Research on Solar Processes

Project Final Report

**Submitted by the
SOLAR ENERGY LABORATORY
University of Wisconsin - Madison**

October 1999

**Prepared for the US Department of Energy
Under Subcontract No.
DE-FC36-96G010152**

DISCLAIMER

This report was prepared as an account of work sponsored by an agency of the United States Government. Neither the United States Government nor any agency thereof, nor any of their employees, make any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government or any agency thereof. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States Government or any agency thereof.

DISCLAIMER

Portions of this document may be illegible in electronic image products. Images are produced from the best available original document.

Research on Solar Processes
Project Final Report

Submitted by the
SOLAR ENERGY LABORATORY
University of Wisconsin - Madison

October 1999

Prepared for the US Department of Energy
Under Subcontract No. DE-FC36-96G010152

The University of Wisconsin Solar Energy Laboratory (UWSEL) has been supported by the US Department of Energy through the Golden Field office. The contract was moved to National Renewable Energy Laboratory (NREL) for the period of May 26, 1999 through May 25, 2000. The majority of the projects were to be completed after the end of the DoE contract period (December 1999) since continued DoE funding was anticipated. Although the DoE funding has continued, it is now monitored through NREL. This report summarizes work to date. Two MS theses will be submitted to NREL in December 1999 as part of the current contract. Copies will also be sent to DoE..

Task 1: Thermal Analysis Simulation Tool Maintenance and User Support.

The SEL has addressed a number of issues with the developments to TRNSYS in anticipation of release of version 15. The two major developments have been a complete overhaul of the IISiBat graphical user interface and the addition

IISiBat

The Center for the Scientific Study of Buildings (CSTB) in Sophia Antipolis France has completely rewritten IISiBat as a native Windows program, replacing the current version written in LISP. LISP, while it allows programs to run on both Windows and Unix platforms cannot make use of many of the standard Windows programming tools. Consequently the current IISiBat version (2.0) feels rather clumsy to use.

Other additions to IISiBat include the ability to create multi-layered system diagrams, and the ability to automatically generate html based documentation for new Types that the user has developed and wishes to add to the standard TRNSYS library.

Component Changes, Additions and Deletions

At the time of TRNSYS's inception, computers were very different than the ones used today. A great many precautions were taken to avoid unnecessary lines in source code. Consequently, there have always been two types of inputs to a TRNSYS Type; PARAMETERS, which do not change with time and are only read once at the start of a simulation and INPUTS, whose values do change with time and which are read in at every timestep. As computer processing speed increases, the distinction between these two types of input becomes less important. Consequently, various PARAMETERS have been moved to INPUTS so as to make the Types more flexible. Specifically the dead band temperatures on the controller model and the set point temperature on the auxiliary heater have been changed. It is probable that in future versions of TRNSYS PARAMETERS will be removed entirely.

One of the distinct advantages of TRNSYS is its modularity. From a user's point of view a separate Type models each piece of equipment in the system. In a few cases, however different pieces of equipment have been lumped together as different modes of the same Type. The solar collector model is a prime example. Type1 Mode 1 models a standard flat plate collector. Switch the mode to 4 and you have a concentrating parabolic collector: not the same piece of equipment at all. Consequently Types that suffer from this problem have been broken up into different models. A related problem has been in combined subsystems. These are types that model groups of equipment instead of single pieces. All the combined subsystem types have been removed from the standard TRNSYS library. They are still available from the SEL website for download and can be relinked by any TRNSYS user.

A number of new Types have been added to the TRNSYS library

The new TRNSYS types are:

- Diesel Generator

- UA House: A simple building zone model that takes various sensible and latent gains into account.

- Wind Turbine

- Auxiliary Cooling Device: The compliment to the Type6 Auxiliary heater, this Type removes energy from a fluid flow so that it meets a specified set point.

- Photovoltaic Array: This Type has the capability of modeling both single crystal and thin film photovoltaic modules

- Holiday Calculator: computes the dates on which the standard US legal holidays will fall in a given year. It can also read a file of holiday dates.

- Utility Rate Schedule Processor: This Type calculates electric bills for grid-tied buildings by interpreting utility rate schedules and applying them to the building's demand.. It may be used to determine savings in buildings employing electric parallel generation.

New Functionality

The SEL has used three techniques for allowing TRNSYS to make use of other program's strengths. These are communicating with other programs using Windows API calls, calling DLLs and using dynamic data exchange.

Calling (and being called by) other programs

Perhaps the most basic mode of communication between two programs is to allow one program to "call" another. In other words, while one program is running, allow it to launch a second program, wait for completion of a task and then continue its calculations. A distinct advantage of this solution is that the second program need not be modified. However, the process of calling and waiting is slow, especially in simulations where the second program must be called at each timestep.

Windows API calls

TRNSYS currently contains a number of utility routines: subroutines that perform common tasks such as the calculation of thermodynamic properties of fluids. These utility routines are called from within component models such as a pump or a chiller. In order to allow calls to external programs, a utility subroutine named "CALLPROGRAM" has been created. CALLPROGRAM uses several Win32 Application Programming Interface (API) commands to start the external program. API commands directly control the Windows operating system.

Depending on the mode specified in the call statement, CALLPROGRAM will either wait for the second program to finish its task before proceeding or will merely start the second program or then proceed with other TRNSYS calculations. The call statement for CALLPROGRAM is:

CALL CALLPROGRAM(CMDLINE,bwait,prochand, thrdhand)

Where:

CMDLINE is the command line text string containing the path and name for the executable program

BWAIT is a logical variable that determines whether TRNSYS will wait for the second program to finish running or not.

PROCHAND and THRDHAND are Windows variables that may be useful in some cases.

In order to make use of the information generated by the second program, it was necessary to develop a method for inserting data into the TRNSYS input file. Until now, the TRNSYS input file has been one continuous text file containing all the information that describes the system being simulated. However, in certain situations it would be advantageous to have part of the TRNSYS input file (for example, equations passed from another program) in a separate file and reference it with an INCLUDE statement from the main file. In this way, it is possible to change items or rewrite the include file without changing the main TRNSYS file. TRNSYS treats all the statements in the include file as though they were in the main file. The syntax would look like:

INCLUDE c:\trnwin\file.inc

Calling Dynamic Link Libraries

Because of the slow speed, calling an external program and waiting for it to finish is often an unacceptable solution. However more and more often, Windows based programs are comprised of both an executable, and a dynamic link library (DLL). The executable contains all the program's peripheral tools as well as the interface and calls the DLL, which is essentially the compiled source code. The DLL can be accessed quickly because none of the program's overhead actions, such as screen refreshing, are required. Another distinct advantage to interacting directly with DLLs is that since a DLL is compiled and linked, its original language is irrelevant. Simulation programs that can access external DLLs can use component models written in any language.

A component was added to TRNSYS that makes calls to DLLs possible. Type 61, allows a user to write a component in any language (as long as the language supports DLLs) and then link that component into TRNSYS. The code of Type 61 handles the standard TRNSYS initialization routines and then calls the DLL called "EXTDLL". The calling command for the DLL, within Type 61, is:

CALL EXTDLL(Spass,Sarraypass,simarray,xin,out,t,dtdt, par,info,icntrl)

TRNSYS passes its standard arrays of information to the DLL (parameters, info, inputs, etc.) but also passes several other arrays. In this manner, EXTDLL has every capability available to a normal TRNSYS component. The three additional arrays contain information accessible to other TRNSYS components through FORTRAN common blocks. The SPASS array is a character string that can be used for passing either error or text messages back to TRNSYS from the external DLL. The SARRAYPASS array is a 100-place array set aside for values

that need to be saved from one timestep to the next. This capability is normally provided to standard TRNSYS components through the S-array. The SIMARRAY array passes the current timestep value, the simulation start time, stop time and timestep values to the DLL.

As an alternative to using Type 61, it is possible to write a DLL using the standard TRNSYS call statement, place the DLL in the TRNSYS "userlib" directory and call it as though it were a normal TRNSYS component. No matter what language the component is written in, it can be directly used within the TRNSYS simulation. Unlike the EXTDLL call described above, the SPASS, SIMARRAY, and SARRAYPASS arrays are not included. The advantage of this method is that adding components becomes extremely simple; it suffices to place them into the proper "userlib" directory.

The capability of loading and using external DLL's has been successfully applied to REFPROP [2], a program designed to calculate the thermodynamic properties of pure refrigerant fluids as well as those of user defined mixtures. The connection to REFPROP can replace the TRNSYS fluids subroutine, which makes the same calculations for a limited selection of refrigerants.

Using Dynamic Data Exchange

The third method used to interconnect simulation programs is Dynamic Data Exchange (DDE); a method of sending messages to between programs through the Microsoft Windows operating system. DDE is a special set of Windows API commands that allow two programs to communicate. Many programs support DDE, including the applications within Microsoft Office and Engineering Equation Solver [3]. Engineering Equation Solver (EES) is a non-linear equation solver that incorporates two distinct advantages for energy simulation work. First, EES allows equations to be entered in any order with unknown variables placed anywhere in the equations. Second, EES provides numerous built-in mathematical and thermophysical property functions. For example, the steam tables are implemented such that any thermodynamic property can be obtained from a built-in function call in terms of any two other properties. Unfortunately, researchers wishing to use EES models with TRNSYS need to rewrite them in FORTRAN. Using DDE, however, removes this necessity.

A TRNSYS component (Type66) performs all the necessary DDE message handling between TRNSYS and EES as shown in Figure 1.

Fig. 1: Type66 DDE Message Handling Between TRNSYS and EES

Type 66 allows the user to call an EES file, pass it inputs and receive outputs from EES using DDE and text files to exchange the information and commands. To begin, the user creates a model in EES, and then decides which variables should be inputs and which should be outputs. The variables chosen to be inputs are placed in an include file automatically. The

EES program is set up to look for the include file, manipulate the inputs and then set the outputs as shown in Figure 2.

```
$include E:\EES32\XYZ.INC
x^2 = Out[1]
x + y = Out[2]
```

Fig. 2: A sample EES file

The user then enters information into the TRNSYS input file in the form of a standard type description. An example is shown in Figure 3:

```
UNIT 66 TYPE 66 CALL EES XYZ
INPUTS 1
xvalue yvalue
x      y
LABELS 3
*the command line to run ees
c:\EES32\EES.EXE c:\EES32\XYZ.EES
*the include file to write
c:\EES32\XYZ.INC
*the output file that ees writes
c:\EES32\XYZ.OUT
```

Fig. 3: TRNSYS input file call to EES

The above type description is similar to a standard TRNSYS type description with the exception that there are no parameters, only inputs. The first line of the inputs specifies either the variable names or the TRNSYS component outputs that contain the values of the inputs at each time step. The second line of inputs contains the variable names that are to be printed in the include file by Type66. The assignment of variable names is necessary because EES is expecting them to have specific names. There are also three LABELS associated with this type. The first label is the command line that runs the EES file. This tells TRNSYS where to look for EES, where to find the EES file to be run and to start EES if it is not already running. The second LABEL is the name of the Include file into which TRNSYS will place the input values and their labels. The third LABEL is the name and location of the output file from which TRNSYS will read the EES solution.

In the course of a simulation timestep, new values for the inputs (xvalue and yvalue in this case) are set by other components in the simulation. These new values, along with the labels ("x" and "y") are written to the EES include file. Then, Type 66 sends a DDE message to EES to open and calculate the EES file (c:\EES32\XYZ.EES). EES does this calculation and writes the results to an output file (c:\EES32\xyz.out). Type 66 then opens the output file, reads in the values for the OUT array (OUT[1] and OUT[2]) and passes these values to the TRNSYS solver to be used as inputs for other components.

Task 2: Provide design guidelines for reduced caost SDHW systems.

The development of TRNSED allows engineers without TRNSYS experience to make design changes and simulate the SDHW performance over any time period and in any location. A TRNSED program will be developed to simulate many different types of SDHW systems. The program will be arranged so that either of two levels of information can be specified for each component: simple and detailed. This two-level model for each component will allow a company specializing in, for example, heat exchangers to investigate the impact of design changes in their equipment on system performance without the necessity of having to specify

details of other components. For task to be useful to the SDHW industry, it will be necessary to keep abreast of suggested cost reduction proposals (none-proprietary ones) and ensure that TRNSYS is capable of modeling and analyzing innovative designs.

The first component to receive detailed modelling is the collector in the Solar Collector Design Program (CoDePro). Comparisons have been made with simulated results and with some solar collector test reports. The program predicts the constant term ($F_r(\tau\alpha)$) of the efficiency equation very well while it does not calculate coefficients of the 1st and 2nd order terms accurately for some collector models.

To improve the accuracy of CoDePro, all the factors and values related to the heat loss coefficient are being checked. The values of emittance of the absorber, plate-tube bond conductance, and conductivity of the edge and back insulation should be verified. Various configurations of collector absorber will be added in the CoDePro. The convection heat transfer between the collector cover and the absorber will be studied for more accurate calculation.

A detailed SDHW heat exchanger model was developed using EES and the ϵ -NTU method. As one model of heat exchanger design program, the thermosyphon heat exchangers for solar water heaters were studied. The heat exchanger design program will include the external (2 pump), internal (1 pump), and thermosyphon heat exchanger.

According to ASHRAE Standard 95-1981, methods of testing to determine the thermal performance of solar domestic water heating system, an EES code was programmed to calculate the SDHW system performance.

The final report, in the form of a Master of Science thesis in Mechanical Engineering, will be available in January 2000.

Task 3: Provide the tools to use equipment catalog data in TRNSYS.

In a previous DoE funded project a technique was developed to predict the performance of a variety of heat exchangers outside of the range normally provided in a manufacturers catalog. The technique developed in that project is being extended to rooftop packaged air conditioning systems. The goal is to develop a model of a rooftop packaged air conditioning system by using available manufacturer's catalog data only. The model will predict the catalog data accurately and, more important, allow a satisfactory extrapolation over a wide range of operating conditions where catalog data is not available.

To develop the model, the different components of the system such as compressor, evaporator, condenser and expansion valve have to be modeled. Different approaches exist to model the behavior of those components.

One method for modeling the performance of a component is to use a curve fit approach using a polynomial representation. It is very easy to represent the characteristics of a component given enough catalog data points. However, the resulting equations have no physical meaning and a prediction of the component performance outside the range of the fitted data may be very inaccurate.

Another modeling technique is to describe the behavior of a component by using mechanistic models. This requires a detailed knowledge of characteristic parameters such as geometric

specifications, fin efficiencies, or other quantities. Such detailed information is generally not found in manufacturer's catalogs.

To avoid the problems of uncertain extrapolation with a polynomial representation and to accommodate the lack of specific information about the system's components for a complex model, a semi-mechanistic modeling technique is used in this project. The characteristic performance parameters for each component are first defined. Those performance parameters contain all specific characteristics of the system's components. Parameter values are then obtained by fitting the model to the catalog data. The resulting model allows reasonable extrapolation over a wide operating range.

The literature was searched for existing models for the system's components. Models for compressor and evaporator were found and were applied to model the system. A new model was developed for air-cooled condensers.

The compressor model used in this project is based on the volumetric efficiency model for compressors. The performance of the compressor is described by a set of equation with four compressor specific parameters. Two of them are needed to predict the mass flow rate and the other two are used for the prediction of the compressor power draw.

The evaporator model as well as the condenser model is based on the Ntu- effectiveness method. For both heat exchangers, evaporator and condenser, expressions for the overall heat transfer coefficient-area product were developed. Fundamental heat and mass transfer correlations, which were found in the literature, were used to model the heat transfer on the air side and refrigerant side. The correlations were modified, so that heat exchanger parameters can be obtained, which are a function of constants such as geometry and flow arrangement only. This modeling technique allows predicting the performance of the heat exchangers very accurately over a wide operating range, since variations in flow conditions and temperature level are taken into account by adjusting the overall heat transfer coefficient-area product

Each component model, the compressor model and heat exchanger models, was evaluated with manufacturer's catalog data and was found to be able to predict the performance very accurately.

These single component models were interconnected to model the vapor compression cycle of the rooftop packaged air conditioning unit. Thus, the system was modeled by a set of equations containing 10 parameters to predict the performance of the vapor compression cycle. These parameters are obtained by fitting the system model to the catalog data of the packaged unit.

The performance of rooftop packaged units of different sizes was predicted by using the model described above. The performance prediction for the unit capacity and the power draw was found to match the given catalog data very well. The model of the rooftop packaged air conditioning system was developed with the Engineering Equation Solver (EES). The parameter estimation was performed using the built-in EES optimization routines.

A software package will be written to easily create the models from catalog data. These models will then be available for use in TRNSYS. The final report, in the form of a Master of Science thesis in Mechanical Engineering, will be available in January 2000.