation
  betaTable.u1=T;
  betaTable.u2=p;
  beta = betaTable.y;

  nuTable.u1=T;
  nuTable.u2=p;

  nu=nuTable.y;

end CalcFluidTemp;
```

Hydraulic.Interfaces.FluidProps.HydraulicProps

Definition of constants describing the fluid.

Information

Class *HydraulicProps* defines default parameters that describe the fluid, mineral hydraulic oil. The values are suitable for HLP 46.

Variable	Description
nuConst	Kinematic viscosity, [[m^2/s]
kinVisc	Table for kinematic viscosity in function of pressure and temperature, [[m^2/s]
rho	Mass density, [kg/m^3]
cp	Heat capacity of oil, [J/K]
	Parameter for Hoffmann's model of pressure dependent bulk modulus, [Pa]
pbeta1	Parameter for Hoffmann's model of pressure dependent bulk modulus
pbeta2	Parameter for Hoffmann's model of pressure dependent bulk modulus
	Parameter for constant bulk modulus, [Pa]
betaconst	Parameter for VolumeConst, i. e. constant bulk modulus, [Pa]

	Parameter for pressure and temperature dependent bulk modulus, [Pa]
bulkModulus	Table for pressure and temperature dependent bulk modulus

Release Notes

Most of the values and basic design are taken from HyLibLight

Modelica definition

```
model HydraulicProps "Definition of constants describing the fluid."


    // parameter for all HyLib models
    // standard mineral oil, e. g. HLP 46

    //"kinematic viscosity";
    // Data from Dubbel
    constant Modelica.SIunits.KinematicViscosity nuConst=46e-6;

    constant Real kinVisc[3, 3] = [0,      1e5,        701e5;
                                   298.15, 129.851e-6, 672.560e-6;
                                   353.15, 13.7612e-6, 49.3489e-6];

    constant Modelica.SIunits.Density rho=865 "mass density";

    //Heat Capacity
    constant Modelica.SIunits.HeatCapacity cp=2080 "heat capacity of oil";

    // parameter for model "OilVolume", the default model
    //  (Hoffmann's model)

    constant Modelica.SIunits.BulkModulus betapmax=1.8e9
    "bulk modulus at maximum pressure";
    constant Real pbeta1=-0.4 "parameter [ ]";
    constant Real pbeta2=-2.e-7 "parameter [1/Pa]";

    // parameter for bulkmodulus for CalcFluidConst
     constant Modelica.SIunits.BulkModulus betaconst=1.6e9
    "constant bulk modulus";

   //parameter for bulkmodulus for CalcFluidTemp from Dubbel

    constant Real bulkModulus[4, 3] =  [0,      1e5,     251e5;
                                        290.95, 2.1e9,   2.27355e9;
                                        323.15, 1.7e9,   1.93141e9;
                                        353.75, 1.371e9, 1.5529e9];

end HydraulicProps;
```

Hydraulic.Interfaces.FluidProps.FluidProp

Definition of default class that contains the constants which describe the fluid.

Information

Class *FluidProp* defines the default class that contains all parameters that describe the fluid, mineral hydraulic oil.

```
  replaceable model DefaultProps = HyLibProps;
  extends DefaultProps;
```

To change fluid parameters for one component, e. g. a lumped volume, define a model "MyProps" and use for the volume the modifier:

redeclare model Prop = MyProps

Release Notes

Taken from HyLibLight

Modelica definition

```
model FluidProp "Definition of default class that contains the
constants which describe the fluid."

protected
    replaceable model DefaultProps =
      Hydraulic.Interfaces.FluidProps.HydraulicProps;
    extends DefaultProps;
end FluidProp;
```

Hydraulic.Interfaces.FluidProps.FluidProp.DefaultProps

Modelica definition

```
replaceable model DefaultProps =
  Hydraulic.Interfaces.FluidProps.HydraulicProps;
```

HTML-documentation generated by Dymola Thu Jul 21 09:16:22 2011.

Hydraulic.Pumps

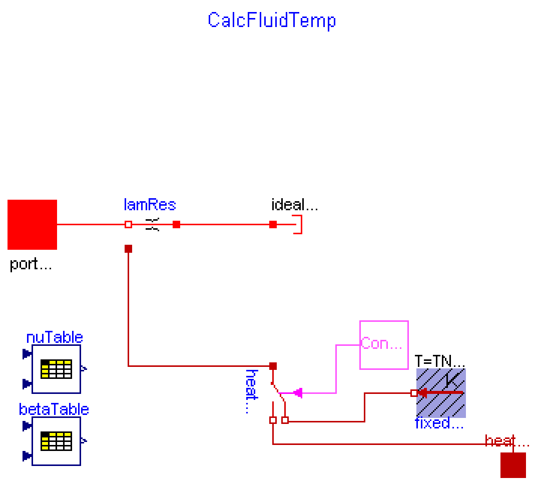
Pump models

Package Content

Name	Description
Tank	Tank with fixed pressure
ModTank	Modulated pressure source
FlowSource	Flow source
ModFlowSource	Modulated flow source
ForcedFlow	Forced flow
ModForcedFlow	Modulated forced flow
PressureSource	Pressure source
ModPressureSource	Modulated pressure source
Turbine	Rotational turbine
TransTurbine	Translational turbine
Piston	Model of a piston with leakage
TwoChamberPiston	Piston with two chambers and leakage
Ideal	Ideal pump models

Hydraulic.Pumps.Tank

Tank with fixed pressure



Information

Huge Chamber (e.g. Environment), with Constant Pressure inside and laminar resistance at the Port

Parameters

Type	Name	Default	Description

Pressure	p0	1e5	Pressure in Tank [Pa]
Conductance	G	2e-8	Conductance of Port A [m3/(s.Pa)]
Temperature	TNom	298.15	Temperature if heatPort not connected [K]
Oil			
Properties			
replaceable model VolumeType		CalcFluidTemp	model of oil bulk modulus and kinematic viscosity
replaceable model Prop		FluidProp	constants of fluid models

Connectors

Type	Name	Description
HeatPort_a	heatPort	
Oil		
Properties		
Port_A	port_A	

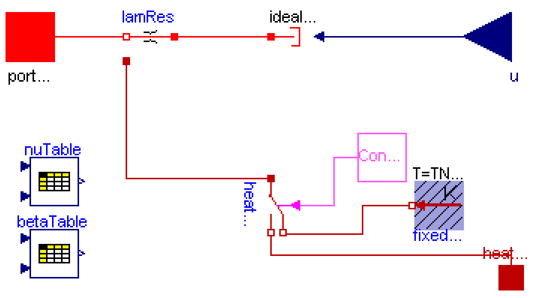
Modelica definition

```
model Tank "Tank with fixed pressure"
  extends Hydraulic.Interfaces.OnePortTemp;

  parameter Modelica.SIunits.Pressure p0=1e5 "Pressure in Tank";
  parameter Interfaces.HydraulicUnits.Conductance G=2e-8
    "Conductance of Port A";
  parameter Modelica.SIunits.Temperature TNom=298.15
    "Temperature if heatPort not connected";

  Ideal.IdealTank idealTank(p0=p0);
  Components.LamRes lamRes(G=G);
  BondLib.Thermal.HeatTransfer.Switches.HeatBranch heatBranch;
  BondLib.Thermal.HeatTransfer.Sources.FixedTemperature fixedTemperature(T=TNom);
equation
  T=heatBranch.p.T;
  if cardinality(heatPort) <= 1 then
    heatBranch.control = false;
  else
    heatBranch.control = true;
  end if;

  connect(lamRes.port_A, idealTank.port_A);
  connect(lamRes.port_B, port_A);
  connect(lamRes.heatPort, heatBranch.p);
  connect(fixedTemperature.port, heatBranch.n1);
  connect(heatBranch.n2, heatPort);
end Tank;
```

Hydraulic.Pumps.ModTank

Modulated pressure source

CalcFluidTemp



Information

Huge Chamber with controlled Pressure inside and laminar resistance at the Port

Parameters

Type	Name	Default	Description
Conductance	G	2e-8	Conductance at Port_A [m3/(s.Pa)]
Temperature	TNom	298.15	Temperature if heatPort not connected [K]
Oil			
Properties			
replaceable model VolumeType	CalcFluidTemp	model of oil bulk modulus and kinematic viscosity
replaceable model Prop		FluidProp	constants of fluid models

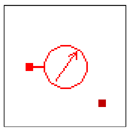
Connectors

Type	Name	Description
HeatPort_a	heatPort	
input RealInput	u	Pressure in Chamber
Oil		
Properties		
Port_A	port_A	

Modelica definition

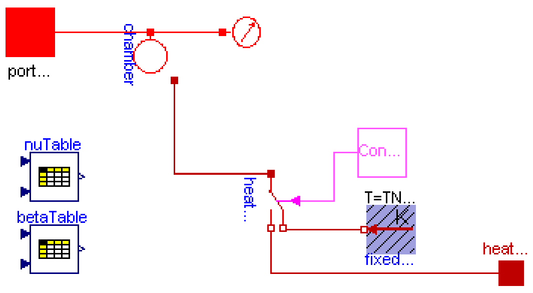
```
model ModTank "Modulated pressure source"
  extends Hydraulic.Interfaces.OnePortTemp;

  parameter Interfaces.HydraulicUnits.Conductance G=2e-8
    "Conductance at Port_A";
  parameter Modelica.SIunits.Temperature TNom=298.15
    "Temperature if heatPort not connected";

  Ideal.IdealModTank idealTank;
  Components.LamRes lamRes(G=G);
  Modelica.Blocks.Interfaces.RealInput u "Pressure in Chamber";
  BondLib.Thermal.HeatTransfer.Switches.HeatBranch heatBranch;
  BondLib.Thermal.HeatTransfer.Sources.FixedTemperature fixedTemperature(T=TNom);
equation
  T=heatBranch.p.T;
  if cardinality(heatPort) <= 1 then
    heatBranch.control = false;
  else
    heatBranch.control = true;
  end if;
  connect(lamRes.port_A, idealTank.port_A);
  connect(lamRes.port_B, port_A);
  connect(idealTank.u, u);
  connect(lamRes.heatPort,heatBranch. p);
  connect(fixedTemperature.port, heatBranch.n1);
  connect(heatBranch.n2, heatPort);
end ModTank;
```

Hydraulic.Pumps.FlowSource



Flow source

CalcFluidTemp



Information

Flow Source with losses at outflow

Parameters

Type	Name	Default	Description
VolumeFlowRate	f0	0	Volumetric Flow Rate [m3/s]
Volume	V0	1e-6	Volume of Outflow [m3]
Temperature	TNom	298.15	Temperature if heatPort not connected [K]
Oil			
Properties			
replaceable model VolumeType		CalcFluidTemp	model of oil bulk modulus and kinematic viscosity
replaceable model Prop		FluidProp	constants of fluid models

Connectors

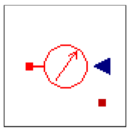
Type	Name	Description
HeatPort_a	heatPort	
Oil		
Properties		

Port_A	port_A	

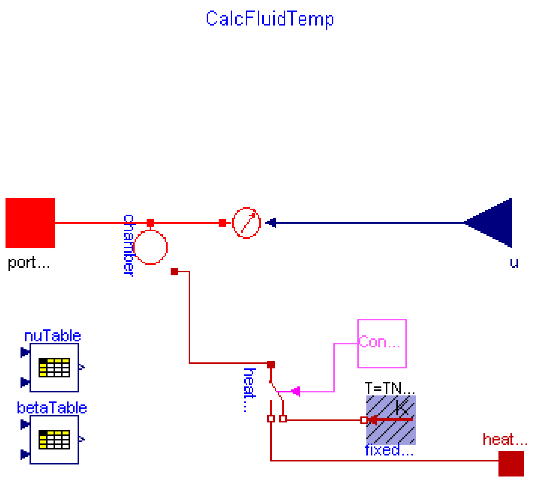
Modelica definition

```
model FlowSource "Flow source"
  extends Hydraulic.Interfaces.OnePortTemp;

  parameter Modelica.SIunits.VolumeFlowRate f0=0 "Volumetric Flow Rate";
  parameter Modelica.SIunits.Volume V0=1e-6 "Volume of Outflow";
  parameter Modelica.SIunits.Temperature TNom=298.15
    "Temperature if heatPort not connected";
  Ideal.IdealFlowSource idealFlowSource(f0=f0);

  Components.Chamber chamber(V=V0);
  BondLib.Thermal.HeatTransfer.Switches.HeatBranch heatBranch;
  BondLib.Thermal.HeatTransfer.Sources.FixedTemperature fixedTemperature(T=TNom);
equation
  T=heatBranch.p.T;
  if cardinality(heatPort) <= 1 then
    heatBranch.control = false;
  else
    heatBranch.control = true;
  end if;

  connect(chamber.port_A, idealFlowSource.port_A);
  connect(chamber.port_A, port_A);
  connect(heatBranch.n2,heatPort);
  connect(fixedTemperature.port,heatBranch. n1);
  connect(chamber.heatPort, heatBranch.p);
end FlowSource;
```

Hydraulic.Pumps.ModFlowSource



Modulated flow source



Information

Modulated flow source with losses at outflow

Parameters

Type	Name	Default	Description
Volume	V0	1e-6	Volume at Port [m3]
Temperature	TNom	298.15	Temperature if heatPort not connected [K]

Oil		
Properties		
replaceable model VolumeType	CalcFluidTemp	model of oil bulk modulus and kinematic viscosity
replaceable model Prop	FluidProp	constants of fluid models

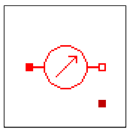
Connectors

Type	Name	Description
HeatPort_a	heatPort	
input RealInput	u	
Oil		
Properties		
Port_A	port_A	

Modelica definition

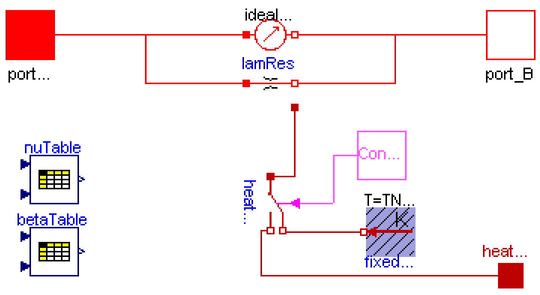
```
model ModFlowSource "Modulated flow source"
  extends Hydraulic.Interfaces.OnePortTemp;

  parameter Modelica.SIunits.Volume V0=1e-6 "Volume at Port";
  parameter Modelica.SIunits.Temperature TNom=298.15
    "Temperature if heatPort not connected";
  Ideal.IdealModFlowSource idealFlowSource;
  Modelica.Blocks.Interfaces.RealInput u;
  Components.Chamber chamber(V=V0);
  BondLib.Thermal.HeatTransfer.Switches.HeatBranch heatBranch;
  BondLib.Thermal.HeatTransfer.Sources.FixedTemperature fixedTemperature(T=TNom);
equation
  T=heatBranch.p.T;
  if cardinality(heatPort) <= 1 then
    heatBranch.control = false;
  else
    heatBranch.control = true;
  end if;

  connect(idealFlowSource.u, u);
  connect(chamber.port_A, idealFlowSource.port_A);
  connect(chamber.port_A, port_A);
  connect(fixedTemperature.port,heatBranch. n1);
  connect(chamber.heatPort, heatBranch.p);
  connect(heatBranch.n2, heatPort);
end ModFlowSource;
```

Hydraulic.Pumps.ForcedFlow



Forced flow

CalcFluidTemp



Information

Forced flow with leakage

Parameters

Type	Name	Default	Description
VolumeFlowRate	q0	1	Forced Volume Flowrate [m3/s]
Conductance	G	1e-20	Parallel Conductance [m3/(s.Pa)]
Temperature	TNom	298.15	Temperature if heatPort not connected [K]
Oil			
Properties			
replaceable model VolumeType	CalcFluidTemp		model of oil bulk modulus and kinematic viscosity
replaceable model Prop	FluidProp		constants of fluid models

Connectors

Type	Name	Description
HeatPort_a	heatPort	
Oil		
Properties		
Port_A	port_A	
Port_B	port_B	Hydraulic Connector B

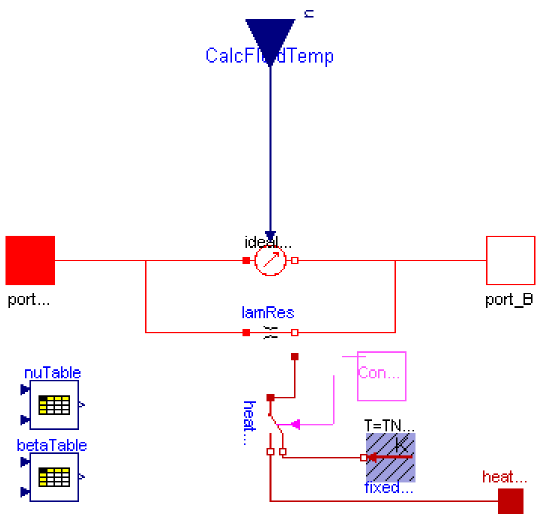
Modelica definition

```
model ForcedFlow "Forced flow"
  extends Hydraulic.Interfaces.TwoPortComponentTemp;

  parameter Modelica.SIunits.VolumeFlowRate q0=1  "Forced Volume Flowrate";
  parameter Interfaces.HydraulicUnits.Conductance G=1e-20
    "Parallel Conductance";
  parameter Modelica.SIunits.Temperature TNom=298.15
    "Temperature if heatPort not connected";


  Ideal.IdealForcedFlow idealForcedFlow(q0=q0);
  Components.LamRes lamRes(G=G);
  BondLib.Thermal.HeatTransfer.Switches.HeatBranch heatBranch;
  BondLib.Thermal.HeatTransfer.Sources.FixedTemperature fixedTemperature(T=TNom);
```

```
equation
  T=heatBranch.p.T;
  if cardinality(heatPort) <= 1 then
    heatBranch.control = false;
  else
    heatBranch.control = true;
  end if;
  connect(lamRes.port_B, port_B);
  connect(idealForcedFlow.port_B, port_B);
  connect(lamRes.port_A, port_A);
  connect(idealForcedFlow.port_A, port_A);
  connect(fixedTemperature.port, heatBranch.n1);
  connect(heatBranch.n2, heatPort);
  connect(lamRes.heatPort, heatBranch.p);
end ForcedFlow;
```

Hydraulic.Pumps.ModForcedFlow

Modulated forced flow



Information

Modulated forced flow with leakage

Parameters

Type	Name	Default	Description
VolumeFlowRate	f0	1	[m3/s]
Conductance	G	1e-20	[m3/(s.Pa)]
Temperature	TNom	298.15	[K]
Oil			
Properties			
replaceable model VolumeType	CalcFluidTemp		model of oil bulk modulus and kinematic viscosity
replaceable model Prop	FluidProp		constants of fluid models

Connectors

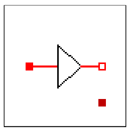
Type	Name	Description
HeatPort_a	heatPort	

input RealInput	u	
Oil		
Properties		
Port_A	port_A	
Port_B	port_B	Hydraulic Connector B

Modelica definition

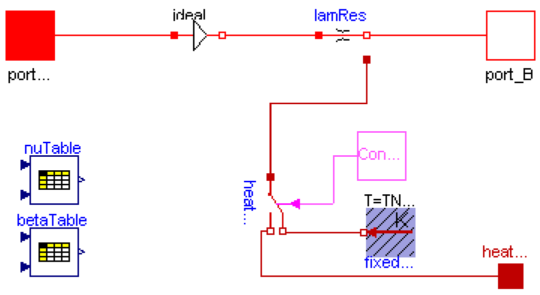
```
model ModForcedFlow "Modulated forced flow"
  extends Hydraulic.Interfaces.TwoPortComponentTemp;

  parameter Modelica.SIunits.VolumeFlowRate f0=1;
  parameter Interfaces.HydraulicUnits.Conductance G=1e-20;
  parameter Modelica.SIunits.Temperature TNom=298.15;

  Ideal.IdealModForcedFlow idealForcedFlow;
  Components.LamRes lamRes(G=G);
  Modelica.Blocks.Interfaces.RealInput u;
  BondLib.Thermal.HeatTransfer.Switches.HeatBranch heatBranch;
  BondLib.Thermal.HeatTransfer.Sources.FixedTemperature fixedTemperature(T=TNom);
equation
  T=heatBranch.p.T;
  if cardinality(heatPort) <= 1 then
    heatBranch.control = false;
  else
    heatBranch.control = true;
  end if;
  connect(lamRes.port_B, port_B);
  connect(idealForcedFlow.port_B, port_B);
  connect(lamRes.port_A, port_A);
  connect(idealForcedFlow.port_A, port_A);
  connect(idealForcedFlow.u, u);
  connect(fixedTemperature.port,heatBranch. n1);
  connect(heatBranch.n2, heatPort);
  connect(heatBranch.p, lamRes.heatPort);
end ModForcedFlow;
```

Hydraulic.Pumps.PressureSource



Pressure source

CalcFluidTemp



Information

Pressure source with small resistance

Parameters

Type	Name	Default	Description
Pressure	p0	1e5	Pressure drop from port_B to port_A [Pa]
Conductance	G	1	Conductance of Pressuresource [m3/(s.Pa)]
Temperature	TNom	298.15	Temperature if heatPort not connected [K]
Oil			
Properties			
replaceable model VolumeType	CalcFluidTemp		model of oil bulk modulus and kinematic viscosity
replaceable model Prop	FluidProp		constants of fluid models

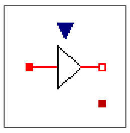
Connectors

Type	Name	Description
HeatPort_a	heatPort	
Oil		
Properties		
Port_A	port_A	
Port_B	port_B	Hydraulic Connector B

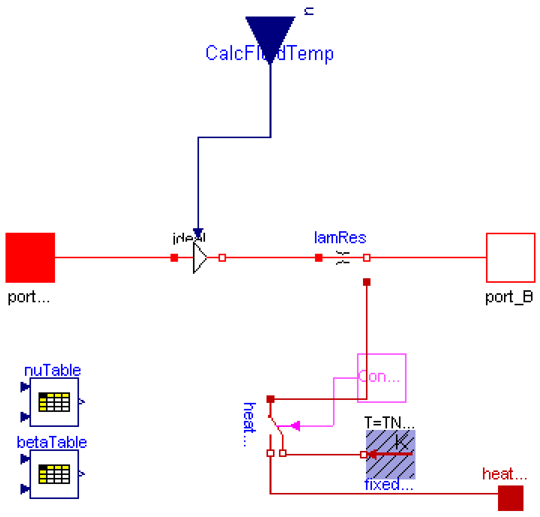
Modelica definition

```
model PressureSource "Pressure source"
  extends Hydraulic.Interfaces.TwoPortComponentTemp;

  parameter Modelica.SIunits.Pressure p0=1e5
    "Pressure drop from port_B to port_A";
  parameter Interfaces.HydraulicUnits.Conductance G=1
    "Conductance of Pressuresource";
    parameter Modelica.SIunits.Temperature TNom=298.15
    "Temperature if heatPort not connected";

  Ideal.IdealPressureSource idealPressureSource(p0=p0);
  Components.LamRes lamRes(G=G);
  BondLib.Thermal.HeatTransfer.Switches.HeatBranch heatBranch;
  BondLib.Thermal.HeatTransfer.Sources.FixedTemperature fixedTemperature(T=TNom);
equation
  T=heatBranch.p.T;
  if cardinality(heatPort) <= 1 then
    heatBranch.control = false;
  else
    heatBranch.control = true;
  end if;
  connect(idealPressureSource.port_A, port_A);
  connect(lamRes.port_A, idealPressureSource.port_B);
  connect(lamRes.port_B, port_B);
  connect(fixedTemperature.port,heatBranch. n1);
  connect(heatBranch.n2, heatPort);
  connect(lamRes.heatPort,heatBranch. p);
end PressureSource;
```

Hydraulic.Pumps.ModPressureSource



Modulated pressure source



Information

Modulated pressure source with small resistance

Parameters

Type	Name	Default	Description
Conductance	G	1	Conductance of pressure source [m3/(s.Pa)]
Temperature	TNom	298.15	Temperature if heatPort not connected [K]
Oil			
Properties			
replaceable model VolumeType		CalcFluidTemp	model of oil bulk modulus and kinematic viscosity
replaceable model Prop		FluidProp	constants of fluid models

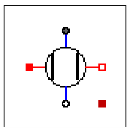
Connectors

Type	Name	Description
HeatPort_a	heatPort	
input RealInput	u	
Oil		
Properties		
Port_A	port_A	
Port_B	port_B	Hydraulic Connector B

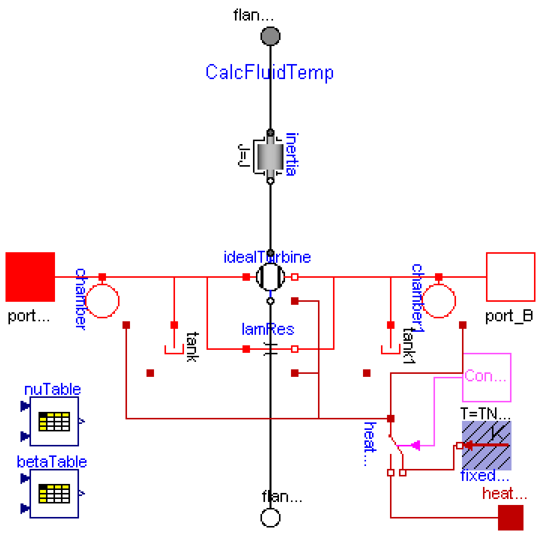
Modelica definition

```
model ModPressureSource "Modulated pressure source"
  extends Hydraulic.Interfaces.TwoPortComponentTemp;

  parameter Interfaces.HydraulicUnits.Conductance G=1
    "Conductance of pressure source";
  parameter Modelica.SIunits.Temperature TNom=298.15
    "Temperature if heatPort not connected";

  Ideal.IdealModPressureSource idealPressureSource;
  Components.LamRes lamRes;
  Modelica.Blocks.Interfaces.RealInput u;
  BondLib.Thermal.HeatTransfer.Switches.HeatBranch heatBranch;
  BondLib.Thermal.HeatTransfer.Sources.FixedTemperature fixedTemperature(T=TNom);
equation
  T=heatBranch.p.T;
  if cardinality(heatPort) <= 1 then
    heatBranch.control = false;
  else
    heatBranch.control = true;
  end if;
  connect(idealPressureSource.port_A, port_A);
  connect(lamRes.port_A, idealPressureSource.port_B);
  connect(lamRes.port_B, port_B);
  connect(idealPressureSource.u, u);
  connect(fixedTemperature.port,heatBranch. n1);
  connect(heatBranch.n2, heatPort);
  connect(lamRes.heatPort,heatBranch. p);
end ModPressureSource;
```

Hydraulic.Pumps.Turbine



Rotational turbine



Information

Rotational turbine with mechanical mass, hydraulic volume and hydraulic losses

Parameters

Type	Name	Default	Description
RotPerFlow	RotFactor	6.2e3	Rotation in rad per m3 of fluid [rad/m3]
Volume	V	2e-6	hydraulic Volume of Turbine [m3]
Conductance	GExt	2e-13	Conductance of the turbine [m3/(s.Pa)]
Conductance	GInt	1e-13	Conductanc of leakage along turbine [m3/(s.Pa)]
Inertia	J	1	Inertia of turbine [kg.m2]
Temperature	TNom	298.15	Temperature if heatPort not connected [K]
Oil			
Properties			
replaceable model VolumeType	CalcFluidTemp	model of oil bulk modulus and kinematic viscosity	
replaceable model Prop	FluidProp	constants of fluid models	

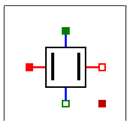
Connectors

Type	Name	Description
Flange_a	flange_a	Rotational Connector
Flange_b	flange_b	Rotational Connector
HeatPort_a	heatPort	
Oil		
Properties		
Port_A	port_A	
Port_B	port_B	Hydraulic Connector B

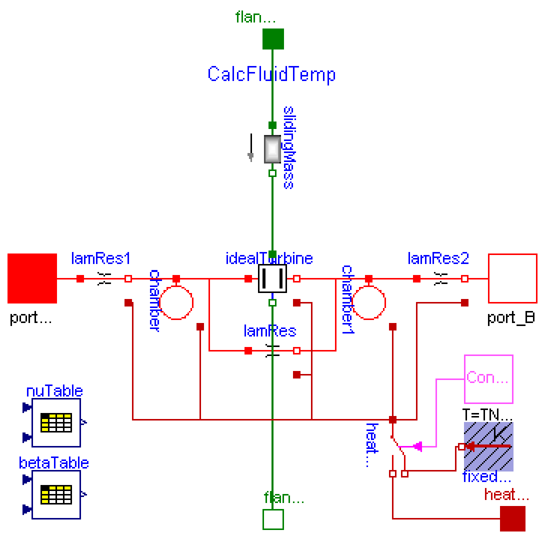
Modelica definition

```
model Turbine "Rotational turbine"
  extends Hydraulic.Interfaces.RotationSystemTemp;

  parameter Interfaces.HydraulicUnits.RotPerFlow RotFactor = 6.2e3
    "Rotation in rad per m3 of fluid";
  parameter Modelica.SIunits.Volume V=2e-6 "hydraulic Volume of Turbine";
  parameter Interfaces.HydraulicUnits.Conductance GExt=2e-13
    "Conductance of the turbine";
  parameter Interfaces.HydraulicUnits.Conductance GInt=1e-13
    "Conductanc of leakage along turbine";
  parameter Modelica.SIunits.Inertia J=1 "Inertia of turbine";
  parameter Modelica.SIunits.Temperature TNom=298.15
    "Temperature if heatPort not connected";
  Ideal.IdealTurbine idealTurbine(trans_factor=RotFactor);
  Components.LamRes lamRes(G=GInt);
  BondLib.Mechanical.Rotational.Passive.Inertia inertia(J=J);


  Components.Chamber chamber(V=V/2);
  Components.Chamber chamber1(V=V/2);
  BondLib.Thermal.HeatTransfer.Switches.HeatBranch heatBranch;
  BondLib.Thermal.HeatTransfer.Sources.FixedTemperature fixedTemperature(T=TNom);
  Tank tank(G=GExt);
  Tank tank1(G=GExt);
equation
  T=heatBranch.p.T;
  if cardinality(heatPort) <= 1 then
    heatBranch.control = false;
  else
    heatBranch.control = true;
  end if;
  connect(idealTurbine.flange_b, flange_b);
  connect(inertia.flange_a, flange_a);
  connect(inertia.flange_b, idealTurbine.flange_a);
  connect(chamber.port_A, idealTurbine.port_A);
  connect(chamber.port_A, lamRes.port_A);
  connect(chamber1.port_A, lamRes.port_B);
  connect(idealTurbine.port_B, chamber1.port_A);
  connect(fixedTemperature.port,heatBranch. n1);
  connect(chamber1.heatPort, heatBranch.p);
  connect(idealTurbine.heatPort, heatBranch.p);
  connect(lamRes.heatPort, heatBranch.p);
  connect(chamber.heatPort, heatBranch.p);
  connect(heatBranch.n2, heatPort);
  connect(chamber.port_A, port_A);
  connect(chamber1.port_A, port_B);
  connect(tank.port_A, chamber.port_A);
  connect(tank1.port_A, chamber1.port_A);
end Turbine;
```

Hydraulic.Pumps.TransTurbine



Translational turbine



Information

Translational turbine with mechanical mass, hydraulic volume and hydraulic losses

Parameters

Type	Name	Default	Description
DispPerFlow	TransFactor	1	Displacement in m per m3 of fluid [1/m2]
Volume	V	1e-3	Volume of turbine [m3]
Conductance	G0	2e-8	Conductance of turbine [m3/(s.Pa)]
Conductance	G1	1e-20	Conductance of leakage along turbine [m3/(s.Pa)]
Mass	M	100	Mass of turbine [kg]
Temperature	TNom	298.15	Temperature if heatPort not connected [K]
Oil			
Properties			
replaceable model VolumeType		CalcFluidTemp	model of oil bulk modulus and kinematic viscosity
replaceable model Prop		FluidProp	constants of fluid models

Connectors

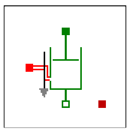
Type	Name	Description
Flange_a	flange_a	Translational Connector
Flange_b	flange_b	Translational Connector
HeatPort_a	heatPort	
Oil		
Properties		
Port_A	port_A	
Port_B	port_B	Hydraulic Connector B

Modelica definition

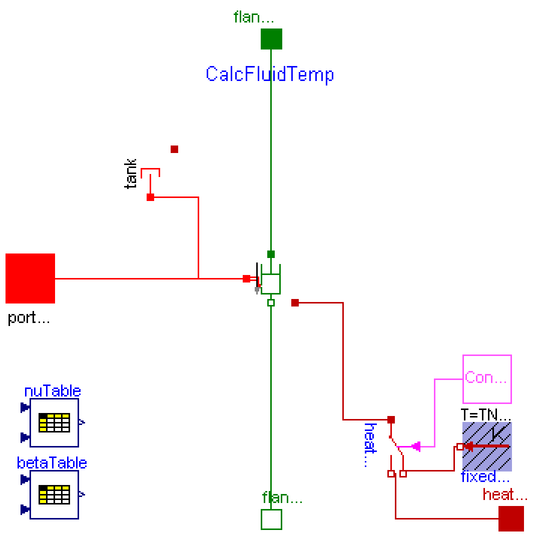
```
model TransTurbine "Translational turbine"
  extends Hydraulic.Interfaces.TranslationSystemTemp;
  parameter Interfaces.HydraulicUnits.DispPerFlow TransFactor = 1
```

```
    "Displacement in m per m3 of fluid";
  parameter Modelica.SIunits.Volume V=1e-3 "Volume of turbine";
  parameter Interfaces.HydraulicUnits.Conductance G0=2e-8
    "Conductance of turbine";
  parameter Interfaces.HydraulicUnits.Conductance G1=1e-20
    "Conductance of leakage along turbine";
  parameter Modelica.SIunits.Mass M=100 "Mass of turbine";
  parameter Modelica.SIunits.Temperature TNom=298.15
    "Temperature if heatPort not connected";
  Hydraulic.Pumps.Ideal.IdealTransTurbine idealTurbine(
                                        rot_factor=TransFactor);
  Components.LamRes lamRes(G=G1);
  Components.LamRes lamRes1(G=G0/2);
  Components.LamRes lamRes2(G=G0/2);
  BondLib.Mechanical.Translational.Passive.SlidingMass slidingMass(m=M);
  Components.Chamber chamber(V=V/2);
  Components.Chamber chamber1(V=V/2);
  BondLib.Thermal.HeatTransfer.Switches.HeatBranch heatBranch;
  BondLib.Thermal.HeatTransfer.Sources.FixedTemperature fixedTemperature(T=TNom);
equation
  T=heatBranch.p.T;
  if cardinality(heatPort) <= 1 then
    heatBranch.control = false;
  else
    heatBranch.control = true;
  end if;

  connect(lamRes1.port_A, port_A);
  connect(lamRes2.port_B, port_B);
  connect(slidingMass.flange_a, flange_a);
  connect(slidingMass.flange_b, idealTurbine.flange_a);
  connect(idealTurbine.flange_b, flange_b);
  connect(lamRes2.port_A, chamber1.port_A);
  connect(chamber1.port_A, idealTurbine.port_B);
  connect(chamber1.port_A, lamRes.port_B);
  connect(chamber.port_A, lamRes.port_A);
  connect(idealTurbine.port_A, chamber.port_A);
  connect(lamRes1.port_B, chamber.port_A);
  connect(heatBranch.n2, heatPort);
  connect(fixedTemperature.port,heatBranch. n1);
  connect(heatBranch.p, lamRes2.heatPort);
  connect(chamber1.heatPort, heatBranch.p);
  connect(idealTurbine.heatPort, heatBranch.p);
  connect(lamRes.heatPort, heatBranch.p);
  connect(chamber.heatPort, heatBranch.p);
  connect(lamRes1.heatPort, heatBranch.p);
end TransTurbine;
```

Hydraulic.Pumps.Piston



Model of a piston with leakage



Information

One-sided piston with hydraulic losses

Parameters

Type	Name	Default	Description
Area	A	1e-2	Area of piston [m2]
Conductance	GLeak	1e-20	[m3/(s.Pa)]
Length	xNom	0.5	Start length of piston [m]
Temperature	TNom	298.15	Temperature if heatPort not connected [K]
Oil			
Properties			
replaceable model VolumeType	CalcFluidTemp		model of oil bulk modulus and kinematic viscosity
replaceable model Prop	FluidProp		constants of fluid models

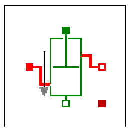
Connectors

Type	Name	Description
HeatPort_a	heatPort	
Flange_a	flange_a	
Flange_b	flange_b	
Oil		
Properties		
Port_A	port_A	

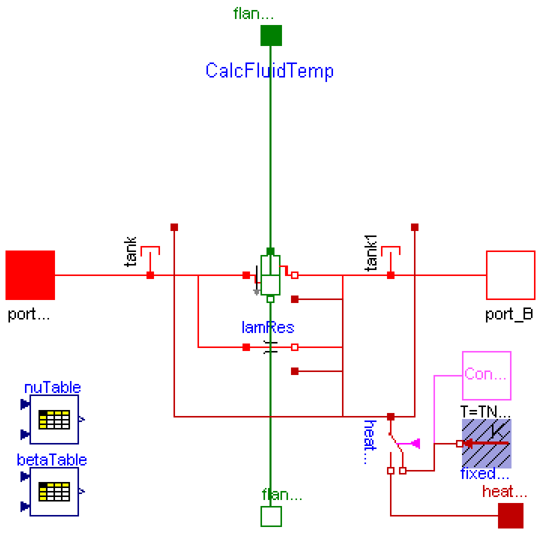
Modelica definition

```
model Piston "Model of a piston with leakage"
  extends Hydraulic.Interfaces.OnePortTemp;

  parameter Modelica.SIunits.Area A=1e-2 "Area of piston";
  parameter Interfaces.HydraulicUnits.Conductance GLeak=1e-20;
  parameter Modelica.SIunits.Length xNom=0.5 "Start length of piston";
  parameter Modelica.SIunits.Temperature TNom=298.15
    "Temperature if heatPort not connected";
  Modelica.SIunits.Length x;
  BondLib.Mechanical.Translational.Interfaces.Flange_a flange_a;
  BondLib.Mechanical.Translational.Interfaces.Flange_b flange_b;

  Ideal.IdealPiston idealPiston(redeclare model Prop = Prop, redeclare model
      VolumeType = VolumeType,
    A=A,
    TNom=TNom,
    xNom=xNom);
  BondLib.Thermal.HeatTransfer.Switches.HeatBranch heatBranch;
  BondLib.Thermal.HeatTransfer.Sources.FixedTemperature fixedTemperature(T=
        TNom);
  Tank tank(G=GLeak);
equation
  x=idealPiston.x;
  T=heatBranch.p.T;
  if cardinality(heatPort) <= 1 then
    heatBranch.control = false;
  else
    heatBranch.control = true;
  end if;
  connect(idealPiston.flange_a, flange_a);
  connect(idealPiston.flange_b, flange_b);
  connect(port_A, idealPiston.port_A);
  connect(fixedTemperature.port,heatBranch. n1);
  connect(heatBranch.p, idealPiston.heatPort);
  connect(heatBranch.n2, heatPort);
  connect(tank.port_A, idealPiston.port_A);
end Piston;
```

Hydraulic.Pumps.TwoChamberPiston



Piston with two chambers and leakage



Parameters

Type	Name	Default	Description
Area	A	1	[m2]
Length	L	1	[m]
Conductance	GLeakEnv		Conductance of leakage relative to environment [m3/(s.Pa)]
Conductance	GLeak		Conductance of leakage between the two chambers [m3/(s.Pa)]
Length	xNom	0.5	Start length of piston [m]

Temperature	pNom	1e5	Pressure at start [K]
Temperature	TNom	298.15	Temperature if heatPort not connected [K]
Oil			
Properties			
replaceable model VolumeType	CalcFluidTemp	model of oil bulk modulus and kinematic viscosity	
replaceable model Prop	FluidProp	constants of fluid models	

Connectors

Type	Name	Description
Flange_a	flange_a	Translational Connector
Flange_b	flange_b	Translational Connector
HeatPort_a	heatPort	
Oil		
Properties		
Port_A	port_A	
Port_B	port_B	Hydraulic Connector B

Modelica definition

```
model TwoChamberPiston "Piston with two chambers and leakage"
  extends Hydraulic.Interfaces.TranslationSystemTemp;
  parameter Modelica.SIunits.Area A=1;
  parameter Modelica.SIunits.Length L=1;
  parameter Interfaces.HydraulicUnits.Conductance GLeakEnv
    "Conductance of leakage relative to environment";
  parameter Interfaces.HydraulicUnits.Conductance GLeak
    "Conductance of leakage between the two chambers";
  parameter Modelica.SIunits.Length xNom=0.5 "Start length of piston";
  parameter Modelica.SIunits.Temperature pNom=1e5 "Pressure at start";
  parameter Modelica.SIunits.Temperature TNom=298.15
    "Temperature if heatPort not connected";
  Modelica.SIunits.Length x;
  Hydraulic.Pumps.Ideal.IdealTwoChamberPiston idealTwoChamberPiston(
    A=A,
    L=L,
    xNom=xNom,
    pNom=pNom,
    TNom=TNom);
  BondLib.Thermal.HeatTransfer.Switches.HeatBranch heatBranch;
  BondLib.Thermal.HeatTransfer.Sources.FixedTemperature fixedTemperature(T=TNom);
  Components.LamRes lamRes(G=GLeak);
  Tank tank(G=GLeakEnv);
  Tank tank1(G=GLeakEnv);
equation
  x=idealTwoChamberPiston.x;
  T=heatBranch.p.T;
  if cardinality(heatPort) <= 1 then
    heatBranch.control = false;
  else
    heatBranch.control = true;
  end if;
  connect(fixedTemperature.port, heatBranch.n1);
  connect(heatBranch.n2, heatPort);
  connect(tank1.port_A, port_B);
  connect(tank1.port_A, idealTwoChamberPiston.port_B);
  connect(idealTwoChamberPiston.port_A, tank.port_A);
  connect(tank.port_A, port_A);
  connect(tank.port_A, lamRes.port_A);
  connect(lamRes.port_B, tank1.port_A);
  connect(idealTwoChamberPiston.flange_a, flange_a);
  connect(idealTwoChamberPiston.flange_b, flange_b);
  connect(lamRes.heatPort, heatBranch.p);
  connect(idealTwoChamberPiston.heatPort, heatBranch.p);
  connect(tank1.heatPort, heatBranch.p);
  connect(tank.heatPort, heatBranch.p);
end TwoChamberPiston;
```

HTML-documentation generated by Dymola Thu Jul 21 09:16:29 2011.

Hydraulic.Pumps.Ideal

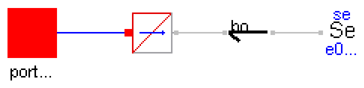
Ideal pump models

Package Content

Name	Description
IdealTank	Huge Chamber (e.g. environment)
IdealModTank	Modulated huge chamber (e.g. environment)
IdealFlowSource	Ideal flow source
IdealModFlowSource	Modulated ideal flow source
IdealPressureSource	Ideal pressure source
IdealModPressureSource	Modulated ideal pressure source
IdealForcedFlow	Ideal forced flow
IdealModForcedFlow	Modulated ideal forced flow
IdealTurbine	Rotational turbine with no losses
IdealTransTurbine	Ideal translational turbine with no losses
IdealPiston	ideal one-chamber piston
IdealTwoChamberPiston	Piston with two chambers

Hydraulic.Pumps.Ideal.IdealTank

Huge Chamber (e.g. environment)



Information

Model of an ideal tank. Pressure is constant

Parameters

Type	Name	Default	Description
Pressure	p0	1e5	[Pa]

Connectors

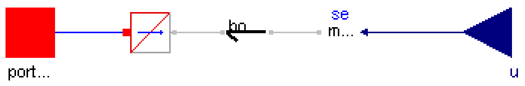
Type	Name	Description
Oil		
Properties		
Port_A	port_A	Hydraulic Connector

Modelica definition

```
model IdealTank "Huge Chamber (e.g. environment)"
  extends Hydraulic.Interfaces.OnePort;

  parameter Modelica.SIunits.Pressure p0=1e5;
  Hydraulic.Interfaces.Hy2BG hy4BG;

  BondLib.Bonds.Bond bond;
  BondLib.Sources.Se se(e0=p0);
equation
  connect(hy4BG.port_A, port_A);
  connect(bond.BondCon2, hy4BG.bondCon);
  connect(bond.BondCon1, se.BondCon1);
end IdealTank;
```

Hydraulic.Pumps.Ideal.IdealModTank

Modulated huge chamber (e.g. environment)



Information

Model of an modulated ideal tank. Pressure is equal to the real input.

Connectors

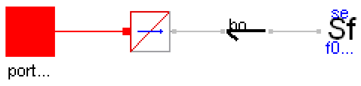
Type	Name	Description
input RealInput	u	
Oil		
Properties		
Port_A	port_A	Hydraulic Connector

Modelica definition

```
model IdealModTank "Modulated huge chamber (e.g. environment)"
  extends Hydraulic.Interfaces.OnePort;

  Hydraulic.Interfaces.Hy2BG hy4BG;
  BondLib.Bonds.Bond bond;
  BondLib.Sources.mSe se;
  Modelica.Blocks.Interfaces.RealInput u;
equation
  connect(hy4BG.port_A, port_A);
  connect(bond.BondCon2, hy4BG.bondCon);
  connect(bond.BondCon1, se.BondCon1);
  connect(se.s, u);
end IdealModTank;
```

Hydraulic.Pumps.Ideal.IdealFlowSource

Ideal flow source



Information

Model of an ideal flow source. The flow of oil is constant.

Parameters

Type	Name	Default	Description
VolumeFlowRate	f0	0	[m3/s]

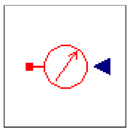
Connectors

Type	Name	Description
Oil		
Properties		
Port_A	port_A	Hydraulic Connector

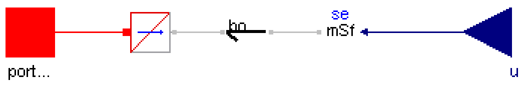
Modelica definition

```
model IdealFlowSource "Ideal flow source"
  extends Hydraulic.Interfaces.OnePort;
  parameter Modelica.SIunits.VolumeFlowRate f0=0;
  Interfaces.Hy2BG hy4BG;
  BondLib.Bonds.Bond bond;
  BondLib.Sources.Sf se(f0=f0);
equation
  connect(bond.BondCon2,hy4BG. bondCon);
  connect(bond.BondCon1,se. BondCon1);
  connect(hy4BG.port_A, port_A);
end IdealFlowSource;
```

Hydraulic.Pumps.Ideal.IdealModFlowSource



Modulated ideal flow source



Information

Model of a modulated ideal flow source. The flow of oil is equal to the input.

Connectors

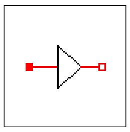
Type	Name	Description
input RealInput	u	
Oil		
Properties		
Port_A	port_A	Hydraulic Connector

Modelica definition

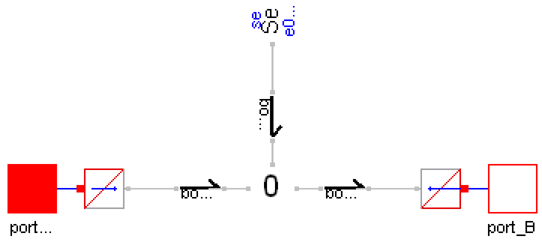
```
model IdealModFlowSource "Modulated ideal flow source"
  extends Hydraulic.Interfaces.OnePort;
  Interfaces.Hy2BG hy4BG;
  BondLib.Bonds.Bond bond;
  BondLib.Sources.mSf se;
  Modelica.Blocks.Interfaces.RealInput u;
equation
  connect(bond.BondCon2,hy4BG. bondCon);
  connect(bond.BondCon1,se. BondCon1);
  connect(hy4BG.port_A, port_A);
  connect(se.s, u);
```

```
end IdealModFlowSource;
```

Hydraulic.Pumps.Ideal.IdealPressureSource



Ideal pressure source



Information

Model of an ideal pressure source. the flow at the left is equal in both sides.

Parameters

Type	Name	Default	Description
Pressure	p0	1e5	[Pa]

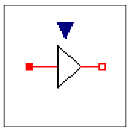
Connectors

Type	Name	Description
Oil		
Properties		
Port_A	port_A	
Port_B	port_B	Hydraulic Connector B

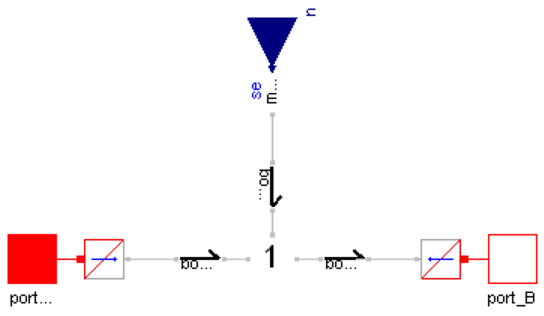
Modelica definition

```
model IdealPressureSource "Ideal pressure source"
  extends Hydraulic.Interfaces.TwoPortComponent;

  parameter Modelica.SIunits.Pressure p0=1e5;

  Hydraulic.Interfaces.Hy2BG hy4BG;
  Hydraulic.Interfaces.Hy2BG hy4BG1;

  BondLib.Bonds.Bond bond;
  BondLib.Bonds.Bond bond1;
  BondLib.Junctions.J0p3 j1p3_1;
  BondLib.Bonds.Bond bond2;
  BondLib.Sources.Se se(e0=p0);
equation
  connect(hy4BG1.port_A, port_B);
  connect(hy4BG.port_A, port_A);
  connect(bond.BondCon1, hy4BG.bondCon);
  connect(bond1.BondCon2, hy4BG1.bondCon);
  connect(j1p3_1.BondCon2, bond.BondCon2);
  connect(j1p3_1.BondCon1, bond1.BondCon1);
  connect(bond2.BondCon2, j1p3_1.BondCon3);
  connect(se.BondCon1, bond2.BondCon1);
end IdealPressureSource;
```

Hydraulic.Pumps.Ideal.IdealModPressureSource



Modulated ideal pressure source



Information

Model of a modulated ideal pressure source. the flow at the left is equal in both sides.

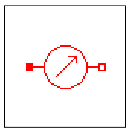
Connectors

Type	Name	Description
input RealInput	u	
Oil		
Properties		
Port_A	port_A	
Port_B	port_B	Hydraulic Connector B

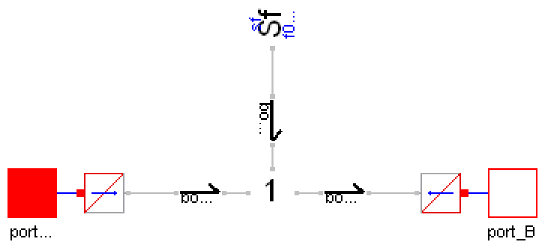
Modelica definition

```
model IdealModPressureSource "Modulated ideal pressure source"
  extends Hydraulic.Interfaces.TwoPortComponent;

  Hydraulic.Interfaces.Hy2BG hy4BG;
  Hydraulic.Interfaces.Hy2BG hy4BG1;
  BondLib.Bonds.Bond bond;
  BondLib.Bonds.Bond bond1;
  BondLib.Junctions.J1p3 j1p3_1;
  BondLib.Bonds.Bond bond2;
  BondLib.Sources.mSe se;
  Modelica.Blocks.Interfaces.RealInput u;
equation
  connect(bond.BondCon1, hy4BG.bondCon);
  connect(bond1.BondCon2, hy4BG1.bondCon);
  connect(j1p3_1.BondCon2, bond.BondCon2);
  connect(j1p3_1.BondCon1, bond1.BondCon1);
  connect(bond2.BondCon2, j1p3_1.BondCon3);
  connect(se.BondCon1, bond2.BondCon1);
  connect(se.s, u);
  connect(hy4BG.port_A, port_A);
  connect(hy4BG1.port_A, port_B);
end IdealModPressureSource;
```

Hydraulic.Pumps.Ideal.IdealForcedFlow



Ideal forced flow



Information

Model of an ideal forced flow. The flow at the left is equal in both sides.

Parameters

Type	Name	Default	Description
VolumeFlowRate	q0	1	[m3/s]

Connectors

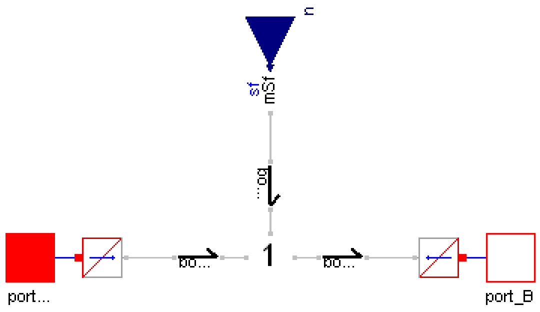
Type	Name	Description
Oil		
Properties		
Port_A	port_A	
Port_B	port_B	Hydraulic Connector B

Modelica definition

```
model IdealForcedFlow "Ideal forced flow"
  extends Hydraulic.Interfaces.TwoPortComponent;

  parameter Modelica.SIunits.VolumeFlowRate q0=1;

  Hydraulic.Interfaces.Hy2BG hy4BG;
  Hydraulic.Interfaces.Hy2BG hy4BG1;

  BondLib.Junctions.J1p3 j1p3_1;
  BondLib.Bonds.Bond bond;
  BondLib.Sources.Sf sf(f0=q0);
  BondLib.Bonds.Bond bond2;
  BondLib.Bonds.Bond bond1;
equation
  connect(hy4BG1.port_A, port_B);
  connect(hy4BG.port_A, port_A);
  connect(bond2.BondCon2, hy4BG1.bondCon);
  connect(j1p3_1.BondCon1, bond2.BondCon1);
  connect(j1p3_1.BondCon3, bond1.BondCon2);
  connect(bond1.BondCon1, sf.BondCon1);
  connect(bond.BondCon2, j1p3_1.BondCon2);
  connect(bond.BondCon1, hy4BG.bondCon);
end IdealForcedFlow;
```

Hydraulic.Pumps.Ideal.IdealModForcedFlow

Modulated ideal forced flow



Information

Model of an ideal modulated forced flow. The flow at the left is equal in both sides.

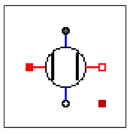
Connectors

Type	Name	Description
input RealInput	u	
Oil		
Properties		
Port_A	port_A	
Port_B	port_B	Hydraulic Connector B

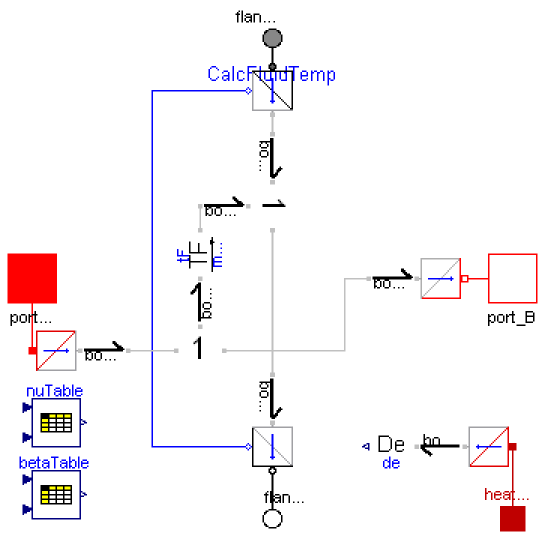
Modelica definition

```
model IdealModForcedFlow "Modulated ideal forced flow"
  extends Hydraulic.Interfaces.TwoPortComponent;

  Hydraulic.Interfaces.Hy2BG hy4BG;
  Hydraulic.Interfaces.Hy2BG hy4BG1;

  BondLib.Junctions.J1p3 j1p3_1;
  BondLib.Bonds.Bond bond;
  BondLib.Sources.mSf sf;
  BondLib.Bonds.Bond bond2;
  BondLib.Bonds.Bond bond1;
  Modelica.Blocks.Interfaces.RealInput u;
equation
  connect(hy4BG1.port_A, port_B);
  connect(hy4BG.port_A, port_A);
  connect(bond2.BondCon2, hy4BG1.bondCon);
  connect(j1p3_1.BondCon1, bond2.BondCon1);
  connect(j1p3_1.BondCon3, bond1.BondCon2);
  connect(bond1.BondCon1, sf.BondCon1);
  connect(bond.BondCon2, j1p3_1.BondCon2);
  connect(bond.BondCon1, hy4BG.bondCon);
  connect(sf.s, u);
end IdealModForcedFlow;
```

Hydraulic.Pumps.Ideal.IdealTurbine



Rotational turbine with no losses



Parameters

Type	Name	Default	Description
Real	trans_factor	1	
Oil			
Properties			
replaceable model VolumeType	CalcFluidTemp	model of oil bulk modulus and kinematic viscosity	
replaceable model Prop	FluidProp	constants of fluid models	

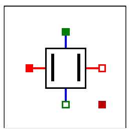
Connectors

Type	Name	Description
Flange_a	flange_a	Rotational Connector
Flange_b	flange_b	Rotational Connector
HeatPort_a	heatPort	
Oil		
Properties		
Port_A	port_A	
Port_B	port_B	Hydraulic Connector B

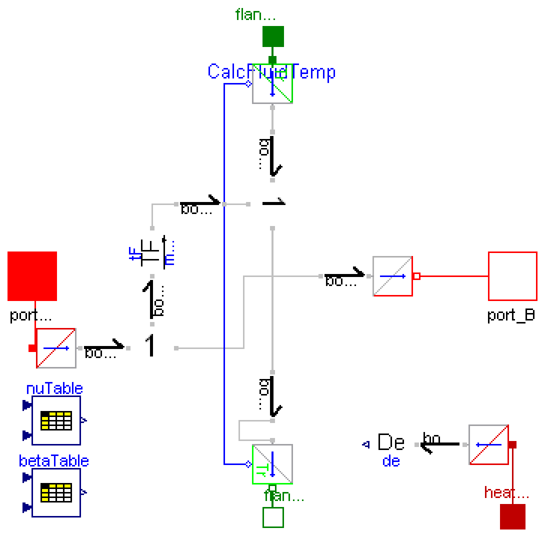
Modelica definition

```
model IdealTurbine "Rotational turbine with no losses"
  extends Hydraulic.Interfaces.RotationSystemTemp;

  parameter Real trans_factor = 1;

  Hydraulic.Interfaces.Hy2BG hy4BG;
  Interfaces.BG2Hy bG2Hy;
  BondLib.Bonds.Bond bond;
  BondLib.Bonds.Bond bond1;
  BondLib.Mechanical.Rotational.Interfaces.Rot2BG rot2BG;
  BondLib.Mechanical.Rotational.Interfaces.BG2Rot bG2Rot;
  BondLib.Junctions.J1p3 j1p3_1;
  BondLib.Passive.TF tF(m=trans_factor);
  BondLib.Bonds.Bond bond2;
  BondLib.Bonds.Bond bond3;
  BondLib.Bonds.Bond bond4;
  BondLib.Junctions.J1p3 j1p3_2;
  BondLib.Bonds.Bond bond5;
  BondLib.Thermal.HeatTransfer.Interfaces.Heat2BG heat2BG;
  BondLib.Bonds.Bond bond6;
  BondLib.Sensors.De de;
equation
  T=de.OutPort1;
  connect(bG2Hy.port_B, port_B);
  connect(bond1.BondCon2, bG2Hy.bondCon);
  connect(bG2Rot.Rot, flange_b);
  connect(rot2BG.Rot, flange_a);
  connect(rot2BG.phi, bG2Rot.phi);
  connect(j1p3_1.BondCon2, bond.BondCon2);
  connect(j1p3_1.BondCon1, bond1.BondCon1);
  connect(bond2.BondCon1, j1p3_1.BondCon3);
  connect(tF.BondCon1, bond2.BondCon2);
  connect(bond3.BondCon1, rot2BG.BondCon1);
  connect(bond4.BondCon2, bG2Rot.BondCon1);
  connect(j1p3_2.BondCon1, bond3.BondCon2);
  connect(j1p3_2.BondCon2, bond4.BondCon1);
  connect(bond5.BondCon1, tF.BondCon2);
  connect(bond5.BondCon2, j1p3_2.BondCon3);
  connect(bond.BondCon1, hy4BG.bondCon);
  connect(hy4BG.port_A, port_A);
  connect(heat2BG.port_a, heatPort);
  connect(bond6.BondCon1,heat2BG. BondCon1);
  connect(de.BondCon1,bond6. BondCon2);
end IdealTurbine;
```

Hydraulic.Pumps.Ideal.IdealTransTurbine



Ideal translational turbine with no losses



Parameters

Type	Name	Default	Description
Real	rot_factor	1	
Oil			
Properties			
replaceable model VolumeType	CalcFluidTemp		model of oil bulk modulus and kinematic viscosity
replaceable model Prop	FluidProp		constants of fluid models

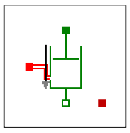
Connectors

Type	Name	Description
Flange_a	flange_a	Translational Connector
Flange_b	flange_b	Translational Connector
HeatPort_a	heatPort	
Oil		
Properties		
Port_A	port_A	
Port_B	port_B	Hydraulic Connector B

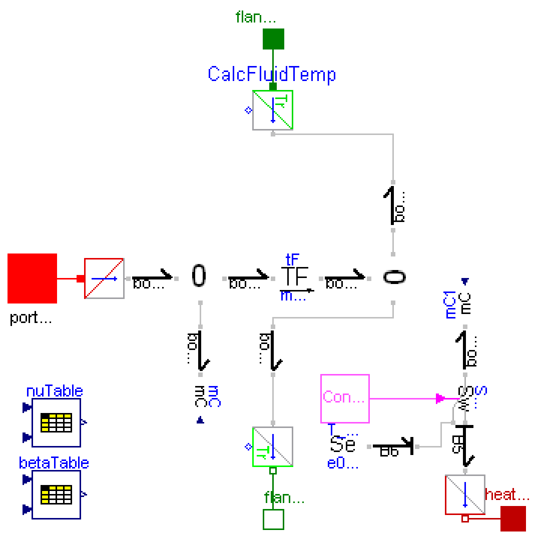
Modelica definition

```
model IdealTransTurbine "Ideal translational turbine with no losses"
  extends Hydraulic.Interfaces.TranslationSystemTemp;
  parameter Real rot_factor = 1;
  BondLib.Mechanical.Translational.Interfaces.Tr2BG tr2BG;
  BondLib.Mechanical.Translational.Interfaces.BG2Tr bG2Tr;
  Interfaces.Hy2BG hy4BG;
  Interfaces.BG2Hy bG2Hy;
  BondLib.Bonds.Bond bond;
  BondLib.Bonds.Bond bond1;
  BondLib.Junctions.J1p3 j1p3_1;
  BondLib.Passive.TF tF(m=rot_factor);
  BondLib.Bonds.Bond bond2;
  BondLib.Bonds.Bond bond3;
  BondLib.Bonds.Bond bond4;
  BondLib.Junctions.J1p3 j1p3_2;
  BondLib.Bonds.Bond bond5;
  BondLib.Thermal.HeatTransfer.Interfaces.Heat2BG heat2BG;
  BondLib.Bonds.Bond bond6;
  BondLib.Sensors.De de;
equation
  T=de.OutPort1;
  connect(tr2BG.Tr, flange_a);
  connect(bG2Tr.Tr, flange_b);
  connect(hy4BG.port_A, port_A);
  connect(bond1.BondCon2,bG2Hy. bondCon);
  connect(j1p3_1.BondCon2,bond. BondCon2);
  connect(j1p3_1.BondCon1,bond1. BondCon1);
  connect(bond2.BondCon1,j1p3_1. BondCon3);
  connect(tF.BondCon1,bond2. BondCon2);
  connect(j1p3_2.BondCon1,bond3. BondCon2);
  connect(j1p3_2.BondCon2,bond4. BondCon1);
  connect(bond5.BondCon1,tF. BondCon2);
  connect(bond5.BondCon2,j1p3_2. BondCon3);
  connect(bond.BondCon1,hy4BG. bondCon);
  connect(bond3.BondCon1, tr2BG.BondCon1);
  connect(bond4.BondCon2, bG2Tr.BondCon1);
  connect(tr2BG.s, bG2Tr.s);
  connect(bG2Hy.port_B, port_B);
  connect(heat2BG.port_a, heatPort);
  connect(bond6.BondCon1, heat2BG.BondCon1);
  connect(de.BondCon1, bond6.BondCon2);
end IdealTransTurbine;
```

Hydraulic.Pumps.Ideal.IdealPiston



ideal one-chamber piston



Parameters

Type	Name	Default	Description
Area	A	1	[m2]
Length	xNom	0.5	Start length of piston [m]
Temperature	TNom	298.15	Temperature if heatPort not connected [K]
Oil			
Properties			
replaceable model VolumeType	CalcFluidTemp	model of oil bulk modulus and kinematic viscosity	
replaceable model Prop	FluidProp	constants of fluid models	

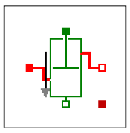
Connectors

Type	Name	Description
HeatPort_a	heatPort	
Flange_a	flange_a	
Flange_b	flange_b	
Oil		
Properties		
Port_A	port_A	

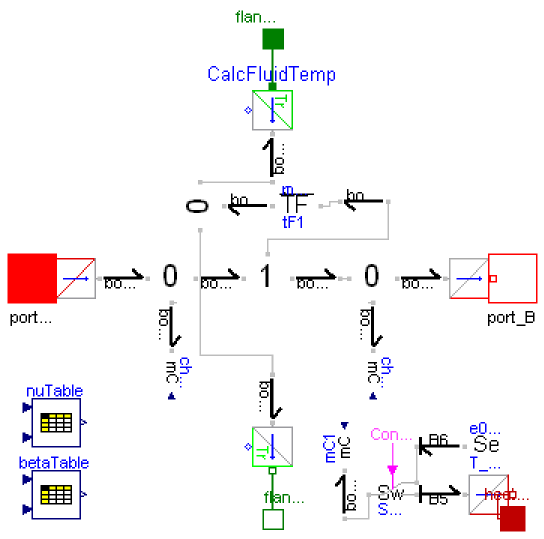
Modelica definition

```
model IdealPiston "ideal one-chamber piston"
  extends Hydraulic.Interfaces.OnePortTemp;
  Modelica.SIunits.Length x(stateSelect=StateSelect.prefer,start=xNom);
  parameter Modelica.SIunits.Area A=1;
  parameter Modelica.SIunits.Length xNom=0.5 "Start length of piston";
  parameter Modelica.SIunits.Temperature TNom=298.15
    "Temperature if heatPort not connected";

  BondLib.Mechanical.Translational.Interfaces.Flange_a flange_a;
  BondLib.Mechanical.Translational.Interfaces.Flange_b flange_b;
  BondLib.Mechanical.Translational.Interfaces.Tr2BG tr2BG;
  BondLib.Mechanical.Translational.Interfaces.BG2Tr bG2Tr;

  Interfaces.Hy2BG hy2BG;
  BondLib.Bonds.Bond bond1;
  BondLib.Junctions.J0p3 j0p3_1;
  BondLib.Bonds.Bond bond2;
  BondLib.Passive.TF tF(m=-1/A);
  BondLib.Junctions.J0p3 j0p3_2;
  BondLib.Bonds.Bond bond;
  BondLib.Bonds.Bond bond3;
  BondLib.Bonds.Bond bond4;
  BondLib.Passive.mC mC;
  BondLib.Bonds.Bond bond5;
protected
  BondLib.Thermal.HeatTransfer.Interfaces.BG2Heat BG2Heat1;
  BondLib.Switches.Sw3 Sw1;
  BondLib.Bonds.eBond B5;
  BondLib.Bonds.fBond B6;
  BondLib.Sources.Se T_nom(e0=TNom);
public
  BondLib.Bonds.Bond bond6;
  BondLib.Passive.mC mC1;
equation
  T=Sw1.p.e;
  x=flange_b.s-flange_a.s;
  mC.s=x*A/beta;
  mC1.s=Prop.cp*x*A*Prop.rho;
  if cardinality(heatPort) <= 1 then
    Sw1.control = false;
  else
    Sw1.control = true;
  end if;

  assert(x>0, "x must not be smaller than zero!");
  connect(tr2BG.Tr, flange_a);
  connect(bond3.BondCon2, tF.BondCon1);
  connect(j0p3_2.BondCon2, bond3.BondCon1);
  connect(bond4.BondCon1, j0p3_2.BondCon3);
  connect(mC.BondCon1, bond4.BondCon2);
  connect(bond5.BondCon1, hy2BG.bondCon);
  connect(bond5.BondCon2, j0p3_2.BondCon1);
  connect(bG2Tr.Tr, flange_b);
  connect(port_A, hy2BG.port_A);
  connect(bond2.BondCon2, bG2Tr.BondCon1);
  connect(bond2.BondCon1, j0p3_1.BondCon2);
  connect(j0p3_1.BondCon1, bond1.BondCon1);
  connect(bond1.BondCon2, tr2BG.BondCon1);
  connect(B5.fBondCon1,Sw1. n2);
  connect(B5.eBondCon1,BG2Heat1. BondCon1);
  connect(Sw1.n1,B6. fBondCon1);
  connect(T_nom.BondCon1,B6. eBondCon1);
  connect(mC1.BondCon1, bond6.BondCon2);
  connect(bond6.BondCon1, Sw1.p);
  connect(BG2Heat1.port_b, heatPort);
  connect(bond.BondCon1, tF.BondCon2);
  connect(bond.BondCon2, j0p3_1.BondCon3);
end IdealPiston;
```

Hydraulic.Pumps.Ideal.IdealTwoChamberPiston



Piston with two chambers



Parameters

Type	Name	Default	Description
Area	A	1	[m2]
Length	L	1	[m]
Length	xNom	0.5	Start length of piston [m]
Pressure	pNom	1e5	Start pressure [Pa]
Temperature	TNom	298.15	Temperature if heatPort not connected [K]
Oil			
Properties			
replaceable model VolumeType	CalcFluidTemp	model of oil bulk modulus and kinematic viscosity	
replaceable model Prop	FluidProp	constants of fluid models	

Connectors

Type	Name	Description
Flange_a	flange_a	Translational Connector
Flange_b	flange_b	Translational Connector
HeatPort_a	heatPort	
Oil		
Properties		
Port_A	port_A	
Port_B	port_B	Hydraulic Connector B

Modelica definition

```
model IdealTwoChamberPiston "Piston with two chambers"
  extends Hydraulic.Interfaces.TranslationSystemTemp;
  Modelica.SIunits.Length x;
  parameter Modelica.SIunits.Area A=1;
  parameter Modelica.SIunits.Length L=1;
  parameter Modelica.SIunits.Length xNom=0.5 "Start length of piston";
  parameter Modelica.SIunits.Pressure pNom=1e5 "Start pressure";
  parameter Modelica.SIunits.Temperature TNom=298.15
    "Temperature if heatPort not connected";
  BondLib.Mechanical.Translational.Interfaces.Tr2BG tr2BG;
  Interfaces.Hy2BG hy2BG;
  BondLib.Bonds.Bond bond1;
  BondLib.Junctions.J0p3 j0p3_2;
  BondLib.Bonds.Bond bond3;
  BondLib.Bonds.Bond bond5;
  BondLib.Bonds.Bond bond4;
  BondLib.Bonds.Bond bond2;
  BondLib.Passive.mC chamb1(e(stateSelect=StateSelect.prefer,start=pNom));
  BondLib.Mechanical.Translational.Interfaces.BG2Tr bG2Tr;
  BondLib.Bonds.Bond bond6;
  BondLib.Junctions.J0p3 j0p3_3;
  BondLib.Bonds.Bond bond7;
  BondLib.Passive.mC chamb2(e(stateSelect=StateSelect.prefer,start=pNom));
  BondLib.Bonds.Bond bond8;
  BondLib.Passive.TF tF1(m=1/A);
protected
  BondLib.Switches.Sw3 Sw1;
  BondLib.Bonds.eBond B5;
public
  BondLib.Bonds.Bond bond10;
  BondLib.Passive.mC mC1;
protected
  BondLib.Sources.Se T_nom(e0=TNom);
  BondLib.Bonds.fBond B6;
protected
  BondLib.Thermal.HeatTransfer.Interfaces.BG2Heat BG2Heat1;
public
  BondLib.Junctions.J1p3 j1p3_1;
  BondLib.Junctions.J0p3 j0p3_1;
  BondLib.Bonds.Bond bond;
  BondLib.Bonds.Bond bond9;
  Interfaces.BG2Hy bG2Hy;
equation
  T=Sw1.p.e;
  x=flange_b.s-flange_a.s;
  chamb1.s=x*A/beta;
  chamb2.s=(L-x)*A/beta;
  mC1.s=Prop.cp*x*A*Prop.rho;
  if cardinality(heatPort) <= 1 then
    Sw1.control = false;
  else
    Sw1.control = true;
  end if;
  assert(x>0, "x must not be smaller than zero!");
  assert(x<L, "x must be smaller than L!");
  connect(bond5.BondCon1,hy2BG. bondCon);
  connect(bond1.BondCon2,tr2BG. BondCon1);
  connect(bG2Tr.Tr, flange_b);
  connect(flange_a, tr2BG.Tr);
  connect(hy2BG.port_A, port_A);
  connect(chamb1.BondCon1, bond4.BondCon2);
  connect(chamb2.BondCon1, bond7.BondCon2);
  connect(bond2.BondCon2, bG2Tr.BondCon1);
  connect(B5.fBondCon1,Sw1. n2);
  connect(Sw1.n1,B6. fBondCon1);
  connect(mC1.BondCon1, bond10.BondCon2);
  connect(bond10.BondCon1, Sw1.p);
  connect(BG2Heat1.port_b, heatPort);
  connect(B5.eBondCon1, BG2Heat1.BondCon1);
  connect(T_nom.BondCon1, B6.eBondCon1);
  connect(j0p3_2.BondCon1, bond5.BondCon2);
  connect(j0p3_2.BondCon3, bond4.BondCon1);
  connect(bond3.BondCon1, j0p3_2.BondCon2);
  connect(j1p3_1.BondCon1, bond3.BondCon2);
  connect(j0p3_3.BondCon3, bond7.BondCon1);
  connect(j0p3_1.BondCon2, bond1.BondCon1);
  connect(j0p3_1.BondCon1, bond2.BondCon1);
  connect(bond.BondCon2, j0p3_1.BondCon3);
  connect(tF1.BondCon2, bond.BondCon1);
  connect(bond9.BondCon2, tF1.BondCon1);
  connect(bond9.BondCon1, j1p3_1.BondCon3);
  connect(bond8.BondCon1, j1p3_1.BondCon2);
  connect(bond8.BondCon2, j0p3_3.BondCon1);
  connect(bond6.BondCon1, j0p3_3.BondCon2);
  connect(bG2Hy.port_B, port_B);
  connect(bG2Hy.bondCon, bond6.BondCon2);
```

```
end IdealTwoChamberPiston;
```

HTML-documentation generated by Dymola Thu Jul 21 09:16:33 2011.

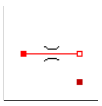
Hydraulic.Components

Hydraulic components

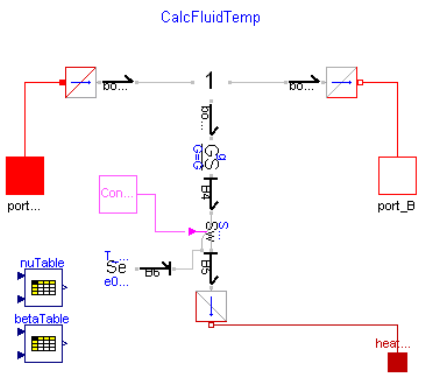
Package Content

Name	Description
LamRes	Model of laminar resistance
TurbRes	Model of turbulent resistance
Line	Hydraulic line
Chamber	Hydaulic chamber
Inertia	Model of Inertance in Line with no compressibility implement non-uniform flow
LineWithKinetics	Model of a line with kinetic enery effects
LongLine	Model of a Long Line with a number of segments

Hydraulic.Components.LamRes



Model of laminar resistance



Information

Laminar resistance with heat port. Via the heat port the produced heat is given out. There is no heat capacity in the model.

Parameters

Type	Name	Default	Description
Conductance	G	1	Conductance [m3/(s.Pa)]
Temperature	Tnom	298.15	Temperature if HeatPort not Connected [K]
Oil			
Properties			
replaceable model VolumeType		CalcFluidTemp	model of oil bulk modulus and kinematic viscosity
replaceable model Prop		FluidProp	constants of fluid models

Connectors

Type	Name	Description
HeatPort_a	heatPort	
Oil		
Properties		
Port_A	port_A	
Port_B	port_B	Hydraulic Connector B

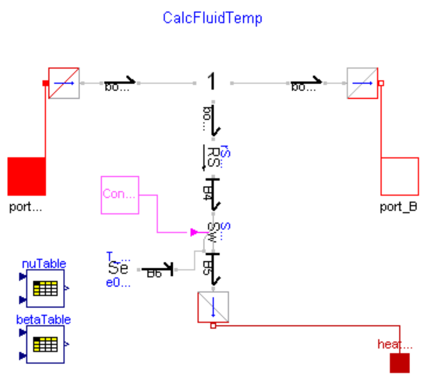
Modelica definition

```
model LamRes "Model of laminar resistance"
  extends Hydraulic.Interfaces.TwoPortComponentTemp;

  parameter Interfaces.HydraulicUnits.Conductance G=1 "Conductance";
  parameter Modelica.SIunits.Temperature Tnom=298.15
    "Temperature if HeatPort not Connected";
```

```
protected
  Hydraulic.Interfaces.Hy2BG hy4BG;

protected
  BondLib.Bonds.Bond bond;
  BondLib.Bonds.Bond bond1;
  BondLib.Junctions.J1p3 j1p3_1;
  BondLib.Bonds.Bond bond2;
  BondLib.Thermal.GS g(
                      G=G);
  Interfaces.BG2Hy bG2Hy;

protected
  BondLib.Bonds.eBond B4;
  BondLib.Thermal.HeatTransfer.Interfaces.BG2Heat BG2Heat1;
  BondLib.Switches.Sw3 Sw1;
  BondLib.Bonds.eBond B5;
  BondLib.Bonds.fBond B6;
  BondLib.Sources.Se T_nom(e0=Tnom);

equation
  T=Sw1.p.e;
  if cardinality(heatPort) <= 1 then
    Sw1.control = false;
  else
    Sw1.control = true;
  end if;

  connect(bond.BondCon1, hy4BG.bondCon);
  connect(j1p3_1.BondCon2, bond.BondCon2);
  connect(j1p3_1.BondCon1, bond1.BondCon1);
  connect(bG2Hy.bondCon, bond1.BondCon2);
  connect(bG2Hy.port_B, port_B);
  connect(Sw1.p,B4. eBondCon1);
  connect(B5.fBondCon1,Sw1. n2);
  connect(B5.eBondCon1,BG2Heat1. BondCon1);
  connect(Sw1.n1,B6. fBondCon1);
  connect(T_nom.BondCon1,B6. eBondCon1);
  connect(hy4BG.port_A, port_A);
  connect(bond2.BondCon1, j1p3_1.BondCon3);
  connect(g.BondCon1, bond2.BondCon2);
  connect(g.BondCon2, B4.fBondCon1);
  connect(BG2Heat1.port_b, heatPort);
end LamRes;
```

Hydraulic.Components.TurbRes

Model of turbulent resistance



Information

Turbulent Resistance with Heatport. There are three possibilities to define the resistance. Via the heat port the produced heat is given out. There is no heat capacity in the model.

Parameters

Type	Name	Default	Description
Length	D	0.01	Diameter [m]
Length	L	0.1	Length of Tube [m]
Boolean	k	true	Use k1 and k2 as parameters
Boolean	zetabool	false	Use zeta as Parameter
Boolean	lambdabool	false	Use lambda
if k=true			

Real	k1	10	
Real	k2	2	
if zeta=true			
Real	zeta	1	
if lambda=true			
Real	lambda	0.005	Resistance Coefficient
Oil			
Properties			
replaceable model VolumeType		CalcFluidTemp	model of oil bulk modulus and kinematic viscosity
replaceable model Prop		FluidProp	constants of fluid models

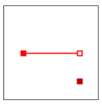
Connectors

Type	Name	Description
HeatPort_a	heatPort	
Oil		
Properties		
Port_A	port_A	
Port_B	port_B	Hydraulic Connector B

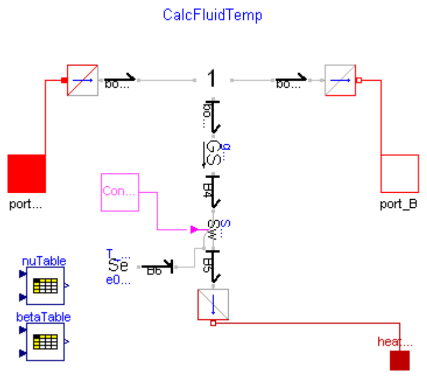
Modelica definition

```
model TurbRes "Model of turbulent resistance"
 extends Hydraulic.Interfaces.TwoPortComponentTemp;
 parameter Modelica.SIunits.Length D=0.01 "Diameter";
 parameter Modelica.SIunits.Length L=0.1 "Length of Tube";
 parameter Boolean k=true "Use k1 and k2 as parameters";
 parameter Boolean zetabool=false "Use zeta as Parameter";

 parameter Boolean lambdabool=false "Use lambda";

 parameter Real k1=10;
 parameter Real k2=2;

 parameter Real zeta=1;

 parameter Real lambda=0.005 "Resistance Coefficient";

protected
 Interfaces.Hy2BG hy2BG;
 Interfaces.BG2Hy bG2Hy;

 BondLib.Bonds.Bond bond;
 BondLib.Bonds.Bond bond1;
 BondLib.Junctions.J1p3 j1p3_1;
 BondLib.Bonds.Bond bond2;
 Hydraulic.Basic.RSquad rS_quad_new(b=if k then 16*k2/(D^4*Modelica.
        Constants.pi) else if zetabool then 8*zeta*Prop.rho/(D^4*Modelica.
        Constants.pi^2) else if lambdabool then 8*lambda*L*Prop.rho/(D^5*
        Modelica.Constants.pi^2) else 1, a=if k then 4*k1/(Modelica.
        Constants.pi*D^2) else if zetabool or lambdabool then 128*Prop.nuConst*Prop.rho*L/(D^4*Modelica.Constants.pi) else 1);

  BondLib.Bonds.eBond B4;
  BondLib.Thermal.HeatTransfer.Interfaces.BG2Heat BG2Heat1;
  BondLib.Switches.Sw3 Sw1;
  BondLib.Bonds.eBond B5;
  BondLib.Bonds.fBond B6;
  BondLib.Sources.Se T_nom(e0=300.15);


equation
  T=Sw1.p.e;
  if cardinality(heatPort) <= 1 then
    Sw1.control = false;
  else
    Sw1.control = true;
  end if;

 connect(bG2Hy.port_B, port_B);
 connect(hy2BG.port_A, port_A);
 connect(j1p3_1.BondCon2,bond. BondCon2);
 connect(j1p3_1.BondCon1,bond1. BondCon1);
 connect(bond2.BondCon1,j1p3_1. BondCon3);
 connect(hy2BG.bondCon, bond.BondCon1);
 connect(bond1.BondCon2, bG2Hy.bondCon);
 connect(bond2.BondCon2, rS_quad_new.BondCon1);
  connect(Sw1.p,B4. eBondCon1);
  connect(B5.fBondCon1,Sw1. n2);
  connect(B5.eBondCon1,BG2Heat1. BondCon1);
  connect(Sw1.n1,B6. fBondCon1);
  connect(T_nom.BondCon1,B6. eBondCon1);
  connect(B4.fBondCon1, rS_quad_new.BondCon2);
  connect(BG2Heat1.port_b, heatPort);
end TurbRes;
```

Hydraulic.Components.Line



Hydraulic line



Information

Model of a Line with laminar or turbulent flow depending on Reynolds Number.

Parameters

Type	Name	Default	Description
Length	D	0.01	Diameter of Pipe [m]
Length	L	1	Length of Pipe [m]
Length	e_r	1e-5	Roughness of Pipe [m]
Oil			
Properties			
replaceable model VolumeType	CalcFluidTemp		model of oil bulk modulus and kinematic viscosity
replaceable model Prop	FluidProp		constants of fluid models

Connectors

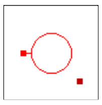
Type	Name	Description
HeatPort_a	heatPort	
Oil		
Properties		
Port_A	port_A	
Port_B	port_B	Hydraulic Connector B

Modelica definition

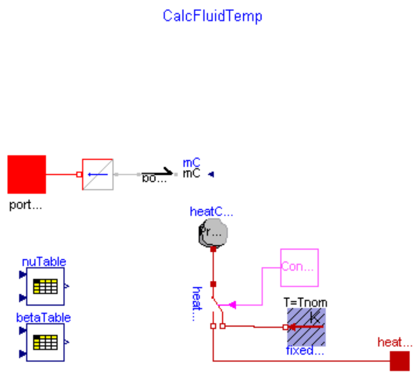
```
model Line "Hydraulic line"

  extends Hydraulic.Interfaces.TwoPortComponentTemp;

  parameter Modelica.SIunits.Length D=0.01 "Diameter of Pipe";
  parameter Modelica.SIunits.Length L=1 "Length of Pipe";
  parameter Modelica.SIunits.Length e_r=1e-5 "Roughness of Pipe";

public
  Hydraulic.Basic.GSswitch g_turb(
    D=D,
    L=L,
    rho=Prop.rho,
    e_r=e_r,
    nu=Prop.nuConst);

protected
  Hydraulic.Interfaces.Hy2BG hy4BG;
  BondLib.Bonds.Bond bond;
  BondLib.Bonds.Bond bond1;
  BondLib.Junctions.J1p3 j1p3_1;
  BondLib.Bonds.eBond bond2;
  Interfaces.BG2Hy bG2Hy;
  BondLib.Bonds.eBond B4;
  BondLib.Thermal.HeatTransfer.Interfaces.BG2Heat BG2Heat1;
  BondLib.Switches.Sw3 Sw1;
  BondLib.Bonds.eBond B5;
  BondLib.Bonds.fBond B6;
  BondLib.Sources.Se T_nom(e0=298.15);
```

```
equation
  T=Sw1.p.e;
  if cardinality(heatPort) <= 1 then
    Sw1.control = false;
  else
    Sw1.control = true;
  end if;
  connect(bond.BondCon1, hy4BG.bondCon);
  connect(j1p3_1.BondCon2, bond.BondCon2);
  connect(j1p3_1.BondCon1, bond1.BondCon1);
  connect(bG2Hy.bondCon, bond1.BondCon2);
  connect(bG2Hy.port_B, port_B);

  connect(Sw1.p,B4. eBondCon1);
  connect(B5.fBondCon1,Sw1. n2);
  connect(B5.eBondCon1,BG2Heat1. BondCon1);
  connect(Sw1.n1,B6. fBondCon1);
  connect(T_nom.BondCon1,B6. eBondCon1);
  connect(B4.fBondCon1, g_turb.BondCon2);
  connect(hy4BG.port_A, port_A);
  connect(BG2Heat1.port_b, heatPort);
  connect(bond2.fBondCon1, j1p3_1.BondCon3);
  connect(bond2.eBondCon1, g_turb.BondCon1);
end Line;
```

Hydraulic.Components.Chamber



Hydaulic chamber



Information

Hydraulic Chamber

Parameters

Type	Name	Default	Description
Volume	V	1	Volume [m3]
Initialization			
Pressure	pNom	1e5	initial Pressure at Port [Pa]
Temperature	Tnom	298.15	Temperature if Heatport not connected [K]
Oil			
Properties			
replaceable model VolumeType		CalcFluidTemp	model of oil bulk modulus and kinematic viscosity
replaceable model Prop		FluidProp	constants of fluid models

Connectors

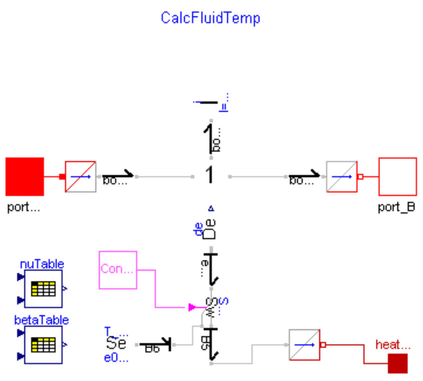
Type	Name	Description
HeatPort_a	heatPort	
Oil		
Properties		
Port_A	port_A	

Modelica definition

```
model Chamber "Hydaulic chamber"
  extends Hydraulic.Interfaces.OnePortTemp(p(start=pNom));
  parameter Modelica.SIunits.Volume V=1 "Volume";
  parameter Modelica.SIunits.Pressure pNom=1e5 "initial Pressure at Port";

  Hydraulic.Interfaces.BG2Hy bG2Hy;

  BondLib.Bonds.Bond bond;
  BondLib.Passive.mC mC;
  BondLib.Thermal.HeatTransfer.Passive.HeatCapacitor heatCapacitor(C=Prop.cp*V*
        Prop.rho);
  BondLib.Thermal.HeatTransfer.Switches.HeatBranch heatBranch;
  BondLib.Thermal.HeatTransfer.Sources.FixedTemperature fixedTemperature(T=Tnom);

  parameter Modelica.SIunits.Temperature Tnom=298.15
    "Temperature if Heatport not connected";
equation
  mC.s=V/beta;
  heatBranch.p.T=T;
  if cardinality(heatPort) <= 1 then
    heatBranch.control = false;
  else
    heatBranch.control = true;
  end if;

  connect(bond.BondCon1, bG2Hy.bondCon);
  connect(bond.BondCon2, mC.BondCon1);
  connect(heatCapacitor.port, heatBranch.p);
  connect(fixedTemperature.port, heatBranch.n1);
  connect(bG2Hy.port_B, port_A);
  connect(heatBranch.n2, heatPort);
end Chamber;
```

Hydraulic.Components.Inertia

Model of Inertance in Line with no compressibility implement non-uniform flow



Information

Moving Mass

Parameters

Type	Name	Default	Description
Area	A	1	[m2]
Length	L	1	[m]
Oil			
Properties			
replaceable model VolumeType	CalcFluidTemp		model of oil bulk modulus and kinematic viscosity
replaceable model Prop	FluidProp		constants of fluid models

Connectors

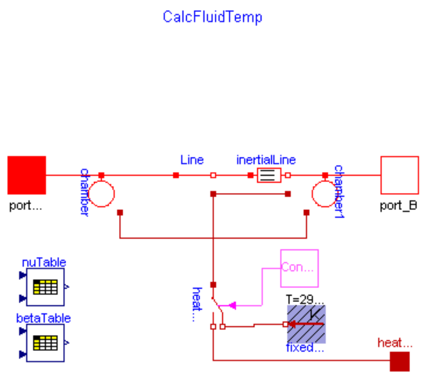
Type	Name	Description
HeatPort_a	heatPort	
Oil		
Properties		
Port_A	port_A	
Port_B	port_B	Hydraulic Connector B

Modelica definition

```
model Inertia
  "Model of Inertance in Line with no compressibility implement non-uniform flow"
  extends Interfaces.TwoPortSystemTemp;

  parameter Modelica.SIunits.Area A=1;
  parameter Modelica.SIunits.Length L=1;

  Hydraulic.Interfaces.Hy2BG hy4BG;

  BondLib.Bonds.Bond bond;
  BondLib.Bonds.Bond bond1;
  BondLib.Junctions.J1p3 j1p3_1;
  BondLib.Bonds.Bond bond2;
  BondLib.Passive.I i(I=L*Prop.rho/A);
  Interfaces.BG2Hy bG2Hy;
protected
  BondLib.Thermal.HeatTransfer.Interfaces.BG2Heat BG2Heat1;
  BondLib.Switches.Sw3 Sw1;
  BondLib.Bonds.eBond B5;
  BondLib.Bonds.fBond B6;
  BondLib.Sources.Se T_nom(e0=298.15);
public
  BondLib.Sensors.De de;
  BondLib.Bonds.eBond eBond;
equation
  T=de.OutPort1;
  if cardinality(heatPort) <= 1 then
    Sw1.control = false;
  else
    Sw1.control = true;
  end if;
  connect(i.BondCon1, bond2.BondCon2);
  connect(hy4BG.bondCon, bond.BondCon1);
  connect(j1p3_1.BondCon2, bond.BondCon2);
  connect(j1p3_1.BondCon1, bond1.BondCon1);
  connect(bond2.BondCon1, j1p3_1.BondCon3);
  connect(bG2Hy.bondCon, bond1.BondCon2);
  connect(bG2Hy.port_B, port_B);
  connect(hy4BG.port_A, port_A);
  connect(B5.fBondCon1,Sw1. n2);
  connect(B5.eBondCon1,BG2Heat1. BondCon1);
  connect(Sw1.n1,B6. fBondCon1);
  connect(T_nom.BondCon1,B6. eBondCon1);
  connect(eBond.fBondCon1, de.BondCon1);
  connect(eBond.eBondCon1, Sw1.p);
  connect(BG2Heat1.port_b, heatPort);
end Inertia;
```

Hydraulic.Components.LineWithKinetics

Model of a line with kinetic enery effects



Information

Line with compressibility, turbulent resistance and moving mass

Parameters

Type	Name	Default	Description
Length	D	1	Diameter of Line [m]
Length	L	1	Length of Line [m]
Length	e_r	1e-5	Roughness of Tube [m]
Initialization			
Pressure	pNom	1e5	Initial Pressure [Pa]
Oil			

Properties		
replaceable model VolumeType	CalcFluidTemp	model of oil bulk modulus and kinematic viscosity
replaceable model Prop	FluidProp	constants of fluid models

Connectors

Type	Name	Description
HeatPort_a	heatPort	
Oil		
Properties		
Port_A	port_A	
Port_B	port_B	Hydraulic Connector B

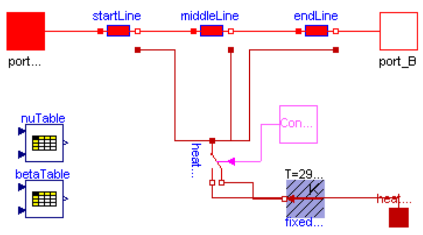
Modelica definition

```
model LineWithKinetics "Model of a line with kinetic enery effects"
  extends Interfaces.TwoPortSystemTemp;

  parameter Modelica.SIunits.Length D = 1 "Diameter of Line";
  parameter Modelica.SIunits.Length L=1 "Length of Line";
  parameter Modelica.SIunits.Length e_r=1e-5 "Roughness of Tube";
  parameter Modelica.SIunits.Pressure pNom=1e5 "Initial Pressure";
  Hydraulic.Components.Inertia inertialLine(
    L=L,
    A=D^2*Modelica.Constants.pi/4,
    redeclare model Prop = Prop,
    redeclare model VolumeType = VolumeType);
  Hydraulic.Components.Line Line(
    D=D,
    L=L,
    redeclare model Prop = Prop,
    redeclare model VolumeType = VolumeType,
    e_r=e_r);

  Chamber chamber(
    V=D^2*Modelica.Constants.pi*L/8,
    redeclare model Prop = Prop,
    redeclare model VolumeType = VolumeType,
    pNom=pNom);
  Chamber chamber1(
    V=D^2*Modelica.Constants.pi*L/8,
    redeclare model Prop = Prop,
    redeclare model VolumeType = VolumeType,
    pNom=pNom);
  BondLib.Thermal.HeatTransfer.Switches.HeatBranch heatBranch;
  BondLib.Thermal.HeatTransfer.Sources.FixedTemperature fixedTemperature;
equation
  T=heatBranch.p.T;
  if cardinality(heatPort) <= 1 then
    heatBranch.control = false;
  else
    heatBranch.control = true;
  end if;

  connect(chamber.heatPort, heatBranch.p);
  connect(Line.heatPort, heatBranch.p);
  connect(chamber1.heatPort, heatBranch.p);
  connect(heatBranch.n2, heatPort);
  connect(fixedTemperature.port, heatBranch.n1);
  connect(port_A, chamber.port_A);
  connect(chamber.port_A, Line.port_A);
  connect(Line.port_B, inertialLine.port_A);
  connect(inertialLine.port_B, chamber1.port_A);
  connect(chamber1.port_A, port_B);
  connect(inertialLine.heatPort, heatBranch.p);
end LineWithKinetics;
```

Hydraulic.Components.LongLine

Model of a Long Line with a number of segments

CalcFluidTemp



Information

Long Line with inertia, compressibility and turbulent resistance

Parameters

Type	Name	Default	Description
Length	L	1	Length of long Line [m]
Length	D	0.01	Diameter [m]
Integer	n	10	Number of lines inside
Length	e_r	1e-5	Roughness of Tube [m]
Initialization			
Pressure	pNom	1e5	Initial Pressure [Pa]
Oil			
Properties			
replaceable model VolumeType		CalcFluidTemp	model of oil bulk modulus and kinematic viscosity
replaceable model Prop		FluidProp	constants of fluid models

Connectors

Type	Name	Description
HeatPort_a	heatPort	
Oil		
Properties		
Port_A	port_A	
Port_B	port_B	Hydraulic Connector B

Modelica definition

```
model LongLine "Model of a Long Line with a number of segments"
  extends Interfaces.TwoPortSystemTemp;
  parameter Modelica.SIunits.Length L=1 "Length of long Line";
  parameter Modelica.SIunits.Length D=0.01 "Diameter";
  parameter Integer n=10 "Number of lines inside";
  parameter Modelica.SIunits.Length e_r=1e-5 "Roughness of Tube";
  parameter Modelica.SIunits.Pressure pNom=1e5 "Initial Pressure";

  LineWithKinetics[n-2] middleLine(
    each D=D,
    each L=L/n,
    redeclare model Prop = Prop,
    redeclare model VolumeType = VolumeType,
    each e_r=e_r,
    each pNom=pNom);

  LineWithKinetics startLine(
    D=D,
    L=L/n,
    redeclare model Prop = Prop,
    redeclare model VolumeType = VolumeType,
    e_r=e_r,
    pNom=pNom);
  LineWithKinetics endLine(
    D=D,
    L=L/n,
    redeclare model Prop = Prop,
    redeclare model VolumeType = VolumeType,
    e_r=e_r,
    pNom=pNom);
  BondLib.Thermal.HeatTransfer.Switches.HeatBranch heatBranch;
  BondLib.Thermal.HeatTransfer.Sources.FixedTemperature fixedTemperature;

equation
  T=heatBranch.p.T;
  if cardinality(heatPort) <= 1 then
    heatBranch.control = false;
  else
    heatBranch.control = true;
  end if;

    for i in 1:(n - 3) loop
      connect(middleLine[i].port_B, middleLine[i + 1].port_A);
    end for;

    for i in 1:n-2 loop
      connect(middleLine[i].heatPort, heatBranch.p);
    end for;

  connect(fixedTemperature.port,heatBranch. n1);
  connect(heatBranch.n2, heatPort);

  connect(startLine.port_A, port_A);
  connect(startLine.port_B, middleLine[1].port_A);
  connect(middleLine[n-2].port_B, endLine.port_A);
  connect(endLine.port_B, port_B);
  connect(startLine.heatPort, heatBranch.p);
  connect(endLine.heatPort, heatBranch.p);
end LongLine;
```

HTML-documentation generated by Dymola Thu Jul 21 09:16:36 2011.

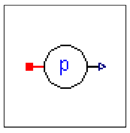
Hydraulic.Sensors

Package sensors

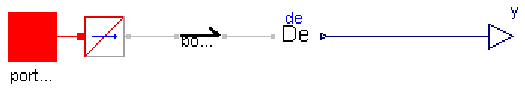
Package Content

Name	Description
PressureSensor	Model of an absolute pressure sensor
FlowSensor	Model of an ideal flow sensor

Hydraulic.Sensors.PressureSensor



Model of an absolute pressure sensor



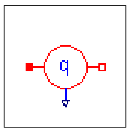
Connectors

Type	Name	Description
output RealOutput	y	
Oil		
Properties		
Port_A	port_A	Hydraulic Connector

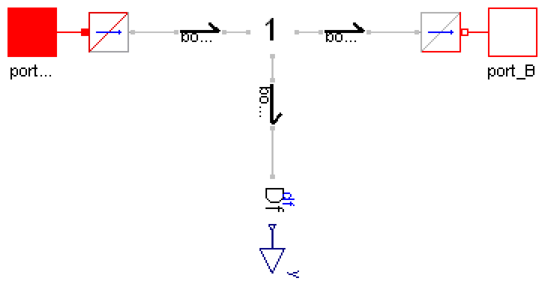
Modelica definition

```
model PressureSensor "Model of an absolute pressure sensor"
  extends Interfaces.OnePort;
  Modelica.Blocks.Interfaces.RealOutput y;
  Interfaces.Hy2BG hy2BG;
  BondLib.Bonds.Bond bond;
  BondLib.Sensors.De de;
equation
  connect(hy2BG.port_A, port_A);
  connect(bond.BondCon1, hy2BG.bondCon);
  connect(de.BondCon1, bond.BondCon2);
  connect(de.OutPort1, y);
end PressureSensor;
```

Hydraulic.Sensors.FlowSensor



Model of an ideal flow sensor



Connectors

Type	Name	Description
output RealOutput	y	
Oil		
Properties		
Port_A	port_A	
Port_B	port_B	Hydraulic Connector B

Modelica definition

```
model FlowSensor "Model of an ideal flow sensor"
  extends Interfaces.TwoPortComponent;
  Interfaces.Hy2BG hy2BG;
  Interfaces.BG2Hy bG2Hy;

  BondLib.Bonds.Bond bond;
  BondLib.Bonds.Bond bond1;
  BondLib.Junctions.J1p3 j1p3_1;
  BondLib.Bonds.Bond bond2;
  BondLib.Sensors.Df df;
  Modelica.Blocks.Interfaces.RealOutput y;
equation
  connect(bG2Hy.port_B, port_B);
  connect(hy2BG.port_A, port_A);
  connect(bond1.BondCon2, bG2Hy.bondCon);
  connect(bond.BondCon1, hy2BG.bondCon);
  connect(j1p3_1.BondCon2, bond1.BondCon1);
  connect(j1p3_1.BondCon1, bond.BondCon2);
  connect(bond2.BondCon1, j1p3_1.BondCon3);
  connect(df.BondCon1, bond2.BondCon2);
  connect(df.OutPort1, y);
end FlowSensor;
```

HTML-documentation generated by Dymola Thu Jul 21 09:16:37 2011.

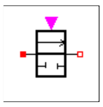
Hydraulic.Controllers

Hydraulic Controllers

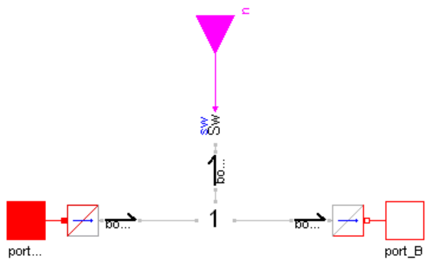
Package Content

Name	Description
SimpleValve	Simple on-off valve
ThreeTwoValve	Model of 3-2-valve with chambers
ModValve	Modulated Valve
CheckValve	Check valve with turbulent resistance
ReliefValve	Relief valve with laminar resistance
Basic	

Hydraulic.Controllers.SimpleValve



Simple on-off valve



Information

Valve which can be fully open or completely shut. If u=true then valve is open.

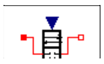
Connectors

Type	Name	Description
input BooleanInput	u	input switch
Oil		
Properties		
Port_A	port_A	
Port_B	port_B	Hydraulic Connector B

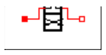
Modelica definition

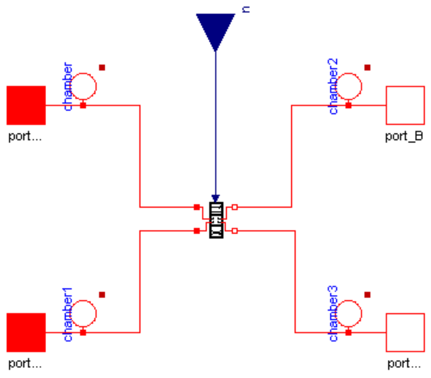
```
model SimpleValve "Simple on-off valve"
  extends Hydraulic.Interfaces.TwoPortComponent;


  Interfaces.Hy2BG hy2BG;
  Interfaces.BG2Hy bG2Hy;
  Modelica.Blocks.Interfaces.BooleanInput u "input switch";
  BondLib.Bonds.Bond bond;
  BondLib.Bonds.Bond bond1;
  BondLib.Switches.Sw sw;
  BondLib.Junctions.J1p3 j1p3_1;
  BondLib.Bonds.Bond bond2;
equation
  connect(bG2Hy.port_B, port_B);
  connect(bond1.BondCon2, bG2Hy.bondCon);
  connect(bond.BondCon1, hy2BG.bondCon);
  connect(sw.BooleanInPort1, u);
  connect(j1p3_1.BondCon2, bond.BondCon2);
  connect(j1p3_1.BondCon1, bond1.BondCon1);
  connect(bond2.BondCon2, sw.BondCon1);
  connect(bond2.BondCon1, j1p3_1.BondCon3);
  connect(hy2BG.port_A, port_A);
end SimpleValve;
```

Hydraulic.Controllers.ThreeTwoValve



Model of 3-2-valve with chambers





Information

Model of a 3-2 way valve with hydraulic Chambers. There are three states: opened, closed and crossed. If input is -1 then valve is opened, if 0 then closeda and if 1 then crossed

Parameters

Type	Name	Default	Description
Oil			
Geometric Data			
Volume	VA	1e-6	Hydraulic Volume at port_A [m3]
Volume	VA1	1e-6	Hydraulic Volume at port_A1 [m3]
Volume	VB	1e-6	Hydraulic Volume at port_B [m3]
Volume	VB1	1e-6	Hydraulic Volume at port_B [m3]

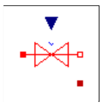
Connectors

Type	Name	Description
input RealInput	u	
Oil		
Properties		
Port_A	port_A	
Port_A	port_A1	
Port_B	port_B	
Port_B	port_B1	

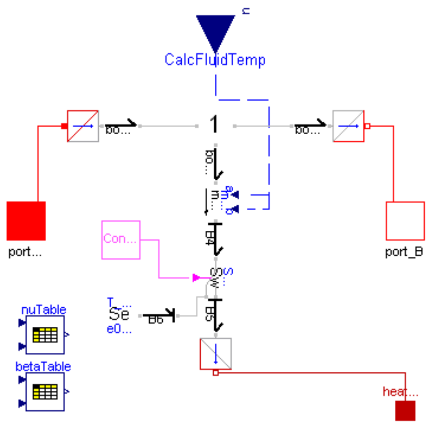
Modelica definition

```
model ThreeTwoValve "Model of 3-2-valve with chambers"

  parameter Modelica.SIunits.Volume VA=1e-6 "Hydraulic Volume at port_A";
  parameter Modelica.SIunits.Volume VA1=1e-6 "Hydraulic Volume at port_A1";
  parameter Modelica.SIunits.Volume VB=1e-6 "Hydraulic Volume at port_B";
  parameter Modelica.SIunits.Volume VB1=1e-6 "Hydraulic Volume at port_B";

  Interfaces.Port_A port_A;
  Interfaces.Port_A port_A1;
  Interfaces.Port_B port_B;
  Interfaces.Port_B port_B1;

  Modelica.Blocks.Interfaces.RealInput u;
  Basic.ThreeTwoValveNoStates threeTwoValveNoStates;
  Components.Chamber chamber(V=VA, redeclare model VolumeType =
        Hydraulic.Interfaces.FluidProps.CalcFluidConst);
  Components.Chamber chamber1(V=VA1, redeclare model VolumeType =
        Hydraulic.Interfaces.FluidProps.CalcFluidConst);
  Components.Chamber chamber2(V=VB, redeclare model VolumeType =
        Hydraulic.Interfaces.FluidProps.CalcFluidConst);
  Components.Chamber chamber3(V=VA, redeclare model VolumeType =
        Hydraulic.Interfaces.FluidProps.CalcFluidConst);
equation
  connect(threeTwoValveNoStates.u, u);
  connect(chamber.port_A, port_A);
  connect(chamber.port_A, threeTwoValveNoStates.port_A);
  connect(port_A1, chamber1.port_A);
  connect(chamber1.port_A, threeTwoValveNoStates.port_A1);
  connect(chamber2.port_A, threeTwoValveNoStates.port_B);
  connect(chamber2.port_A, port_B);
  connect(chamber3.port_A, port_B1);
  connect(chamber3.port_A, threeTwoValveNoStates.port_B1);
end ThreeTwoValve;
```

Hydraulic.Controllers.ModValve



Modulated Valve



Information

Modulated Drossel

Parameters

Type	Name	Default	Description
Length	L	0.1	Length of Resistance, used for laminar part [m]
Length	D0	0.01	Diameter of valve if fully opened [m]
Length	Dmin	1e-4	Minimal diameter of valve [m]
Initialization			
Temperature	TNom	298.15	Temperature if not connected [K]
Oil			
Properties			
replaceable model VolumeType	CalcFluidTemp	model of oil bulk modulus and kinematic viscosity	
replaceable model Prop	FluidProp	constants of fluid models	

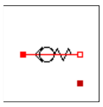
Connectors

Type	Name	Description
HeatPort_a	heatPort	
input RealInput	u	
Oil		
Properties		
Port_A	port_A	
Port_B	port_B	Hydraulic Connector B

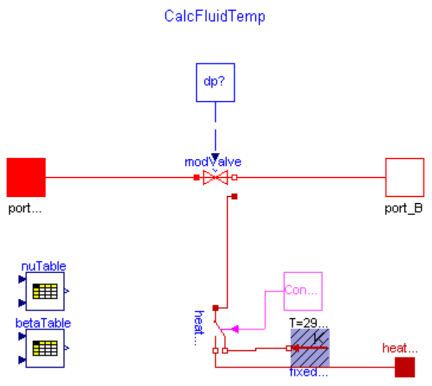
Modelica definition

```
model ModValve "Modulated Valve"

  extends Hydraulic.Interfaces.TwoPortComponentTemp;
  Real zeta;
  Modelica.SIunits.Length D;
  Boolean closed(start=true);

  parameter Modelica.SIunits.Length L=0.1
    "Length of Resistance, used for laminar part";
  parameter Modelica.SIunits.Length D0=0.01 "Diameter of valve if fully opened";
  parameter Modelica.SIunits.Length Dmin=1e-4 "Minimal diameter of valve";
  parameter Modelica.SIunits.Temperature TNom=298.15
    "Temperature if not connected";

  Interfaces.Hy2BG hy2BG;
  Interfaces.BG2Hy bG2Hy;

  BondLib.Bonds.Bond bond;
  BondLib.Bonds.Bond bond1;
  BondLib.Junctions.J1p3 j1p3_1;
  BondLib.Bonds.Bond bond2;
  Modelica.Blocks.Interfaces.RealInput u;
  Hydraulic.Basic.mRSquad mRSquad;
protected
  BondLib.Bonds.eBond B4;
  BondLib.Thermal.HeatTransfer.Interfaces.BG2Heat BG2Heat1;
  BondLib.Switches.Sw3 Sw1;
  BondLib.Bonds.eBond B5;
  BondLib.Bonds.fBond B6;
  BondLib.Sources.Se T_nom(e0=TNom);
equation
  mRSquad.a=128*nu*Prop.rho*L/(D^4*Modelica.Constants.pi);
  mRSquad.b=8*zeta*Prop.rho/(D^4*Modelica.Constants.pi^2);
  zeta=(1-D^2/D0^2)^2;
  D=if closed then Dmin else u*D0;
  closed = (u<=Dmin/D0);
  T=Sw1.p.e;
  if cardinality(heatPort) <= 1 then
    Sw1.control = false;
  else
    Sw1.control = true;
  end if;
  connect(bG2Hy.port_B, port_B);
  connect(hy2BG.port_A, port_A);
  connect(hy2BG.bondCon, bond.BondCon1);
  connect(bond.BondCon2, j1p3_1.BondCon2);
  connect(j1p3_1.BondCon1, bond1.BondCon1);
  connect(bond1.BondCon2, bG2Hy.bondCon);
  connect(bond2.BondCon1, j1p3_1.BondCon3);
  connect(Sw1.p,B4. eBondCon1);
  connect(B5.fBondCon1,Sw1. n2);
  connect(B5.eBondCon1,BG2Heat1. BondCon1);
  connect(Sw1.n1,B6. fBondCon1);
  connect(T_nom.BondCon1,B6. eBondCon1);
  connect(BG2Heat1.port_b, heatPort);
  connect(B4.fBondCon1, mRSquad.BondCon2);
  connect(bond2.BondCon2, mRSquad.BondCon1);
end ModValve;
```

Hydraulic.Controllers.CheckValve



Check valve with turbulent resistance



Information

Model of a Check Valve with turbulent resistances. If Pressuredrop is smaller than pclosed, then the diameter is Dmin if Pressuredrop is bigger than popen, then diameter is Dmax, else in between.

Parameters

Type	Name	Default	Description
Length	Dmin		Diameter if closed (Leakage) [m]
Length	Dmax		Diameter if fully opened [m]
Pressure	pclosed	1e5	[Pa]
Pressure	popen	1.5e5	[Pa]
Temperature	TNom	298.15	Temperature if not connected [K]
Oil			
Properties			

replaceable model VolumeType	CalcFluidTemp	model of oil bulk modulus and kinematic viscosity
replaceable model Prop	FluidProp	constants of fluid models

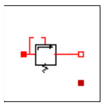
Connectors

Type	Name	Description
HeatPort_a	heatPort	
Oil		
Properties		
Port_A	port_A	
Port_B	port_B	Hydraulic Connector B

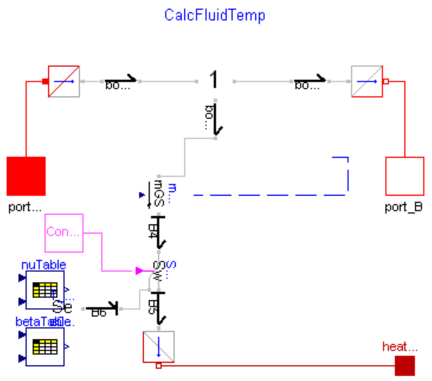
Modelica definition

```
model CheckValve "Check valve with turbulent resistance"

 extends Hydraulic.Interfaces.TwoPortComponentTemp;
 parameter Modelica.SIunits.Length Dmin "Diameter if closed (Leakage)";
 parameter Modelica.SIunits.Length Dmax "Diameter if fully opened";
 parameter Modelica.SIunits.Pressure pclosed=1e5;
 parameter Modelica.SIunits.Pressure popen=1.5e5;

 parameter Modelica.SIunits.Temperature TNom=298.15
    "Temperature if not connected";


  ModValve modValve(
    D0=Dmax,
    Dmin=Dmin,
    TNom=TNom);
  BondLib.Thermal.HeatTransfer.Switches.HeatBranch heatBranch;
  BondLib.Thermal.HeatTransfer.Sources.FixedTemperature fixedTemperature;
equation
  modValve.u = if dp>popen then 1 else if dp>pclosed then (dp-pclosed)/(popen-pclosed) else 0;
  assert(popen>pclosed, "popen needs to be bigger than pclosed");

  heatBranch.p.T=T;
  if cardinality(heatPort) <= 1 then
    heatBranch.control = false;
  else
    heatBranch.control = true;
  end if;

  connect(modValve.port_B, port_B);
  connect(modValve.port_A, port_A);
  connect(modValve.heatPort, heatBranch.p);
  connect(heatBranch.n1, fixedTemperature.port);
  connect(heatPort, heatBranch.n2);
end CheckValve;
```

Hydraulic.Controllers.ReliefValve



Relief valve with laminar resistance



Information

Model of a Relief Valve with laminar Resistances.

Parameters

Type	Name	Default	Description
Conductance	GLeak	1e-10	Conductance of Leakage if closed [m3/(s.Pa)]
Conductance	GThrou	1e-4	Conductance if opened [m3/(s.Pa)]
Pressure	pclosed	1e5	Opening pressure [Pa]
Pressure	popen	2e5	Fully opened [Pa]
Temperature	TNom	298.15	Temperature if not connected [K]
Oil			
Properties			
replaceable model VolumeType	CalcFluidTemp		model of oil bulk modulus and kinematic viscosity
replaceable model Prop	FluidProp		constants of fluid models

Connectors

Type	Name	Description
HeatPort_a	heatPort	
Oil		
Properties		
Port_A	port_A	
Port_B	port_B	Hydraulic Connector B

Modelica definition

```
model ReliefValve "Relief valve with laminar resistance"
  extends Hydraulic.Interfaces.TwoPortComponentTemp;

  parameter Interfaces.HydraulicUnits.Conductance GLeak=1e-10
    "Conductance of Leakage if closed";
  parameter Interfaces.HydraulicUnits.Conductance GThrou=1e-4
    "Conductance if opened";
  parameter Modelica.SIunits.Pressure pclosed=1e5 "Opening pressure";
  parameter Modelica.SIunits.Pressure popen=2e5 "Fully opened ";
  parameter Modelica.SIunits.Temperature TNom=298.15
    "Temperature if not connected";
protected
 Interfaces.Hy2BG hy2BG;
 Interfaces.BG2Hy bG2Hy;
 BondLib.Bonds.Bond bond;
 BondLib.Bonds.Bond bond1;
 BondLib.Junctions.J1p3 j1p3_1;
 BondLib.Bonds.Bond bond2;
 BondLib.Thermal.mGS mGS;
  BondLib.Bonds.eBond B4;
  BondLib.Thermal.HeatTransfer.Interfaces.BG2Heat BG2Heat1;
  BondLib.Switches.Sw3 Sw1;
  BondLib.Bonds.eBond B5;
  BondLib.Bonds.fBond B6;
  BondLib.Sources.Se T_nom(e0=TNom);
equation

  mGS.s = if dp < pclosed then GLeak else if dp > popen then GThrou else GLeak +(GThrou-GLeak)*(dp-pclosed)/(popen-pclosed);
  T=Sw1.p.e;
  if cardinality(heatPort) <= 1 then
    Sw1.control = false;
  else
    Sw1.control = true;
  end if;

 connect(j1p3_1.BondCon2,bond. BondCon2);
 connect(j1p3_1.BondCon1,bond1. BondCon1);
 connect(bond2.BondCon1,j1p3_1. BondCon3);
 connect(hy2BG.bondCon,bond. BondCon1);
 connect(bond1.BondCon2,bG2Hy. bondCon);
  connect(Sw1.p,B4. eBondCon1);
  connect(B5.fBondCon1,Sw1. n2);
  connect(B5.eBondCon1,BG2Heat1. BondCon1);
  connect(Sw1.n1,B6. fBondCon1);
  connect(T_nom.BondCon1,B6. eBondCon1);
  connect(B4.fBondCon1, mGS.BondCon2);
  connect(bG2Hy.port_B, port_B);
  connect(hy2BG.port_A, port_A);
  connect(BG2Heat1.port_b, heatPort);
  connect(bond2.BondCon2, mGS.BondCon1);
end ReliefValve;
```

HTML-documentation generated by Dymola Thu Jul 21 09:16:39 2011.

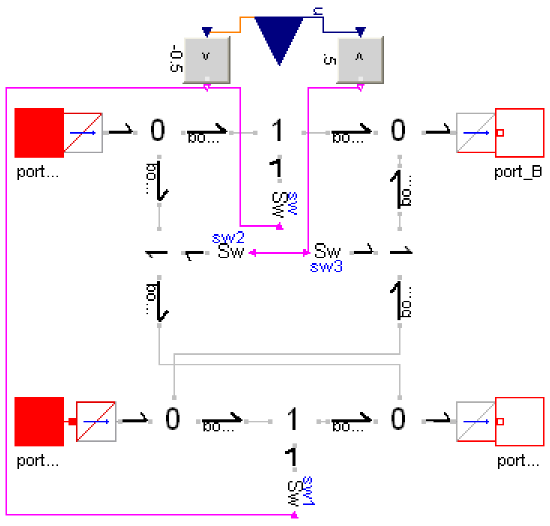
Hydraulic.Controllers.Basic

Package Content

Name	Description
ThreeTwoValveNoStates	Valve with three states

Hydraulic.Controllers.Basic.ThreeTwoValveNoStates

Valve with three states



Information

Model of a 3-2-Valve: If u=-1 Port_A and Port_B resp. Port_A1 and Port_B1 are connected. If u=0 everything is shut. If u=1 Port_A and Port_B1 resp. Port_A1 and Port_B are connected.

Connectors

Type	Name	Description
input RealInput	u	Control input
Oil		
Properties		
Port_A	port_A	Hydraulic Connector
Port_A	port_A1	Hydraulic Connector
Port_B	port_B	Hydraulic Connector
Port_B	port_B1	Hydraulic Connector

Modelica definition

```
model ThreeTwoValveNoStates "Valve with three states"
  Interfaces.Port_A port_A "Hydraulic Connector";
  Interfaces.Port_A port_A1 "Hydraulic Connector";
  Interfaces.Port_B port_B "Hydraulic Connector";
  Interfaces.Port_B port_B1 "Hydraulic Connector";
  Modelica.Blocks.Interfaces.RealInput u "Control input";
  BondLib.Switches.Sw sw;
  Interfaces.Hy2BG hy2BG;
  Interfaces.Hy2BG hy2BG1;
  Interfaces.BG2Hy bG2Hy;
  Interfaces.BG2Hy bG2Hy1;
  BondLib.Bonds.Bond bond;
  BondLib.Bonds.Bond bond1;
  BondLib.Junctions.J1p3 j1p3_1;
  BondLib.Bonds.Bond bond2;
  BondLib.Junctions.J0p3 j0p3_1;
  BondLib.Junctions.J0p3 j0p3_2;
  BondLib.Bonds.Bond bond3;
  BondLib.Bonds.Bond bond4;
  BondLib.Switches.Sw sw1;
  BondLib.Bonds.Bond bond5;
  BondLib.Bonds.Bond bond6;
  BondLib.Junctions.J1p3 j1p3_2;
  BondLib.Bonds.Bond bond7;
  BondLib.Junctions.J0p3 j0p3_5;
  BondLib.Junctions.J0p3 j0p3_6;
  BondLib.Bonds.Bond bond8;
  BondLib.Bonds.Bond bond9;
  BondLib.Junctions.J1p3 j1p3_3;
  BondLib.Bonds.Bond bond10;
  BondLib.Bonds.Bond bond11;
  BondLib.Switches.Sw sw2;
  BondLib.Bonds.Bond bond12;
  BondLib.Bonds.Bond bond13;
  BondLib.Junctions.J1p3 j1p3_4;
  BondLib.Bonds.Bond bond14;
  BondLib.Switches.Sw sw3;
  BondLib.Bonds.Bond bond15;
  Modelica.Blocks.Logical.GreaterThreshold lessThreshold(threshold=-0.5);
  Modelica.Blocks.Logical.LessThreshold greaterThreshold(threshold=.5);
equation
  connect(bG2Hy1.port_B, port_B1);
  connect(bG2Hy.port_B, port_B);
  connect(hy2BG.port_A, port_A);
  connect(hy2BG1.port_A, port_A1);
  connect(bond.BondCon2, j1p3_1.BondCon1);
  connect(j1p3_1.BondCon2, bond1.BondCon1);
  connect(bond2.BondCon1, sw.BondCon1);
  connect(bond2.BondCon2, j1p3_1.BondCon3);
  connect(bond3.BondCon1, hy2BG.bondCon);
  connect(bond3.BondCon2, j0p3_1.BondCon1);
  connect(j0p3_2.BondCon1, bond1.BondCon2);
  connect(bG2Hy.bondCon, bond4.BondCon2);
  connect(bond4.BondCon1, j0p3_2.BondCon2);
  connect(bond5.BondCon2, j1p3_2.BondCon1);
  connect(j1p3_2.BondCon2, bond6.BondCon1);
  connect(bond7.BondCon1, sw1.BondCon1);
  connect(bond7.BondCon2, j1p3_2.BondCon3);
  connect(bond8.BondCon2, j0p3_5.BondCon1);
  connect(j0p3_6.BondCon1, bond6.BondCon2);
  connect(bond9.BondCon1, j0p3_6.BondCon2);
  connect(bond10.BondCon1, j0p3_1.BondCon3);
  connect(bond10.BondCon2, j1p3_3.BondCon2);
  connect(bond11.BondCon1, j1p3_3.BondCon1);
  connect(bond11.BondCon2, j0p3_6.BondCon3);
  connect(bond12.BondCon2, j1p3_3.BondCon3);
  connect(bond12.BondCon1, sw2.BondCon1);
  connect(bond13.BondCon1, j0p3_5.BondCon3);
  connect(j1p3_4.BondCon2, bond13.BondCon2);
  connect(bond14.BondCon2, j1p3_4.BondCon3);
  connect(sw3.BondCon1, bond14.BondCon1);
  connect(bond15.BondCon1, j1p3_4.BondCon1);
  connect(bond15.BondCon2, j0p3_2.BondCon3);
  connect(u, lessThreshold.u);
  connect(lessThreshold.y, sw.BooleanInPort1);
  connect(lessThreshold.y, sw1.BooleanInPort1);
  connect(bG2Hy1.bondCon, bond9.BondCon2);
  connect(bond.BondCon1, j0p3_1.BondCon2);
  connect(bond8.BondCon1, hy2BG1.bondCon);
  connect(bond5.BondCon1, j0p3_5.BondCon2);
  connect(greaterThreshold.u, u);
  connect(greaterThreshold.y, sw3.BooleanInPort1);
  connect(sw2.BooleanInPort1, greaterThreshold.y);
end ThreeTwoValveNoStates;
```

HTML-documentation generated by Dymola Thu Jul 21 09:16:42 2011.

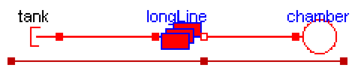
Hydraulic.Examples

Examples for the Hydraulic Library

Package Content

Name	Description
LongLineRelief	
TurbineControl	
HydraulicGear	

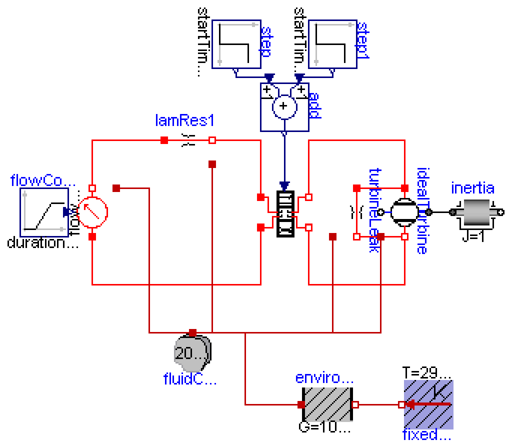
Hydraulic.Examples.LongLineRelief



Modelica definition

```
model LongLineRelief
  Components.Chamber chamber(      pNom=1e5,
    V=1,
    redeclare model VolumeType =
        Hydraulic.Interfaces.FluidProps.CalcFluidPress);
  Pumps.Tank tank(        G=2e-8, p0=1e7,
    redeclare model VolumeType =
        Hydraulic.Interfaces.FluidProps.CalcFluidConst);
  Components.LongLine longLine(
    D=30e-3,
    e_r=1e-6,
    L=100,
    n=10,
    redeclare model VolumeType =
        Hydraulic.Interfaces.FluidProps.CalcFluidPress);
equation
  connect(longLine.port_A, tank.port_A);
  connect(longLine.port_B, chamber.port_A);
  connect(tank.heatPort, longLine.heatPort);
  connect(longLine.heatPort, chamber.heatPort);
end LongLineRelief;
```

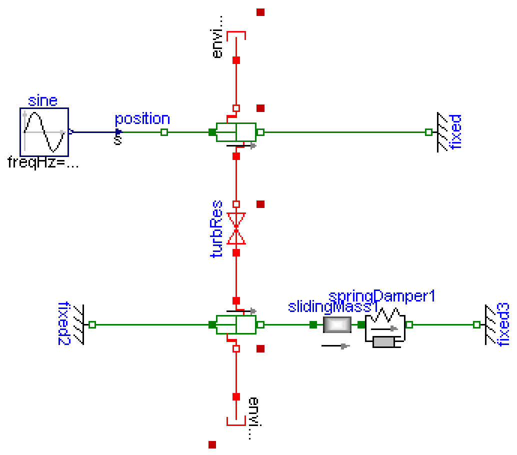
Hydraulic.Examples.TurbineControl



Modelica definition

```
model TurbineControl
  Controllers.Basic.ThreeTwoValveNoStates threeTwoValveNoStates;
  Pumps.ModForcedFlow flowSource(redeclare model VolumeType =
        Hydraulic.Interfaces.FluidProps.CalcFluidTemp, G=1e-6);
  Modelica.Blocks.Sources.Ramp flowControl(
    startTime=1,
    duration=10,
    height=1e-1);
  Pumps.Ideal.IdealTurbine idealTurbine(trans_factor=1e6);
  BondLib.Mechanical.Rotational.Passive.Inertia inertia;
  Components.LamRes turbineLeak(G=1e-8);
  BondLib.Thermal.HeatTransfer.Passive.HeatCapacitor fluidCapacity(C=2028*
        100);
  BondLib.Thermal.HeatTransfer.Passive.ThermalConductor environment(G=1000);
  BondLib.Thermal.HeatTransfer.Sources.FixedTemperature fixedTemperature;
  Components.LamRes lamRes1(G=1e-8);
  Modelica.Blocks.Sources.Step step(
    height=-1,
    offset=1,
    startTime=20);
  Modelica.Blocks.Sources.Step step1(height=-1, startTime=50);
  Modelica.Blocks.Math.Add add;
equation
  connect(threeTwoValveNoStates.port_A1, flowSource.port_A);
  connect(flowControl.y, flowSource.u);
  connect(idealTurbine.port_A, threeTwoValveNoStates.port_B);
  connect(threeTwoValveNoStates.port_B1, idealTurbine.port_B);
  connect(inertia.flange_a, idealTurbine.flange_a);
  connect(fluidCapacity.port, flowSource.heatPort);
  connect(fluidCapacity.port, idealTurbine.heatPort);
  connect(turbineLeak.heatPort, fluidCapacity.port);
  connect(environment.port_a, fluidCapacity.port);
  connect(fixedTemperature.port, environment.port_b);
  connect(lamRes1.port_B, threeTwoValveNoStates.port_A);
  connect(lamRes1.port_A, flowSource.port_B);
  connect(lamRes1.heatPort, fluidCapacity.port);
  connect(add.y, threeTwoValveNoStates.u);
  connect(add.u2, step.y);
  connect(step1.y, add.u1);
  connect(turbineLeak.port_A, idealTurbine.port_A);
  connect(turbineLeak.port_B, idealTurbine.port_B);
end TurbineControl;
```

Hydraulic.Examples.HydraulicGear



Modelica definition

```
model HydraulicGear
  Pumps.Tank envirnonment1(
    redeclare model VolumeType =
        Hydraulic.Interfaces.FluidProps.CalcFluidConst,
    p0=1e5,
    G=1e-3);
  Hydraulic.Pumps.Ideal.IdealTwoChamberPiston idealTwoChamberPistonnew(
    redeclare model VolumeType =
        Hydraulic.Interfaces.FluidProps.CalcFluidConst,
    pNom=1e5,
    A=0.01);
  BondLib.Mechanical.Translational.Passive.Fixed fixed2(s0=0);
  Pumps.Tank environment2(
    redeclare model VolumeType =
        Hydraulic.Interfaces.FluidProps.CalcFluidConst,
    G=1e-3,
    p0=1e5);
  Hydraulic.Pumps.Ideal.IdealTwoChamberPiston idealTwoChamberPistonnew1(
    redeclare model VolumeType =
        Hydraulic.Interfaces.FluidProps.CalcFluidConst,
    pNom=1e5,
    A=0.02);
  BondLib.Mechanical.Translational.Passive.SlidingMass slidingMass1(s0=0.5);
  BondLib.Mechanical.Translational.Passive.SpringDamper springDamper1(
    s_rel0=0.5,
    c=10000,
    d=100);
  BondLib.Mechanical.Translational.Passive.Fixed fixed3(s0=1);
  BondLib.Mechanical.Translational.Sources.Position position;
  BondLib.Mechanical.Translational.Passive.Fixed fixed(s0=1);
  Modelica.Blocks.Sources.Sine sine(
    freqHz=0.2,
    amplitude=0.3,
    offset=0.5);
  Components.TurbRes turbRes(
    redeclare model VolumeType =
        Hydraulic.Interfaces.FluidProps.CalcFluidConst,
    D=0.01,
    L=1);
equation

  connect(idealTwoChamberPistonnew.port_B, envirnonment1.port_A);
  connect(idealTwoChamberPistonnew1.flange_a, fixed2.flange_b);
  connect(idealTwoChamberPistonnew1.port_B, environment2.port_A);
  connect(slidingMass1.flange_b, springDamper1.flange_a);
  connect(fixed3.flange_b, springDamper1.flange_b);
  connect(idealTwoChamberPistonnew.flange_b, fixed.flange_b);
  connect(turbRes.port_B, idealTwoChamberPistonnew.port_A);
  connect(turbRes.port_A, idealTwoChamberPistonnew1.port_A);
  connect(idealTwoChamberPistonnew1.flange_b, slidingMass1.flange_a);
  connect(position.s, sine.y);
  connect(idealTwoChamberPistonnew.flange_a, position.flange_b);
end HydraulicGear;
```

HTML-documentation generated by Dymola Thu Jul 21 09:16:44 2011.