Accessing external media data bases: The ExternalMedia Library

Francesco Casella

Dipartimento di Elettronica e Informazione
Politecnico di Milano

OpenOffice.org

The ExternalMedia Library

- The library comprises
  - A Modelica package
  - A C-code interface layer
    (Modelica external functions are only defined for C and Fortan)
  - A C++ engine, communicating with the external code
- End-user features:
  - 100% Compatible with Modelica.Media
  - Allows to re-use code that cannot be written in Modelica
    because it must also be used in other contexts
  - Easy access to thousands of fluids through, e.g., NIST's RefProp library
- Developer features
  - Complete Modelica framework
    (just add a modifier with the name of your external solver)
  - 95% complete C++ framework, to add a new external solver:
    - Add code to the SolverMap C++ class
    - Develop a child of the BaseSolver C++ class, reimplementing the setState_XX functions
- Default external solver already implemented: FluidProp from TUDelft
  (COM-based, optionally include the full RefProp database from NIST)

The ExternalMedia library – Basic principles

- The Modelica model calls the medium's setState_XX() function
- The medium package setState_XX() functions call the exernal functions of the C layer, passing the input data and some string medium identifiers
- Based on the strings, the interface layer dispatches the request to the right solver (multiple solvers can be used simultaneously)
- A cache record is allocated on a (large enough) circular buffer, containing all the medium properties
- The cache record is filled in with all (or part of) the medium properties
- The setState_XX() function returns a ThermodynamicState function with p, T, h, d, and an integer uniqueID
- When any medium property function is called (e.g. Medium.density(state)), the uniqueID is used to retrieve the results from the cache
- It is possible not to compute all values at once, managing later computations in the C++ solver class
- BaseProperties models set one uniqueID at initialization and always use that (no need of a circular buffer)

FluidProp

- Software for the computation of thermophysical properties of fluids
- COM interface to Ms Excel, Visual Basic, Visual C++, Maple, Matlab/Simulink & other programs supporting COM
- Beta version for Linux available
- Developed at TU Delft
- Available free of charge from http://fluidprop.tudelft.nl/ (100€ donation suggested if used for serious purposes)
- Databases
  - GasMIx: ideal gas mixtures
  - IF97: water/steam model
  - StanMix: describes fluid mixtures with cubic EoS
  - TPSI: accurate models of selected fluids
  - RefProp: interface to the RefProp NIST database of organic fluids and refrigerants (requires separate license)
- At the moment, only pure fluids or predefined mixtures (no composition vector required) are accessible via the ExternalMedia interface
- Documentation available in the .hlp file (see installation directory)

ExternalMedia medium models via FluidProp

- Install FluidProp
- Copy
  - ExternalMedia.dll to C:\Windows\system32 (WinXP),
    or to any directory defined in the system PATH variable (Win7)
  - ExternalMedia.lib to Dymola\bin\lib
  - externalmedialib.h to Dymola\Source
- Load ExternalMedia library
- Extend ExternalMedia.Media.ExternalTwoPhaseMedium
  - set library name to FluidProp.IF97 (or .StanMix, .TPSI, .RefProp)
  - set substance names array with the FluidProp name of the medium
  - the medium name is only used for documentation purposes (has no effect)

```
package WaterTPSI
  extends ExternalMedia.Media.ExternalTwoPhaseMedium(
    mediumName = "Water",
    libraryName = "FluidProp.TPSI",
    substanceNames = {"H2O"});
end WaterTPSI;

package CarbonDioxide
  extends ExternalMedia.Media.ExternalTwoPhaseMedium(
    mediumName = "Carbon Dioxide",
    libraryName = "FluidProp.RefProp",
    substanceNames = {"CO2"});
end CarbonDioxide;
```

Interfacing your own external solver

- Download Visual Studio projects and source code from SVN repo: https://svn.modelica.org/projects/ExternalMediaLibrary/trunk
- Develop a child of the BaseSolver class, reimplementing the setState_XX() functions so that they call your external solver appropriately
- The existing TestMedium and FluidProp solvers can be used as a template
- Modify the SolverMap::addSolver() function so that it recognizes the tag of your solver (passed as libraryName from the Modelica package) and instantiates it appropriately
- Set appropriate flags within include.h
- Recompile to static library or dll using MS Visual Studio 2005/2008

If you want to help extending ExternalMedia to fluid mixtures and/or porting it to other environments (e.g. OpenModelica) and/or OS (Linux), contact francesco.casella@polimi.it!

References

- Francesco Casella and Christoph C. Richter, “ExternalMedia: a Library for Easy Re-Use of External Fluid Property Code in Modelica”. In Proceedings 6th International Modelica Conference, Bielefeld, Germany, Mar. 3-4, 2008, pp. 157-161.
- User manual and code documentation (contained in the package)