M. Tiller:
Modelica Thermal Library.
Modelica Workshop 2000 Proceedings, pp. 137-144.

Paper presented at the Modelica Workshop 2000, Oct. 23.-24., 2000, Lund, Sweden.

All papers of this workshop can be downloaded from
http://www.Modelica.org/modelica2000/proceedings.html

Modelica Thermal Library

Michael Tiller
Modelica Association

October 13, 2000

Abstract

This paper presents a new library of thermal components to be incorporated into the Modelica Standard Library. The goal for this package is to provide a reasonably simple set of basic models and connectors that can be used as the basis for further development in the thermal domain.

1 Introduction

The Modelica Standard Library is an excellent example of how a common set of `connector` definitions, interfaces and basic models can be used not just as building blocks for complex models but also as a mechanism for collaboration between model developers and users. The key to the Modelica Standard Library's success is the fact common connector definitions are, for the most part, sufficient to allow models developed by different people or organizations to be used together. A great deal has been accomplished using the connector definitions in the current version[1] of the Modelica Standard Library.

However, the current version of the Modelica Standard Library does not provide the basic set of objects needed to allow collaboration in some common engineering domains. In particular, connector definitions for both thermal and hydraulic systems have not yet been incorporated into the Modelica Standard Library. This is not to say these domains have been overlooked. An extensive library of Modelica models is being developed for complex thermo-hydraulic systems[7] and a commercial Modelica library already exists for hydraulic components[3]. Nevertheless, the basic connector definitions required for **simple** thermal and hydraulic systems are not yet part of the Modelica Standard Library.

The focus of the library is a connector definition for thermal systems. Such a connector needs to include temperature as an across variable and power as a through variable. Several names were considered for this library. A similar library, called the `HeatFlow` library, was developed as part of the `ModelicaAdditions` library[2]. The name `HeatFlow` did not convey the scope of this new library very well. Another consideration was `HeatTransfer` which seemed very appropriate. However, the name `HeatTransfer` implies the scope of the library is limited to the three modes of heat transfer: conduction, convection and radiation. This seemed too constraining. The name `Thermal` was decided upon because it captured the essence of the library and left room for many models beyond the scope of simple heat transfer (see Section 6).

2 Interfaces

2.1 Connector definition

The connector definition is the foundation of any library in Modelica. Care needs to be taken when designing a library to make sure that the appropriate level of detail is present in a connector. To little detail will limit the applicability of the connector while too much detail might make the library overly complex for many purposes.

[1] The current version of the Modelica Standard Library is version 1.3.1

For the most part, the connector definition for the `Thermal` library was straightforward to arrive at. As previously mentioned, the across variable at the connector (*i.e.,* the non-flow quantity) is temperature. The only real issue was what to make the through (or `flow`) quantity. The choice was between a scalar representation for heat flow or a vector representation of heat flow. The latter was decided upon because it does not increase the complexity much (the `Modelica.Blocks` library is a precedent for such an approach) and it leaves open the possibility to develop two and three dimensional components.

The one remaining issue was what to call the connector. The convention for the Modelica Standard Library has been to name connectors after their physical counterparts. For example, in the case of the rotational mechanics library the connector is called a flange. Unfortunately, naming a thermal connector is difficult because there is no clear physical counterpart to name it after. Instead, conventions in other computational disciplines were considered. For heat transfer equations in multiple dimensions, the term *node* is often used[5, 6, 4]. Because of this precedent, the name `Node` was settled on. After all this consideration, the following connector definition was adopted:

```
connector Node "Temperature nexus"
  package SIunits=Modelica.SIunits;
  parameter Integer ndim=1 "Spatial dimensions";
  SIunits.Temperature T(start=300);
  flow SIunits.HeatFlowRate q[ndim];
end Node;
```

Note the fact that a `start` value is provided for the temperature. This is important since the units are Kelvin and the default value is normally zero. The `start` attribute was changed because very few practical applications in the thermal domain have temperatures near absolute zero.

The `Node` definition does not include any graphical annotations. However, graphical annotations are useful in diagrams for distinguishing different connectors. For this reason, three additional connectors, with different graphical representations, are derived from `Node`.

2.2 Simple One-Dimensional Models

Because one-dimensional heat transfer is likely to be quite common, the following `partial model` was developed:

```
partial model Element1D
  Node_a a(final ndim=1);
  Node_b b(final ndim=1);
protected
  Modelica.SIunits.HeatFlowRate q "Flow from a->b";
  Modelica.SIunits.Temperature dT "a.T-b.T";
equation
  dT = a.T - b.T;
  a.q[1] = q;
  b.q[1] = -q;
end Element1D;
```

where `dT` is the temperature difference across the component and `q` is a scalar representing heat flow through the component. By using this `model`, the fact that `a.q` and `b.q` are arrays does not complicate the writing simple models.

3 Linear Components

A set of basic one-dimensional, linear models was developed to describe energy storage and each of the three modes of heat transfer.

3.1 Capacitance

The constitutive equation for the thermal capacitance model is:

$$Vc_p\rho\frac{dT}{dt} = q \tag{1}$$

where V is the volume, c_p is the specific heat, ρ is the density, T is the uniform temperature of the mass and q is the heat flow. This relationship can be expressed in Modelica as:

```
model Capacitance
  parameter Modelica.SIunits.SpecificHeatCapacity cp(start=1.0);
  parameter Modelica.SIunits.Density rho(start=1.0);
  parameter Modelica.SIunits.Volume V(start=1.0);

  Modelica.Thermal.Interfaces.Node_c n(final ndim=1);
equation
  V*cp*rho*der(n.T) = n.q[1];
end Capacitance;
```

3.2 Conduction

The constitutive equation for conduction is:

$$q = Ak\frac{dT}{dx} \tag{2}$$

where A is the area and k is the thermal conductivity. Since we do not have a convenient way of expressing spatial derivatives in Modelica, the spatial derivative must be approximated as:

$$\frac{dT}{dx} = \frac{\Delta T}{L} \tag{3}$$

where ΔT is the temperature across the conducting element and L is the length of the conducting element. The Modelica code for this model can be written as follows:

```
model Conduction
  extends Modelica.Thermal.Interfaces.Element1D;
  parameter Modelica.SIunits.ThermalConductivity k(start=1.0);
  parameter Modelica.SIunits.Length L(start=1.0);
  parameter Modelica.SIunits.Area A(start=1.0);
equation
  q = A*k*dT/L;
end Conduction;
```

3.3 Convection

The constitutive equation for convection is:

$$q = Ah\Delta T \tag{4}$$

where h is the coefficient of heat transfer. This model is then expressed in Modelica as:

```
model Convection
  extends Modelica.Thermal.Interfaces.Element1D;
  package SI=Modelica.SIunits;
  parameter SI.CoefficientOfHeatTransfer h(start=1.0);
  parameter SI.Area A(start=1.0);
equation
  q = A*h*dT;
end Convection;
```

3.4 Radiation

Finally, the model for black body radiation is based on the equation:

$$q = F\sigma A(T_a^4 - T_b^4) \tag{5}$$

where F is the view factor and σ is the Stefan-Boltzmann constant. From this equation, we construct the following model:

```
model BlackBodyRadiation
  extends Modelica.Thermal.Interfaces.Element1D;
  parameter Real F "View factor";
  parameter Modelica.SIunits.Area A(start=1.0);
equation
  q = F*Modelica.Constants.sigma*A*(a.T^4 - b.T^4);
end BlackBodyRadiation;
```

4 Boundary Conditions

Once we have the basic mechanisms for describing the flow of energy through a thermal system, we require models to specific boundary conditions. The models in this `package` are general enough to function for any number of spatial dimensions.

4.1 Prescribed Temperature

There are two types of prescribed temperature boundary conditions. One assumes the prescribed temperature is a constant and is represented as:

```
model FixedTemperature
  parameter Modelica.SIunits.Temperature T;
  Interfaces.Node_c n;
equation
  n.T = T;
end FixedTemperature;
```

The other type assumes that a temperature will be provided by a `block` from the `Modelica.Blocks` `package`. For this latter case the boundary condition is represented as:

```
model VariableTemperature
  Modelica.Blocks.Interfaces.InPort T(final n=1);
  Interfaces.Node_c n;
equation
  n.T = T.signal[1];
end VariableTemperature;
```

4.2 Prescribed Heat Flux

The other simple boundary condition provided is a prescribed heat flux which can be represented by the following code:

```
model PrescribedHeatFlux
  parameter Integer ndim=1;
  Modelica.Blocks.Interfaces.InPort q(final n=ndim);
  Modelica.SIunits.Area A(start=1.0);
  Interfaces.Node_c n(final ndim=ndim);
equation
  n.q = A*q.signal;
end PrescribedHeatFlux;
```

5 Sensors

5.1 Temperature

There are really two temperature sensors available in the `Modelica.Thermal` package. The first is an ideal temperature sensor (*i.e.*, it senses temperature changes instantly without any heat transfer). This model can be expressed simply as:

```
model TemperatureSensor
  Modelica.Blocks.Interfaces.OutPort T;
  Modelica.Thermal.Interfaces.Node_c n;
equation
  T.signal[1] = n.T;
end TemperatureSensor;
```

Another temperature sensor that is available is the `Thermocouple` model. Strictly speaking, this model should probably have been discussed in Section 6 because it actually mixes two physical domains (*i.e.*, thermal and electrical). However, since the `Thermocouple` model is primarily a sensor the model is included here.

One way used by instrumentation manufacturers[1] to describe the behavior of a thermocouple is to use the following general equation:

$$E = \sum_{j=0}^{n} C_j T^j \tag{6}$$

where E is the potential across the thermocouple (in the absence of any current flowing), C is the vector of calibration coefficients for the thermocouple and T is the temperature of the thermocouple. This behavior can be represented in Modelica using the following code:

```
model Thermocouple
  extends Modelica.Electrical.Analog.Interfaces.OnePort;
  parameter Real C[:] "Calibration coefficients";
  Modelica.Thermal.Interfaces.Node_c node_c;
equation
  v = polyval(C, node_c.T);
  node_c.q = zeros(node_c.ndim);
end Thermocouple;
```

5.2 Heat Flow

The last basic sensor model included is the `HeatFlow` sensor which is written as follows:

```
model HeatFlow
  extends Modelica.Icons.RotationalSensor;
  Modelica.Thermal.Interfaces.Node_a a;
  Modelica.Thermal.Interfaces.Node_b b;
  Modelica.Blocks.Interfaces.OutPort heat(n=size(1, a.q))
     "Heat flowing from a->b";
equation
  a.q + b.q = 0;
  heat.signal = a.q;
  a.T = b.T;
end HeatFlow;
```

6 Mixed Domain

As mentioned in Section 1, one of the reasons the name `Modelica.Thermal` was chosen was that the scope of the library goes beyond simple heat transfer. In particular, many models span more than one physical domain.

One example of a mixed domain model is the `HeaterElement` model which extends the `Resistor` model provided in `Modelica.Electrical` package. The `HeaterElement` model functions as a resistor but also contributes the energy dissipated by the resistor to a thermal node as shown in the following model:

```
model HeaterElement
  extends Modelica.Electrical.Analog.Basic.Resistor;
  parameter Real efficiency(start=.90);
  Modelica.Thermal.Interfaces.Node_c thermal;
equation
  thermal.q[1] = -i^2*R*efficiency;
end HeaterElement;
```

Another example of a mixed domain model is a rotational spring that changes shape in response to temperature changes:

```
model RotationalSpring
  extends Modelica.Mechanics.Rotational.Interfaces.Compliant;
  parameter Real c(final unit="N.m/rad", final min=0,
              start=1.0) "Spring stiffness";
  parameter Modelica.SIunits.Temperature T_nom(start=273.15)
    "Nominal temperature";
  parameter Real dudT(final unit="rad/L", start=1.0)
    "Angular expansion coefficient";
  parameter Modelica.SIunits.Angle unstretched_nom
    "Nominal unstretched length (at T_nom)";
  Modelica.SIunits.Angle phi_rel0(start=0)
    "Unstretched spring angle";
  Modelica.SIunits.Angle dphi "Delta phi";
  Modelica.Thermal.Interfaces.Node_c node_c;
protected
  Modelica.SIunits.Energy P;
equation
  dphi = (phi_rel - phi_rel0);
  P = c*dphi^2;
  der(P) = node_c.q[1];
  phi_rel0 = unstretched_nom + dudT*(node_c.T - T_nom);
  tau = c*dphi;
end RotationalSpring;
```

There are numerous other examples of mixed domain models (*e.g.*, temperature sensitive resistors) that could be included as well.

7 Examples

A complex example which exercises many of these components is to consider a control system regulating the temperature inside of a single story house. Figure 1 shows a sample problem consisting of two houses, two furnaces and two thermostats. The houses and the furnaces are identical but the thermostats are different.

One thermostat, shown on the left in Figure 2, is a mechanical thermostat. The furnace is controlled my a mercury switch mounted on mechanism which moves as a result of thermal expansion. The other thermostat, shown on the right in Figure 2, is an electronic thermostat. The integrated circuit shown in Figure 2 represents the logic of the controller. The circuit uses the voltage drop across two of its pins to compute the temperature in the house (by inverting the temperature relationship in Equation 6). Based on this temperature and the logic of the controller the two pins leading to the furnace are shorted to turn on the furnace.

Figure 3 shows the diagram of the single story house. The model includes conduction of heat into the ground, convection of heat through the walls to the ambient air and heat transfer due to radiation from the roof.

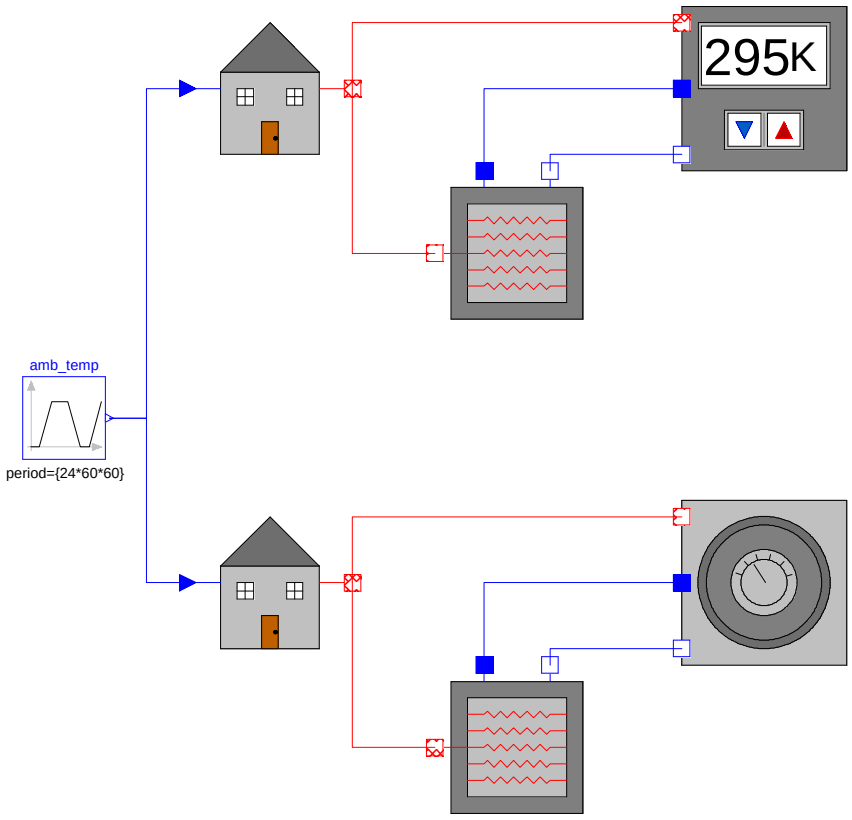

Figure 1: An example to exercise the `Modelica.Thermal` library

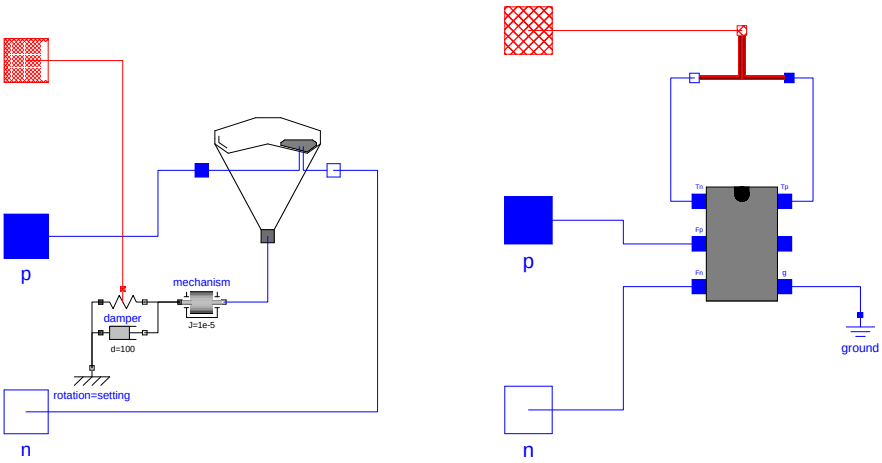

Figure 2: Mechanical thermostat (left) and digital thermostat (right)

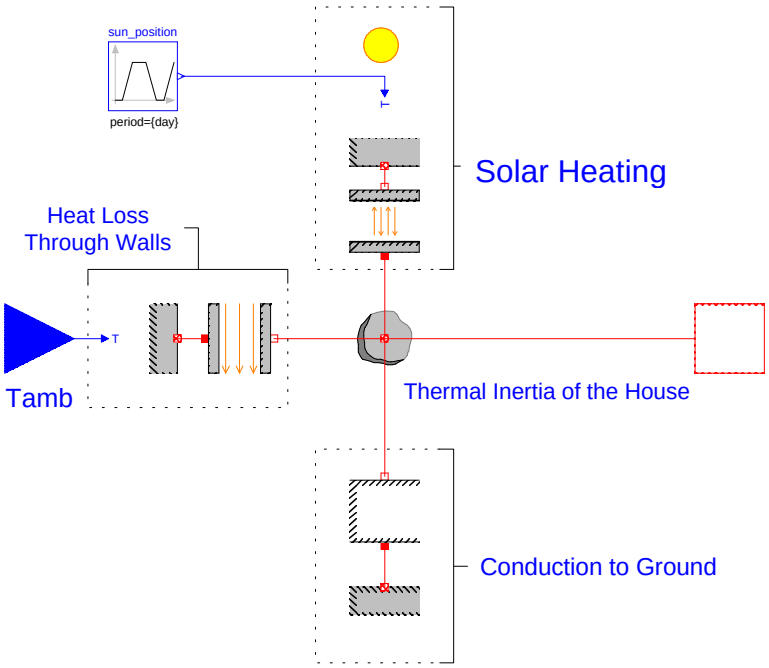

Figure 3: Diagram of heat transfer within the house

8 Conclusion

This paper serves several purposes. First, the paper shows how the thermal library `connector` was defined and also shows how numerous simple models were built from common constitutive relations. Another useful function of this paper is to demonstrate how model libraries are created. Note that the structure of the paper mimics the structure of the `Modelica.Thermal` `package`. Last, but not least, the paper discusses the current status of the thermal library. This is just a first draft of the library and should be considered a request for comments on how the library may be improved or expanded.

9 References

References

[1] *Omega's Temperature Handbook*, chapter Z, pages Z–201. Omega Engineering, Inc., "http://www.omega.com", 2000.

[2] Modelica Association. ModelicaAdditions.HeatFlow, http://www.modelica.org/library/library.html, 2000.

[3] P. Beater. Modeling and simulation of hydraulic systems in design and engineering education using Modelica and HyLib. In *Proceedings of the Modelica 2000 Workshop*, Lund, Sweden, October 2000. Modelica Association.

[4] Robert D. Cook, David S. Malkus, and Michael E. Plesha. *Concepts and Applications of Finite Element Analysis*. John Wiley and Sons, third edition, 1989.

[5] Claes Johnson. *Numerical Solution of Partial Differential Equations by the Finite Element Method*. Cambridge University Press, 1992.

[6] J. N. Reddy. *An Introduction to the Finite Element Method*. McGraw-Hill, 1984.

[7] H. Tummescheit, J. Eborn, and F. Wagner. Development of a Modelica base library for modeling of thermo-hydraulic systems. In *Proceedings of the Modelica 2000 Workshop*, Lund, Sweden, October 2000. Modelica Association.