Building Simulation 2009

*Eleventh International IBPSA Conference*
**Glasgow, Scotland**
**July 27-30, 2009**

# COUPLING OF TRNSYS WITH SIMULINK – A METHOD TO AUTOMATICALLY EXPORT AND USE TRNSYS MODELS WITHIN SIMULINK AND VICE VERSA

P. Riederer, W. Keilholz, V. Ducreux
*Centre Scientifique et Technique du Bâtiment, 290, Route des Lucioles, 06904 Sophia Antipolis Cedex, France
Corresponding author : peter.riederer@cstb.fr

## ABSTRACT

A large variety of simulation environments exists for building and system simulation. Collaborative work is sometimes time-consuming since, in the different steps of building and system conception and optimization, different tools have to be used, each of them specifically dedicated to a particular problem: for example the overall conception of a building can be done using the TRNSYS simulation environment, while optimization of control strategies is likely to be done using the Matlab/Simulink simulation environment. The same system and building is thus modelled several times in order to be able to simulate in the different environments.

This paper describes a methodology to export the models contained in the TRNSYS model library as well as non standard models (calles “types” in TRNSYS terminology) into the Simulink environment, and to use them in a Simulink model assembly. Almost any TRNSYS model can be used in the Simulink environment in this way.

The paper illustrates the methodology for calling TRNSYS types within Matlab or Simulink and gives advices for integrating those models into existing model assemblies. An automatic routine for exporting TRNSYS types from the TRNSYS Simulation Studio is presented as well as validation examples for the coupling..

## INTRODUCTION

Most modern simulation environments have a modular software architecture, in the sense that the different physical system components (pumps, ducts, buildings, …) are represented as components (modules) which are interconnected in order to define a given, particular system. Components are typically defined by an interface (Application Programming Interface, API) and an implementation (algorithm, set of equations, rules, charts, etc.).

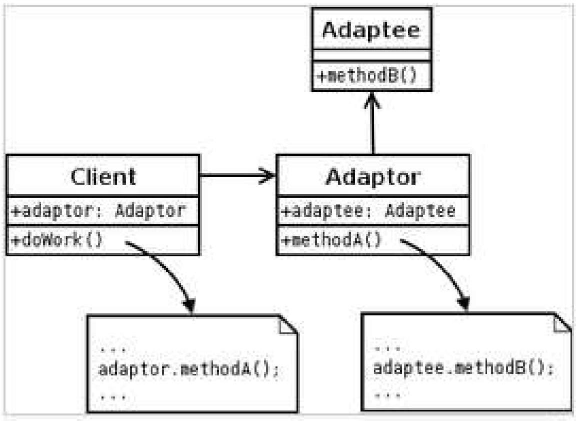
The component interface defines the way the component will communicate with the rest of the system: it enumerates the variables used by the component (component inputs) and the values computed by the component (component outputs) which can potentially be passed on to other components.

The component implementation defines the algorithm which allows computing output values from input values.

Both the syntax and the semantics of these component definitions vary between different simulation environments. However, different environments often use the same underlying operating system technology to implement these concepts. Under the WINDOWS operating system, the Dynamic Link Library (DLL) technology is a commonly used approach for implementing components.

It is usually possible to convert component models from one environment to another (using the same underlying technology) by adapting an existing DLL for a new simulation environment using a traditional programming design pattern known as “Adapter”.

In computer programming, the adapter design pattern (often referred to as the wrapper pattern or simply a wrapper) translates one interface into a compatible interface.



The object adapter pattern expressed in UML. The adapter *hides* the adaptee's interface from the client.

*Figure 1: the adaptor design pattern (source: Wikipedia)*

An adapter allows modules to work together that normally could not work together because their interfaces are incompatible, by providing its interface to clients while using the original interface. The adapter translates calls to its interface into calls to the original interface. The amount of code necessary to do this is typically small. The adapter is also responsible for transforming data into appropriate forms.

This article shows how to “wrap” TRNSYS component models for use with MATLAB/SIMULINK and vice versa using adaptors, as well as how to automate the process. Indications about the use of such adapted models are also given.

## TRNSYS ARCHITECTURE

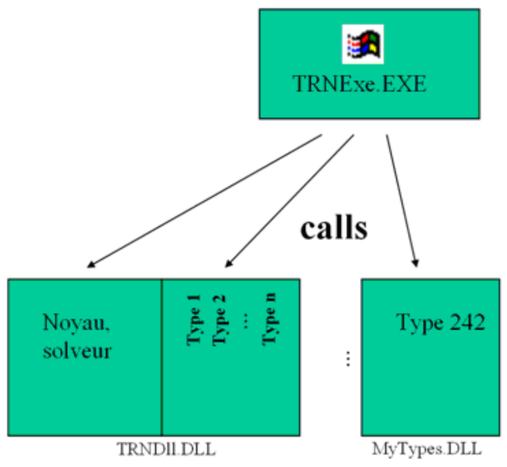
The TRNSYS Simulation Studio® is a complete and extensible simulation environment for the transient simulation of systems, including multi-zone buildings. It is composed of a graphical user interface (Simulation Studio), a simulation kernel and component models called ‘TYPES’. A simulation project is defined as a set of component models (solar collectors, pumps, multi zone building, weather data reader, …), which are interconnected in order to define a specific system (solar domestic hot water system, wind energy plant, building equipped with different systems, etc.).

A connection between two components represents information flow – e.g. a fluid temperature computed as the output of a ‘solar collector’ component which is connected to a ‘auxiliary heater’ component as an input.

The simulation kernel will read a project definition file (called the ‘deck’), analyze it and perform a transient simulation to compute the outputs requested by the user. The kernel uses one of its built-in solvers to compute the results requested by the user, calling each component used in the project in an iterative way, until convergence is reached during each time step. Successive substitution is the most frequently used solution method. TRNSYS uses a fixed time step.

A TRNSYS component is defined as function computing output variables based on inputs (variables computed by other components) and parameters (constant values). They are implemented as WINDOWS DLLs, respecting a well-defined API (Application Programming Interface). The API defines, among other things, the component function’s signature – the number and types of the arguments used. A TRNSYS component can thus be implemented using any programming language able to compile WINDOWS DLLs (e.g. C, C++, FORTRAN, PASCAL, …). Besides the input variables and parameters, the function implementing the TRNSYS component has arguments to provide control information about the current state of the simulation (e.g. if convergence has been reached by all components, the number of iterative calls in the current time step, and so forth).

TRNSYS components are described by PROFORMA (*.tmf) files. These files only contain the component’s variables and graphical representation, but not the actual algorithm (which resides in the DLL).



```
SUBROUTINE TYPE242 (TIME,XIN,OUT,T,DTDT,PAR,INFO,ICNTRL,*)
```

| | |
|---|---|
| TIME - | The simulation time which can be used in the component |
| XIN( ) - | Inputs |
| OUT( ) - | Outputs for the component |
| T( ) & DTDT( ) - | Arrays used to evaluate derivatives within the component |
| PAR ( ) - | Parameters |
| INFO( ) - | Information about the component |
| ICNTRL( ) - | Information for new control statements (used with solver 1 only) |
| * - | Mark for alternate return |

*Figure 2: TRNSYS software architecture and component function (entry point) signature (a part of the TRNSYS API)*

The TRNSYS kernel provides a certain number of auxiliary kernel functions which can be called by the function implementing a component. Examples include functions to determine the start and end time of the simulation (as defined in the currently executed simulation project), functions to check to component’s configuration with respect to the user’s input, mechanisms to store values between time steps, etc. These functions are implemented, together with a set of standard components, in a DLL called TRNDll.dll.

TRNSYS offers the possibility to group a set component models together to form a new component, called “Macro” in TRNSYS terminology.

## MATLAB ARCHITECTURE

MATLAB® is a general purpose, high-level language and interactive environment allowing performing computationally intensive tasks. It is widely used in industry and research in a variety of application domains, ranging from signal and image processing, communications, control design, test and measurement, financial modelling / analysis, and computational biology to thermal system and building simulation.

One of the numerous MATLAB extensions, Simulink®, is an environment for multi domain simulation and model-based design for dynamic and embedded systems. It provides an interactive graphical environment and a customizable set of component libraries (called block libraries in MATLAB terminology) that allow to design, simulate, implement, and test systems.

Connections between SIMULINK components represent information flow. Contrary to TRNSYS connections, SIMULINK allows connecting not only simple variables, but entire vectors or matrices, and the output variables of a component may not only be simple values, but also vectors or matrices.

SIMULINK uses one of several solvers built into MATLAB to compute the results requested by the user. Depending on the solver chosen by the user, this may be an iterative process, using discrete or continuous values. MATLAB can use fixed or variable time steps.

MATLAB components (blocks) compute a series of output variables as a function of input variables. Components can be defined in a variety of ways, including sets of equations directly entered into the environment and calling external functions. A very rich and complete API (Application Programming Interface), allowing to create new blocks using a programming language, is provided. Like for TRNSYS, the program defining the component's algorithm can be compiled into a WINDOWS Dll.

SIMULINK offers the possibility to group a set component models together to form a new component, called "subsystem" in SIMULINK terminology.

## USING TRNSYS COMPONENTS IN MATLAB

### THE CONCEPT

As both TRNSYS and MATLAB components can reside in WINDOWS DLLs, it is easy to adapt an existing TRNSYS component for use with MATLAB and/or Simulink: we simply need to encapsulate the TRNSYS component using the MATLAB API, applying the 'adaptor' design pattern. This can be done using in a Matlab script or in SIMULINK using the "embedded function", which will pass its input values (which are passed through MATLAB data structures defined in the MATLAB API) to the TRNSYS component (using TRNSYS data structures and calling the TRNSYS component's function), recover the values computed by the component, and return them to MATLAB, again using MATLAB data structures.

In the embedded MATLAB function, the TRNSYS function's arguments concerning control information about the current state of the simulation must be filled correctly, based on the current state of the MATLAB simulation.

This first approach will work for very simple components, which only contain basic computation. Complications occur if the component to be used makes use of TRNSYS kernel functions – which is probably the case for more than 99% of all existing components. To be able to handle these cases, it is important that the TRNSYS kernel be initialized when the TRNSYS DLL is called by MATLAB. In other words, TRNSYS kernel functions allowing to query values like simulation start and stop time, etc., must return coherent values during this process. The initialization is done by functions added to the TRNSYS kernel for this purpose. (They should be standard functions in TRNSYS 17). They are called during simulation startup in the StartFcn callback of the MATLAB embedded function.

The initialization of the TRNSYS kernel also includes opening so-called 'external files', if the component uses this concept. External files contain configuration parameters for a given component. Unfortunately, they are opened by the TRNSYS kernel, not the component itself. We therefore must open these files this during the initialization of the MATLAB simulation.

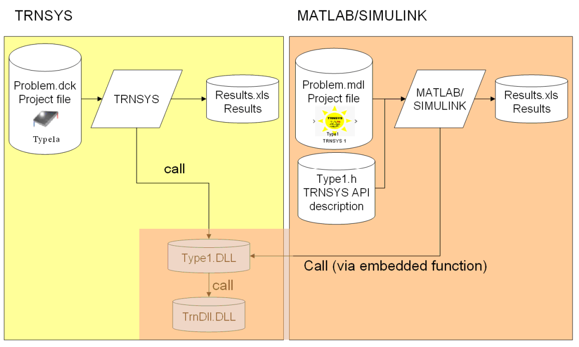


*Figure 3: Calling TRNSYS components from MATLAB*

The process of loading and calling the TRNSYS DLL is as follows:

During initialisation:

- Generate "Typexx.h"
- Generate list of model parameters
- Initialise variables (inputs and outputs) as well as derivatives
- Attach files
- Load DLL
- Gerenate INFO array before each separate call
- Set TRNSYS Version, Timestep, StartTime, StopTime
- Initialisation (3 calls as shown in figure)

At each time step:

- Increment number of calls and iterations, create INFO array
- Call Dll
- Save states and derivatives
- Use outputs

### IMPLEMENTATION - AUTOMATIC GENERATION FROM SIMULATION STUDIO

The generation of MATLAB subsystems from TRNSYS components has been implemented into TRNSYS' graphical user interface, Simulation Studio. The user can simply choose a component in a simulation project and select "File/Export to Simulink". This will produce the C header file needed by MATLAB to execute the component, as well as a project file containing a subsystem, an input vector and default output components. The subsystem contains all necessary code to set up the TRNSYS kernel (by calling the appropriate initialization functions), extract necessary information from MATLAB data structure, call the TRNSYS component and fill the subsystem's output vector with the values returned by the call.

The generated project opens automatically in SIMULINK and can be executed directly. As the generated input vector contains the default initial values and parameters from the original TRNSYS project, executing the simulation will yield the same result as if it were executed by TRNSYS.

For this to work, both the DLL containing the TRNSYS component and the kernel (TrnDll.dll) must be available for MATLAB/SIMULINK (included in the search PATH). The source code of the component is not required.

Simulation Studio also adds a SIMULINK mask to the generated project. It allows displaying the component's variable names (instead of generic vectors), required units and default values for inputs, parameters and derivatives (derivatives are a special TRNSYS concept that is not detailed here).

## USING MATLAB COMPONENTS IN TRNSYS

### THE CONCEPT

Another one of the numerous MATLAB extensions, real time workshop, provides mechanisms to compile Simulink subsystems into autonomous applications or DLLs. In Matlab, the Matlab compiler allows to generate the DLLs. The latter is not implemented yet. This module can be used to produce a DLL containing functions able to run the subsystem's algorithm.

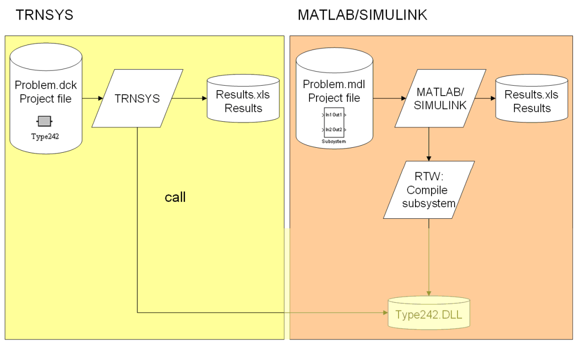


*Figure 6: Using a SIMULINK subsystem in TRNSYS*

As the MATLAB API generated by real time workshop is not compatible with the TRNSYS API, we must again 'wrap' this MATLAB function into a function respecting the signature for TRNYS component functions. The role of the wrapper is to initialize the MATLAB model, receive information from TRNSYS (variable values and simulation state), pass it to the function generated by MATLAB real time workshop, return the results to TRNSYS and finally to shut down the model at the end of the simulation.

### RTW TEMPLATE FOR THE GENERATION OF TRNSYS TYPES

The MATLAB real time workshop uses a pearl script to generate a complete compile project from the subsystem, including source code and a makefile. In this process, a template file is used in order to generate C or C++ code, defining the functions to be exported, including their names and calling conventions. This simplifies our task tremendously: we can simply provide a variant of the standard template, adding the adaptor. This way, the generated Dll remains a 'normal' MATLAB DLL, containing just an extra function – the one that makes it compatible with TRNSYS.

The generated source code uses the usual TRNSYS control information in order to initialize the component, execute its algorithm, or shut it down at the end of the simulation, by simply calling the appropriate MATLAB functions residing in the same DLL.

Using the templates we provide, MATLAB/SIMULINK users can thus simply choose a new compilation target called "TRNSYS". Generating code for a subsystem will now produce a compile project for Microsoft Visual Studio. Compilation using the C++ compiler in Microsoft Visual Studio then yields a TRNSYS compatible DLL, which can be directly used by TRNSYS.

In order to add the new component to the TRNSYS component library, the user must edit a component description, the PROFORMA file (*.tmf). This file contains the component's number, descriptions and variable names.

## NUMERICAL CONSIDERATIONS

While TRNSYS uses the successive substitition method, different solvers can be used in Matlab/Simulink. To date, validation examples are in progress to verify the use of TRNSYS DLLs in Matlab/Simulink. First validation cases showed correct results, but due to the variety of solvers in Matlab, the approach has to be validated for all cases.

One phenomenon has however been observed in the case of coupling several TRNSYS models in Simulink. The interconnection created an algebraic loop that stopped simulation in most cases, contrarely to Simulink models where algebraic loops can, in many cases, been solved. In order to overcome this problem, the algebraic loops have to be broken by inserting Simulink blocks without direct feed-through (e.g. the memory block). This approach is valid when small time steps are used as usually done in Simulink simulations. If simulations with hourly time steps shall be carried out this can cause false results.

## VALIDATION EXAMPLES

The validation section only shows the export of TRNSYS types to Simulink environment.

The comparison carried out is divided into several steps:

- Step 1: Comparison of single components (controllers etc.)
- Step 2: Comparison of single components with derivatives (storage tank etc.)
- Step 3: Comparison of single components importing files (type 56 etc.)
- Step 4: Comparison of model assemblies without control loops
- Step 5: Comparison of model assemblies including control loops

For steps 1-3, single TRNSYS components have been exported to Simulink. The results show identical numerical results in both environments, the results are thus not shown here. Simulation time is however faster in TRNSYS as in Matlab/Simulink (about factor 10).

In this paper, one validation related to step 4 is presented. The SDHW example in the TRNSYS projects has been chosen to validate the coupling. The model is shown in Figure 7.

The controller acting the pump of the solar loop has been replaced by a simple forcing function imposing a flowrate scenario .

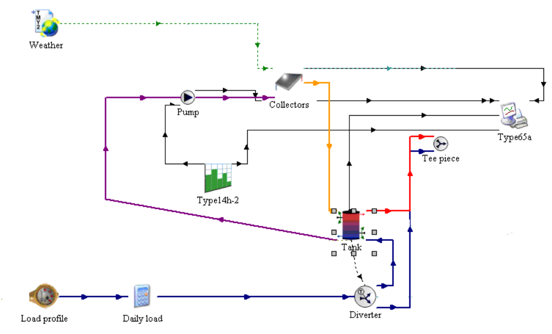


*Figure 7 : The SDHW example in the TRNSYS environment*

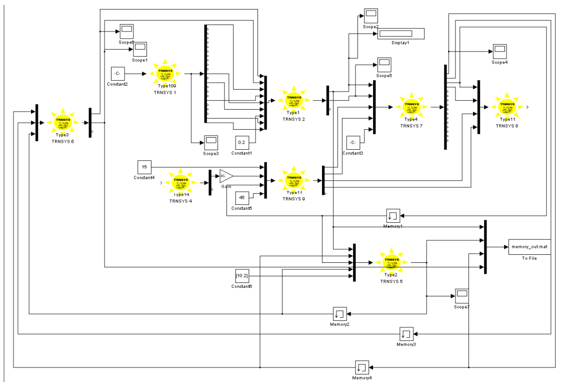


*Figure 8: The SDHW example in Simulink environment using TRNSYS DLLs*

For the comparison, all components of the TRNSYS environment have been exported to Simulink and coupled (Figure 8). All loops have been broken by memory blocks. The time step has been chosen to 120 seconds.

As already stated in a previous section, the insertion of memory blocks will result in errors in the simulation results. These are presented in Figures 9 and 10. The results agree well except for high solar radiation periods, where differences of about 1-1.5 K are observed. More detailed results will be presented in the future in order to quantify simulation errors in terms of power or energy.

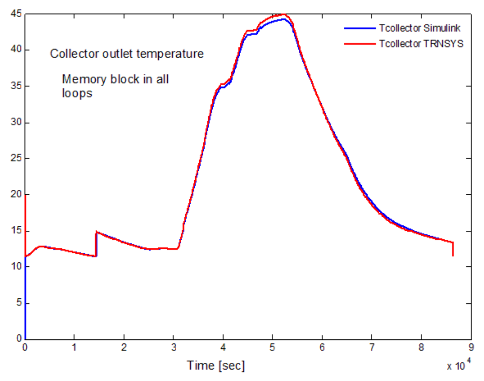


*Figure 9: Comparison of collector outlet temperatures in TRNSYS and Simulink*

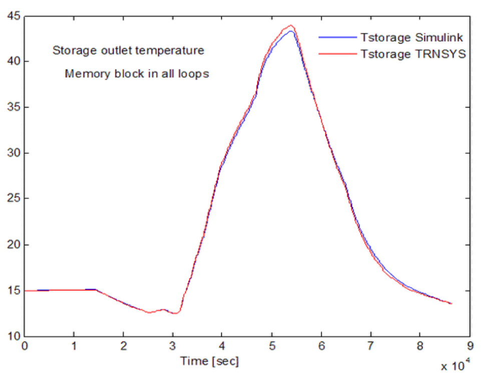


*Figure 10: Comparison of storage tank outlet temperatures in TRNSYS and Simulink*

## FUTURE WORK

### HANDLING VECTORS

This first version only handles MATLAB/ SIMULINK models which use real values as inputs and outputs. Subsystems using vectors and/or matrices must first be encapsulated in a higher-level subsystem in order to serialize the vector/matrix data (using Mux and Demux blocks). This process could be automated and will be object of future work.

### BUILDING DESCRIPTION

The mechanisms described in this article allow to re-use existing components designed for one simulation environment in another simulation environment. For example, MATLAB/SIMULINK users can gain access to detailed building models, such as TRNSYS type 56.

In order to execute an existing TRNSYS simulation project in MATLAB/SIMULINK, however, it is not sufficient to make the TRNSYS components (algorithms) accessible. It is also necessary to translate the project itself (i.e. the way components are interconnected in a given project).

The NBDM (Neutral Building Data Model) project proposes a neutral data format in order to represent a building simulation project in a neutral way. Translators from and to this format exist for all major building simulation tools used in France, and also for TRNSYS. It could be interesting to extend NBDM by the system description (currently only the building description is covered).

### EXTENSION TO OTHER COMPILERS

Currently, users need to have access to Microsoft Visual Studio in order to compile MATLAB subsystems for TRNSYS. It might be useful to adapt the templates for other compilers, such as the one built into MATLAB real time workshop, thus removing one external tool from chain.

## CONCLUSION

This article has described how DLL-based components designed for one simulation environment can be used in another simulation environment. The concept has been illustrated by automating the adaption of TRNSYS components to MATLAB/SIMULINK and vice versa. The resulting components can be distributed and used without access to the source code.

The validation component per component showed identical results, the coupling is thus valid if only one component is used in the other environment. However, if an assembly of models is exporter and "re-assembled" in the other environment, the validity has to be verified case by case: in the example of the solar hot water heater, algebraic loops had to be broken manually by the memory block in Simulink environment. The results showed good agreement, but they were not exactly the same.

## ACKNOWLEDGEMENT

The work presented in this paper has been carried out in the frame of the French Dynasimul project, funded by the ANR (Agence Nationale de Recherche).

## REFERENCES

Keilholz W. 2007. NBDM: A neutral data model to link building energy performance simulation tools, www.buildingsmart.fr/documents/stand-inn-sophia-antipolis/09_nbdm.pdf

Werner Keilholz, 2008: Bernard Ferries, Franck Andrieux, Jean Noel: A Simple, Neutral Building Data Model; ECPPM, Sophia Antipolis, September 13, 2008

SIMBAD, 2004. SIMBAD Building and HVAC Toolbox, Version 4.0, CSTB, France

Simulink, 2004. Simulink dynamic System Simulation for Matlab. Version 6.0, Mathworks Inc., Ma., USA.

Trnsys, 2000. Trnsys: a transient system simulation program. SEL, University of Wisconsin, Madison USA.

MATLAB: http://www.mathworks.com/

TRNSYS: http://software.cstb.fr

MSDN: http://msdn.microsoft.com/

Wikipedia : http://www.wikipedia.org