CHAPTER 2
THE TRNSYS INPUT FILE

2.1 INTRODUCTION

The concepts and definitions presented in Chapter 1 indicate the nature of the information required to describe a modular system consisting of interconnected components. The purpose of this chapter is to describe the format of the control statements and associated data in the TRNSYS input file required to simulate such a system.

TRNSYS control statements fall into three categories, simulation control statements, component control statements, and listing control statements. Simulation control statements direct the operation of the TRNSYS system and define such things as simulation length and error tolerances. Component control statements define the components in the system to be simulated and their interconnections. Listing control statements affect the output of the TRNSYS processor, and the END statement signals the end of the TRNSYS input file. Some control statements must be followed by data that provide the additional information required by TRNSYS.

The format of control statement lines is flexible: they need not start in column one, but at least one blank space or a comma must separate each item on a line. TRNSYS does not interpret tabs as spaces; consequently errors will occur with text editors that automatically replace spaces by tabs. Completely blank lines are ignored. Lines with '*' in column one will be interpreted as comment lines: printed, but otherwise ignored by TRNSYS Comment lines may be up to 80 characters in length.

In the following sections, long versions of each control statement are supplied. However, the TRNSYS processor only requires the first three characters of each control word and ignores subsequent characters until a space or comma is encountered. Thus a three-letter abbreviation of each control word (SIM, TOL, LIM, etc.) is sufficient.

2.2 SIMULATION CONTROL STATEMENTS

Simulation control statements are used to specify information for a TRNSYS simulation such as its duration and the error tolerances to be used. There are twelve simulation control statements:

1. VERSION
2. SIMULATION
2. TOLERANCES
3. LIMITS
4. CONSTANTS
5. EQUATIONS

6. ACCELERATE
7. LOOP-REPEAT
8. DFQ
9. NOCHECK
10. SOLVER
11. ASSIGN
12. INCLUDE
13. END

Each simulation must have SIMULATION and END statements. The other simulation control statements are optional. Default values are assumed for TOLERANCES, LIMITS, SOLVER and DFQ if they are not present. The format and use of each control statement is described in the next twelve sections. With the exception of the CONSTANTS, EQUATIONS, LOOP-REPEAT and ASSIGN statements, only one statement of each type is allowed in a TRNSYS simulation.

2.2.1 THE VERSION STATEMENT

Newly added with TRNSYS version 15, the VERSION statement has the following syntax

VERSION xx.x

The idea of the command is that by labeling decks with the TRNSYS version number that they were created under, it will be easy to keep TRNSYS more backwards compatible. The version number is saved by the TRNSYS kernal and can be acted upon. For example, if in a later version of TRNSYS, a PARAMETER has been moved to an INPUT, TRNSYS will be able to recognize that the deck being simulated was created with an earlier version and that the PARAMETER has not been moved.

2.2.2 THE SIMULATION STATEMENT

The SIMULATION statement is required for all simulations, and must be placed in the TRNSYS input file prior to the first UNIT-TYPE statement. The simulation statement determines the starting and stopping times of the simulation as well as the timestep to be used. The simulation statement has the form:

SIMULATION t_o t_f Δt

where

t_o is the hour of the year at which the simulation is to begin.
t_f is the hour of the year at which the simulation is to end.

Δt is the timestep to be used* (hours).

The value of t_0 must correspond to the hour of the year at the end of the first time interval for which user data (typically weather data) is supplied. This is particularly important when the TYPE 16 Radiation Processor is used, as the position of the sun will otherwise be incorrectly calculated. For example, the SOLMET data base (1,2) begins each day with integrated radiation data and average meteorological data for the hour from midnight to 1:00 A.M. This interval is designated hour one; the corresponding value of t_0 is 1 (or 1 plus a multiple of 24), and the resulting simulation starting time is 1:00 A.M. To begin a simulation at midnight (t_0 = 0 or a multiple of 24) using SOLMET (or similar) data, it is necessary to add an extra data line to the beginning of the meteorological data file. The data line serves as an initial values line for the TYPE 9 Data Reader. See sections 4.1.1 and 4.1.4 for more details.

Example 1: A system is to be simulated for one year with a timestep of 1/2 hour, using SOLMET Typical Meteorological Year (TMY) data. The simulation is to begin at midnight on Jan. 1, so a duplicate data line is inserted at the beginning of the TMY data and the following SIMULATION statement is used:

```
SIMULATION  0  8760  .5
```

Example 2: A system is to be simulated for the first week of February, beginning and ending at midnight, with a timestep of 1/4 hour. The SIMULATION statement is:

```
SIMULATION  744  912  .25
```

Since midnight on Feb. 1 is the end of the 744th hour of the year, and a week contains 168 hours. Again, an initial values data line (which may be a duplicate line or the last hour of January) is required for the TYPE 9 Data Reader prior to the line that contains meteorological data for the first hour of February.

2.2.3 THE TOLERANCES STATEMENT

The TOLERANCES statement is an optional control statement used to specify the error tolerances* to be used during a TRNSYS simulation. The statement has the form:

TOLERANCES ε_D ε_A

or

TOLERANCES $-\zeta_D$ $-\zeta_A$

---

*Appendix 1.3 contains information which may be helpful in choosing error tolerances.

where

ε_D is a relative (and ζ_D is an absolute) error tolerance controlling the integration error (see Section 1.10.3)

ε_A is a relative (and ζ_A is an absolute) error tolerance controlling the convergence of input and output variables (see Section 1.9)

If a TOLERANCES statement is not present in a TRNSYS input file, the following TOLERANCES are assumed:

$\varepsilon_D = .01$

$\varepsilon_A = .01$

The TOLERANCES statement may appear anywhere in the TRNSYS input file.

Example: The error tolerances ε_D and ε_A are both to be set to one-tenth of one percent. The TOLERANCES statement is then

```
TOLERANCES .001 .001
```

The tolerances for the new equation solver (SOLVER=1) are considered differently. The mean standard deviation for all of the equations is calculated and must be less than the specified tolerances to reach convergence.

2.2.4 THE LIMITS STATEMENT

The LIMITS statement is an optional control statement used to set limits on the number of iterations that will be performed by TRNSYS during a time step before it is determined that the differential equations and/or algebraic equations are not converging.
The LIMITS statement has the form:

```
LIMITS m n p
```

where

m is the maximum number of iterations which can be performed during a time-step before a WARNING message is printed out. (See Section 1.10.3)

n is the maximum number of WARNING messages which may be printed before the simulation terminates in ERROR. (See Section 1.10.3)

p is an optional limit. If any component is called p times in one timestep, then the component will be traced (See Section 2.3.5) for all subsequent calls in the timestep. When p is not specified by the user, TRNSYS sets p equal to m.

If a LIMITS statement is not present in a TRNSYS input file the following LIMITS are assumed:

m = 25; n = 10; p = m

Example: LIMITS of 20 warnings and 50 warning messages are desired for a simulation. The LIMITS statement is then:

LIMITS 20 50

If the algebraic or differential equations in this simulation have not converged within 20 iterations, a WARNING message is printed. All components that have been called 20 times will be traced just before the message is printed. If 50 WARNINGS are printed, then the simulation terminates with an error message.

2.2.5 THE CONSTANTS STATEMENT

The CONSTANTS statement is useful when simulating a number of systems with identical component configurations but with different parameter values, initial input values, or initial values of time dependent variables. The EQUATIONS statement (see Section 2.2.6) is a more powerful version of the CONSTANTS statement, however CONSTANTS has been retained for use with the TRNSED utility.

The CONSTANTS statement has the format

CONSTANTS n
$NAME_1$ = $value_1$
$NAME_{i+1}$ = $value_{i+1}$
$NAME_n$ = $value_n$

where $NAME_i$ and $value_i$ are defined as follows:

- $NAME_i$ is a character string consisting of a letter followed by up to seven additional letters or numbers. If more than eight letters are used, the ninth and succeeding letters are ignored. Thus A2, AIZ and BOY are valid constant names. The string TEMPERATURE is treated as if TEMPERAT was supplied.

- $value_i$ is either a numerical value or a simple arithmetic expression. Values may be numbers, constants that have already been defined, or simple expressions. The CONSTANTS processor recognizes simple expressions built from numbers and previously defined constants using addition, subtraction, multiplication, and division. An example is

```
CONSTANTS 2
A = 10
FLW = A * 50
```

The simple expressions are processed much as FORTRAN arithmetic statements are, with three significant exceptions:

(1) Blanks must be placed on both sides of the operation codes +, -, *, and /.
(2) Expressions are evaluated from left to right with no precedence accorded to any operation over another.
(3) Parentheses are not allowed.

Rules (1) and (3) are simple matters of syntax and TRNSYS usually catches any violation. Rule (2), however, must constantly be borne in mind when writing long expressions. Note the value assigned to C in the following examples:

INPUT	EFFECT
CONSTANTS 3	
A = 1	
B = 2	
C = A * B + 2	A = 1 B = 2 C = 4
CONSTANTS 3	
A = 1	
B = A +1	
C = A + B * 2	A = 1 B = 2 C = 6

As many lines as desired, including the line with the word CONSTANTS, may be used to list the n names and values. In the succeeding input lines of the TRNSYS input file, the constant names may appear anywhere that a number is required. When found, the names are replaced by the values defined by the CONSTANTS statement. Note that constant names may not appear in lieu of numerical data read by the TYPE 9 data reader or the subroutine DATA. There is an upper limit of n variables defined by CONSTANT and EQUATION (see Section 2.2.5) statements per simulation, where n is set in the param.inc file. The limit of n CONSTANTS plus EQUATIONS may be changed by experienced TRNSYS users - refer to reference section 6.5 for more details.

Multiple statements of this form may be used, each beginning with the word CONSTANTS.

2.2.6 THE EQUATIONS STATEMENT

The EQUATIONS statement (abbreviated as either EQU or EQN) allows variables to be

defined as algebraic functions of constants, previously defined variables, and outputs from TRNSYS components. These variables can then be used in place of numbers in the TRNSYS input file to represent inputs to components; numerical values of parameters; and initial values of inputs and time-dependent variables. The capabilities of the EQUATIONS statement overlap but greatly exceed those of the CONSTANTS statement described in the previous section.

The EQUATIONS statement has the following form:

```
EQUATIONS n
NAME1 = ... equation 1 ...
NAME2 = ... equation 2 ...
  •
  •
  •
NAMEn = ... equation n ...
```

The capabilities and rules of the EQUATIONS processor are summarized below:

1. Up to n (see Sect. 6.5) variable names may be defined by the EQUATIONS and CONSTANTS statements.
2. There may be as many EQUATIONS statements as needed and they may be placed anywhere in the TRNSYS input file before the END statement.
3. Each equation must be on a separate 160-column line. Long equations can be accommodated by defining intermediate variables and referring to these variables in following equations.
4. The variable name to be defined must begin with a letter and may consist of 1 to 10 significant alphanumeric characters. Characters after the 10th are ignored. Upper or lower case letters may be used in the input file, but all letters are internally converted to upper case. The variable name must not have been previously defined by either preceding EQUATIONS or by a CONSTANTS statement. Certain names are reserved for built-in functions, such as TIME, sin, max, etc. A list of the available functions appears in item 9.
5. Spaces may appear anywhere in the equation for readability, except within variable names.
6. The variable being defined must be followed by an equal sign. Only one equal sign may appear in an equation.
7. The algebraic expression to the right of the equal sign may involve numerical values, variable names previously defined with EQUATIONS or CONSTANTS statements or outputs from TRNSYS components. TRNSYS outputs are referenced by placing the unit and output numbers in square brackets separated by a comma or space. Thus [5,3] refers to the third output of UNIT 5.
8. Equations are formulated using exactly the same rules as used in FORTRAN. Parentheses may be used as needed. Either ** or ^ may be used to denote raising to a power.
9. The EQUATIONS processor recognizes the following functions. These function names are reserved and may not be used as variable names.

   ABS( ) absolute value of the expression in parenthesis.

   ACOS( ) arc cosine of the expression in parentheses, returned in degrees.

AND(,)	returns Boolean AND value of expressions (separated by a comma) in parentheses.
ASIN( )	arc sine of the expression in parentheses, returned in degrees.
ATAN( )	arc tangent of the expression in parentheses, returned in degrees.
COS( )	cosine of the expression in degrees enclosed in the parentheses.
EQL(,)	returns 1 if first expression is equal to second; returns 0 otherwise. (Expressions separated by a comma.)
EXP( )	exponential of the expression enclosed in parentheses.
GT(,)	returns 1 if first expression in parentheses is greater than second; returns 0 otherwise. (Expressions separated by a comma.)
INT( )	integer value of the expression in parentheses. This function will truncate the real value. To round up, add 0.5 to the expression.
OR(,)	returns Boolean OR value of expressions (seperated by a comma) in parentheses.
LN( )	base e logarithm of the expression enclosed in parentheses.
LOG( )	base 10 logarithm of the expression enclosed in parentheses.
LT(,)	returns 1 if first expression in parentheses is less than second; returns 0 otherwise. (Expressions separated by a comma.)
MAX(,)	maximum of the two expressions, separated by a comma, in the parentheses.
MIN(,)	minimum of the two expressions, separated by a comma, in the parentheses.
MOD(,)	modulus, i.e., the remainder of the division of the first expression by the second expression. A repeating daily function can be generated by taking the modulus of time with a 24 hour period, e.g., REPEAT = 7+MOD(TIME,24).
NOT( )	returns Boolean NOT value of expression in parentheses.
SIN( )	sine of the expression in degrees enclosed in the parentheses.
TAN( )	tangent of the expression in degrees enclosed in the parentheses.

10. The following variables have predefined values and may not be redefined:

CONST	indicates 'constant' and may be used interchangeably with 0,0 as the unit and output numbers specifying a constant INPUT. (See section 2.3.3).
TIME	current time in the simulation.

11. Variable names defined by an EQUATIONS or CONSTANTS statement may be used in place of numerical values or the unit number, output number combinations which follow the INPUTS statement. Variables used as INPUTS are evaluated each time one of their constituent quantities changes. Variables used in place of numerical values for parameters, or initial values of inputs and time-dependent quantities are evaluated once at the start of the simulation and therefore should not refer to TIME or to component outputs. Important: Equations that vary with time should not be used as initial values or as parameters since the equation will be calculated only once at the beginning of the simulation.

The following example, illustrates some of the capabilities of the EQUATIONS processor :

```
EQUATIONS 8
PI = 3.1415
twopi = 2*PI
abc = 22.5*(1.005 + 4.19)
TEMP0 = 20
FLOW = [2,2] + [4,2]
TIME2 = MOD(TIME,24)
Load = [20,5]
Fract = MAX (0,(1-[20,4]/Load))

UNIT 17 TYPE 34
PARAMETERS 3
TwoPi   32.3  ABC
INPUTS 5
2,7     FLOW    TIME2       Fract       0,0
0.0     15.0    22.5        abc         15.5
DERIVATIVES 1
TEMP0
```

2.2.7 THE ACCELERATE STATEMENT

The ACCELERATE statement directs TRNSYS to use a convergence promoting numerical method to converge upon the values of specified outputs. The ACCELERATE statement should not be used with the new TRNSYS equation solver (see Section 2.2.11). The accelerate statement was retained for backwards compatibility reasons only.

The format of the statement is:

```
ACCELERATE n
  u1,o1   u2,o2  ...  ui,oi .... un,on
```

where

- n is the number of outputs which are to undergo convergence promotion ($n \leq 50$)
- u_i is the UNIT number of the output which is to undergo convergence promotion.
- o_i is the OUTPUT number (i.e. 1, 2, etc.) of UNIT u_i which is to undergo convergence promotion.

Only 1 ACCELERATE statement is allowed: A maximum of 50 outputs to be accelerated may be specified. The ACCELERATE statement may be placed anywhere between the

SIMULATION and END statements in the TRNSYS input file.

The purpose of the ACCELERATE statement is to reduce the number of iterations required to achieve convergence during each timestep of the simulation. Its use is particularly effective in simulations that have recyclic information loops (see Section 1.9). Figure 2.2.7.1 illustrates a recyclic information loop containing k components. The output from component k that results from an input value x to the first component is termed f(x). Upon convergence, x equals f(x) within the algebraic error tolerance specified on the TOLERANCES control statement. Figure 2.2.6.2 shows an example of the use of successive substitution for this situation. The iteration process begins with an initial guess, x_1, as an input to the first component. After proceeding around the information loop, an output $f(x_1)$ is provided by component k. This output is then used as the second guess, x_2, for the input to the first component. The solution occurs at the intersection of the curve of f(x) versus x and a 45° line (f(x)=x).

Convergence of this system could be improved by using the ACCELERATE statement for the output of component 1 (or any other of the outputs in the information loop). In this case, TRNSYS will use a secant method to find the solution for f(x)=x. After two iterations with successive substitution, the most recent values of x and f(x)-x are used to estimate a straight line solution as illustrated in Figure 2.2.7.3. Previous estimates of x_1 and x_2 result in a new guess of x_3. For the example illustrated in Figure 2.2.7.3, x_2 and x_3 are then used to arrive very nearly at the solution. If a new estimate of x is further from the solution than the previous estimate (as indicated by the values of f(x)-x), then the secant method is abandoned and successive substitution is again applied for the next two iterations. In this way, the algorithm protects against erroneous solutions due to changing control functions.

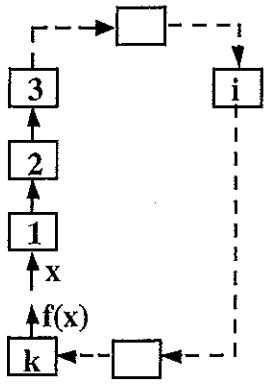

Figure 2.2.7.1: Recyclic Information Flow

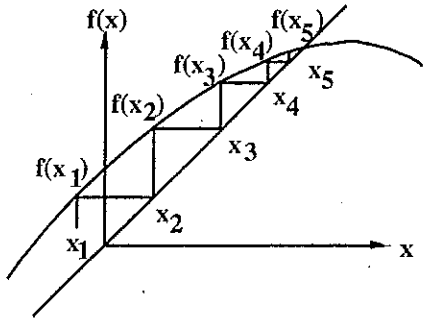

Figure 2.2.7.2: Convergence by Successive Substitution

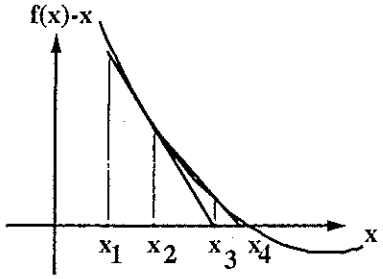

Figure 2.2.7.3: Convergence by Secant Method

The ACCELERATE statement should not be used to calculate the new values of control signals due to the on/off nature of controllers.

2.2.8 THE LOOP-REPEAT STATEMENT AND CALCULATION ORDER

The LOOP statement allows the user to have some control over the calling order of the components during the simulation. The LOOP statement is not required and should not be used with the new TRNSYS solver (SOLVER 1) described in section 2.2.11. The LOOP command was retained for backwards compatibility.

The format of the LOOP statement is

LOOP n REPEAT y

u_1 $u_2 \ldots u_n$

where

n is the number of units which are to be called in a loop

y is the number of times these units will be called (if convergence is not attained)

u_i is the UNIT number for unit i

Up to 10 LOOP statements may appear in the TRNSYS input file and they may be placed anywhere after the SIMULATION statement and before the END statement. A grand total of 250 units may be listed on the 10 statements.

TRNSYS iteratively calls only the component subroutines whose INPUTS have not converged or whose OUTPUTS depend explicitly upon simulation time, as indicated either by the existence of derivatives (INFO(5)>0) or other time function (INFO(9)=1) (See Section 3.3.3). If a LOOP statement does not appear in the TRNSYS input file, the order in which the units are called is the same as the order in which they appear in the TRNSYS input file with the following exceptions:

- All TYPES with INFO(5) ≠ 0 or INFO(9) = 1 are pulled out of the order and called first.
- The output producing components (TYPES 24 through 29 and those with INFO(9)=2 or INFO(9)=3 (see Section 3.3.3)) are pulled out of the order and called last, after convergence has been attained.

The calculation order may be changed to some extent by simply rearranging the order of the components in the input file. The most efficient calculation scheme appears to be one in which the order is the same as the direction of information flow. Although the order in which the units are called should not affect the results of the simulation, it may affect the amount of calculation required. TRNSYS summarizes the number of times each UNIT was called at the end of the simulation output.

In some simulations, particularly those with multiple recyclic information flow loops, it may be advantageous to call some units several times, thereby allowing them to partially converge upon a local solution, before remaining units in the simulation input file are called. Control of this type is possible with the LOOP statement. The LOOP statement indicates to TRNSYS that the n specified units are to be called after the UNIT preceding the LOOP statement in the input file for up to y times.

The operation of the LOOP statement is best explained by a simple example: Consider a TRNSYS input file containing the following UNIT and LOOP statements in the indicated order.

```
UNIT 1 TYPE 12
.
.
.
UNIT 5 TYPE 13
.
.
.
UNIT 2 TYPE 14
.
```

```
.
.
LOOP 3 REPEAT 2
  1 3 2

UNIT 3 TYPE 36
.
.
.
UNIT 7 TYPE 37
.
.
.
```

Without the LOOP statement, TRNSYS checks the inputs to, and if necessary calls, units 1, 5, 2, 3, 7 repeatedly in this order until convergence within the specified tolerance is attained. With the LOOP statement, the calling order during each iteration is: 1, 5, 2, 1, 3, 2, 1, 3, 2, 3, 7. Although the calling order is changed, TRNSYS will only call a unit if its inputs do not match within tolerance with values from the previous call.

2.2.9 **THE DFQ STATEMENT**

The optional DFQ card allows the user to select one of three algorithms built into TRNSYS to numerically solve differential equations (see Section 1.10.2).

The format of the DFQ card is

```
DFQ k
```

where k is an integer between 1 and 3. If a DFQ card is not present in the TRNSYS input file, DFQ 1 is assumed.

The three numerical integration algorithms are:

1. Modified-Euler method (a 2nd order Runge-Kutta method) (1)
2. Non-self-starting Heun's method (a 2nd order Predictor-Corrector method) (2)
3. Fourth-order Adams method (a 4th order Predictor-Corrector method) (3)

Descriptions of these algorithms can be found in Chapra and Canal (see references) and most other numerical methods books.

All three algorithms have been chosen to allow TRNSYS to simultaneously solve the algebraic and differential equations comprising the system model each timestep. Although it would appear that higher order methods, such as the 4th order predictor-corrector method, would result in improved accuracy and reduced computation, this is not generally the case in TRNSYS simulations because of the need to simultaneously solve algebraic and differential equations.

Experience has shown that Heun's method usually is most efficient, however the modified Euler method is most consistent with the analytical method of solving differential equations used in many components (see Section 1.10.1).

2.2.10 THE NOCHECK STATEMENT

TRNSYS allows up to 20 different INPUTS to be removed from the list of INPUTS to be checked for convergence (see Section 1.9). This is done using the NOCHECK statement, which has the following format:

NOCHECK n
$u_1,i_1 \quad u_2,i_2 \ldots u_i,i_i \ldots u_n,i_n$

where

n is the number of inputs which are to bypass the convergence checking ($n \leq 20$).
u_i is the UNIT number of the input which is to be bypassed in convergence checking.
i_i is the INPUT number (i.e. 1, 2, etc.) of UNIT u_i which is to be bypassed in convergence checking.

Only 1 NOCHECK statement is allowed and it may be placed anywhere before the END statement.

The NOCHECK statement replaces the convergence check inhibiting feature documented in earlier versions of TRNSYS. Placing a negative sign before a unit, output pair in an input specification will not inhibit convergence checking (and will, in fact, cause an error). The NOCHECK statement must be used instead.

This NOCHECK statement should be used with extreme caution since it inhibits the fundamental error checking functions of the TRNSYS processor and can result in unpredictable and incorrect results if it is applied without a careful analysis of the system's information flow. The following discussion concerns a situation in which this option could be applied with good results.

Certain types of simulations require that components have as inputs both temperatures and energy flows. An example is a house space heating system in which tank or rockbed energy losses are added to the heated space. The house model (TYPE 12 or TYPE 19) would also require as inputs the temperature and mass flow rate of fluid out of the storage unit. Since TRNSYS applies the algebraic convergence test to INPUTS of components, in this case fluid temperature, mass flow rate, and heat losses which are input to the room model will be checked for convergence. If relative convergence test using default tolerances is applied to all inputs, and, for example, the storage unit temperature is around 100 degrees, a variation of less than one degree in storage unit outlet temperature will satisfy the convergence test, since this is less than 1% of the magnitude of the input value. Storage losses may be on the order of 3000 and will have to vary by less than 30 from one call to the next for convergence to be obtained. If the

storage fluid flow rate is high, a variation of 1 degree in house inlet temperature may represent a large change in delivered energy. Under these circumstances, default error tolerances will probably result in poor energy balances.

The solution to the energy balance problem is to use absolute error tolerances in the simulation. This could be done by inserting a "TOLERANCES -0.1, -0.1" statement into the simulation input file. With these specified tolerances, the INPUTS to units will have to change by less than 0.1 in magnitude before TRNSYS assumes that algebraic convergence has been obtained. This is a reasonable requirement for the storage fluid temperature input to the house, since it is on the order of 100. However, the storage unit energy losses are much larger and are sensitive to changes in house and storage unit temperature. As a result, iteration may continue far beyond the point required for convergence of the linked temperatures due to the sensitivity of storage losses to slight variations in system temperatures. A situation like this will result in a slow and expensive simulation run.

This leaves a TRNSYS user with two equally undesirable alternatives. The first is to use default (relative) error tolerances and have poor energy balances. The second is to exclude storage unit energy losses from the house model and use absolute error tolerances. Neither alternative will give an accurate picture of system thermal performance. A way around this problem is to use absolute error tolerances to get good energy balances and to prevent TRNSYS from checking the tank loss input of the house model for convergence using the NOCHECK statement.

The NOCHECK statement takes on additional importance when the new TRNSYS solver is employed; especially when backsolving is to be used. When using the new TRNSYS solver, control signals and other specific TRNSYS input types should not be checked. The solver section (2.2.10) provides more information.

2.2.11 THE SOLVER STATEMENT

Although relatively simple, the successive substitution computational scheme used in past versions of TRNSYS has proven to be reliable and efficient for simulating systems with energy storage such as solar domestic hot water systems. These systems typically have less than 50 coupled differential equations and 100 simultaneous nearly-linear algebraic equations with few recyclic loops and controller decisions. The limitations of the computational scheme become apparent when TRNSYS is used to solve sets of non-linear algebraic equations without differential equations. Equations of this type occur in systems for which the energy storage is negligible, such as for a photovoltaic array directly coupled to a load or a refrigeration system operating at steady-state conditions. The successive substitution solution method used in previous versions of TRNSYS does not efficiently solve non-linear algebraic equations and may, in fact, not be able to find a solution if the equations are highly non-linear. Previous versions of TRNSYS had no formal means of supplying suitable guess values of the INPUTS at each timestep and had no provision to bound the values it was trying to determine.

An ACCELERATE command was added to version 13 to improve convergence in problems with recyclic information flow. The ACCELERATE (see Section 2.2.7) command allows the user to break a selected INPUT-OUTPUT connection and replace it with a single-variable Newton's method solution algorithm. Although useful in many circumstances, the ACCELERATE command may be unsatisfactory for two reasons; 1) it requires the user to identify the appropriate INPUT-OUTPUT connection and 2) it implements a single-variable solution method when in many situations, a multiple-variable method is required. Another way in which numerical convergence problems have been handled in past versions is by the user coding convergence-enhancing techniques within the component models. For example, 'combined-component' models have been developed for TRNSYS wherein the non-linear equations describing two or more pieces of equipment are solved internally in a single component model. Although 'combined-component' models may eliminate numerical problems, they reduce component modularity and require the user to implement solution techniques, thereby defeating the original purpose of TRNSYS.

The distinction between INPUTS and OUTPUTS in TRNSYS places a limitation on the generality of the component models. When a model is formulated, a decision must be made as to what are the INPUTS and what are the OUTPUTS. For example, a model may be developed to calculate an energy flow for a given mass flowrate and temperatures. However, in a different application, the energy flowrate may be known and the model must calculate the mass flowrate. The same physical model is used in either situation, but the INPUTS and OUTPUTS are different. Previous TRNSYS versions required different component models (or modes) to handle the change in information flow. The alternative solver, when implemented, is able to backsolve the TRNSYS equations; effectively solving an input for a given output.

The limitation of the past TRNSYS computational scheme provided an incentive to significantly revise the executive routines of this program. A large library of component models has been developed for TRNSYS. A major consideration in this revision has been to maintain compatibility with the existing component library and therefore with the INPUT -OUTPUT-PARAMETER information flow classification.

The equations within any TRNSYS component model can reduced to the following general form:

$$\dot{X}_i = D_i(X_i, \dot{X}_i, U_i, t) \qquad (2.2.11.1)$$

$$X_i = F_i(U_i, t) \qquad (2.2.11.2)$$

where U_i and X_i are, respectively, the vector of INPUTS and OUTPUTS for the ith module, t is time, and D_i and F_i are functions representing the differential and algebraic equations, respectively.

An analogous set of equations can be written for the combined set of equations in all components,

$$\dot{X} = D(X, \dot{X}, U, t) \quad (2.2.11.3)$$

$$X = F(U, t) \quad (2.2.11.4)$$

where U and X are vectors containing the INPUTS and OUTPUTS for all components. Each INPUT in a TRNSYS model is an OUTPUT from another component, an equation, or a specified value. This additional information can be written as a matrix equation

$$AU + BX + D = 0 \quad (2.2.11.5)$$

where **A** and **B** are coupling matrixes with element values of 0 and ±1, and D is a vector of boundary conditions. The system of algebraic-differential equations (2.2.11.3–5) can now be solved to determine U(t) and X(t). This computational scheme is similar to that used in the IDA modular system simulation program (5).

The number of simultaneous non-linear equations resulting from this new computational scheme can become very large. To minimize computational effort, it is necessary to employ methods for decreasing the number of simultaneous equations. Some TRNSYS inputs are always known, either because they are constants or they depend explicitly on time. Moreover, it is possible for all of the inputs in some TRNSYS components to be of this type. In this case, the outputs of these components can be calculated independently of other components. Once the outputs of these components are known, the inputs to which they are connected are now known, possibly allowing additional components to be evaluated independently. This process is repeated until only the components that require simultaneous solution remain. This process is a first step in equation blocking in which a large set of equations is broken into smaller sets that are more easily solved.

The remaining set of equations (2.2.11.3-5) is solved numerically. There are a number of well-developed numerical methods for these types of systems (6,7). In TRNSYS 14 and beyond, differential equations are solved by a backwards differential method (8) so that the combined algebraic and differential systems can be converted to an algebraic system which can be solved at each time step. This method is extremely robust, even for stiff problems. In addition, this method allows use of variable time steps, although TRNSYS does not currently employ variable time steps. Before solving the resulting system of algebraic equations, TRNSYS numerically calculates the Jacobian matrix and permutes it to lower triangular form. This permutation facilitates equation blocking which breaks the original system into smaller sets of equations that can be solved more efficiently (9). Each block of equations is solved using Powell's method (10,11). This method combines Newton's method and steepest descent approach and is one of the more robust and efficient algorithms for solving non-linear equations (Example #14).

A major advantage of the computational scheme is that the solution method does not care whether INPUTS or OUTPUTS are specified as long as a valid set of simultaneous equations are defined. As a result, TRNSYS has the ability to solve backward problems, i.e problems in which

one or more OUTPUTS are specified and the corresponding INPUTS must be determined. An example of a backward solution problem is provided in the Examples section.

Unfortunately, there is also a disadvantage of the new computational scheme. Because the alternative solver determines the Jacobian matrix numerically, TRNSYS usually requires more component calls each time step than the scheme used in TRNSYS 13 - provided that TRNSYS 13 is able to solve the problem. The computational scheme in TRNSYS 14 and above is far more robust and is consequently able to solve a greater variety of problems without experiencing computational difficulties. However, because some problems can be more efficiently solved with the older computational scheme, both computational schemes are made available to the user in TRNSYS version 14 and above. In most cases, users should try to solve their TRNSYS input file using the simpler (and most times quicker) successive substitution method. If problems are encountered, the alternative solver should be employed.

A SOLVER command has been added to TRNSYS to select the computational scheme. The optional SOLVER card allows the user to select one of two algorithms built into TRNSYS to numerically solve the system of algebraic and differential equations. The format of the SOLVER card is

SOLVER k

where k is either the integer 0 or 1. If a SOLVER card is not present in the TRNSYS input file, SOLVER 0 is assumed.

The two solution algorithms are:

0. Successive Substitution (as in previous versions of TRNSYS)
1. Powell's Method (11)

Descriptions of these algorithms can be found in the references and most other numerical methods books.

A problem that plagued TRNSYS 13 and other general equation solvers in the past is the presence of discontinuities or discrete states, particularly when they demonstrate hysteresis properties. Many existing TRNSYS components have equations of this type. The most common example is the ON/OFF Differential Controller provided as TYPE 2 in the standard TRNSYS library. This component calculates a control signal, which is either 0 or 1, based on the differences between two input temperatures and the control signal from the previous calculation. The functional relation between the controller OUTPUT and its INPUTS is shown in Figure 2.2.11.1. If the temperature difference is less than ΔT_{min}, the control signal is 0. If the temperature difference is greater than ΔT_{max}, the control signal is 1. However, if the temperature difference is between ΔT_{min} and ΔT_{max}, the output control signal is either 0 or 1, depending on its previous value. This type of mathematical behavior can lead to oscillations in the calculations preventing convergence of the numerical solution to the system of equations. In fact, there may

not be a stable numerical solution, depending on the selected values of ΔT_{min} and ΔT_{max}.

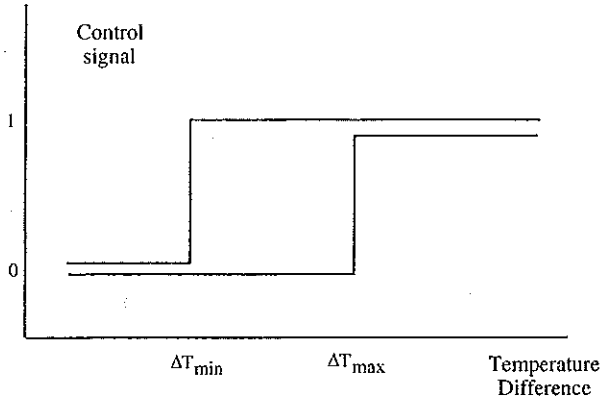

Fig. 2.2.11.1: TRNSYS Type 2 controller with hysteresis properties

There are many examples of ON/OFF controls in the TRNSYS component library, such as the Flow Diverter (Type 11), the Energy/Degree-Day Space Heating or Cooling Load (Type 12) and the Pressure Relief Valve (Type 13). Other examples of discontinuities which appear in TRNSYS component models are the shift from turbulent to laminar flow regime which causes a discrete change in a heat transfer coefficient or friction factor, and the inter-node flow control logic for a stratified water storage tank. During the iteration process the component OUTPUTS can switch states resulting, in effect, in a different set of equations with a different Jacobian. Previous versions of TRNSYS handled the convergence problems that result from this behavior by 'sticking' the state after a specified number of iterations. For example, the TYPE 2 ON/OFF Differential Controller has a parameters called NSTK which is the number of state changes allowed before the controller state is frozen at its current value. Although freezing the controller state eliminates most convergence problems, it may lead to incorrect results in some cases, particularly in short term simulations.

The alternative solver in TRNSYS 14 provided a very different approach for handling these problematic discontinuities. The TRNSYS executive program, rather than the component models, directly controls discrete variables. At the start of each time step, the values of all discrete variables are known. TRNSYS forces these values to remain at their current state until a converged solution to the equations is obtained or an iteration limit is encountered. During these calculations, the component models calculate the 'desired' value of the discrete variables but they do not actually change the settings. After a converged solution is obtained, or the maximum number of iterations is exceeded, TRNSYS compares the current discrete variable values with the 'desired' values. If they do not differ and a converged solution has been obtained, the calculations are completed for the time step. Otherwise, TRNSYS changes the values of the discrete variables to the 'desired' setting and repeats this process, taking care to not repeat the calculations with the same set of discrete variables used previously. If a solution is not obtained after all combinations of discrete variables have been tried, TRNSYS issues a warning message and continues on to the next time step. This method of dealing with discrete variables eliminates numerical oscillations between iterations and finds the correct solution to the equations, provided that a solution truly exists. Minor changes in the component models utilizing discrete variables

are required to take advantage of this computational scheme. These changes have been made in many of the TRNSYS library components. For these reasons, the new TRNSYS controller should be used when the new TRNSYS equation solver (SOLVER=1) is employed. Refer to the TYPE 2 documentation (section 4.4.1) for more details.

To use the backsolving technique described in this section, follow the guidelines listed below:

1) the new TRNSYS equation solver should be used (SOLVER 1)
2) the new control mode should be used (TYPE 2 - MODE 0)
3) initial values should be reasonable and non-zero
4) the input to be solved should be specified as -1,0 in the input file with an initial value equal to the value the input will take if no backwards solution is found.
5) the specified OUTPUT should be set in an equation with the form:
   EQUATION 1
   [5,3] = 65 (set the third output from UNIT 5 to a constant value of 65)
6) the specified OUTPUT can be of any form acceptable to the TRNSYS equation processor ([5,3] = 65 + MOD(TIME) + 0.5 * [1,1] for example)
7) the NOCHECK statement should be used with the following variables:
   - all INPUTS to the TYPE 2 controllers (required to eliminate convergence problems)
   - all INPUTS which do not need to be checked for convergence at each iteration- for example flow rates at all components in a flow loop (recommended to reduce the set of equations to be solved)
8) set the differential equation solver to Modified-Euler method (DFQ 1)

Although the backsolving technique is extremely useful, in many cases it can not be employed due to the formulation of many component routines prior to version 14. Due to internal convergence enhancement algorithms, all users writing component routines should consider the backsolving technique during component formulation.

The following examples illustrate the capabilities of the backsolving technique and of the new equation solver and provide valuable user information. The backsolving example located in the example section of the TRNSYS manual contains additional information.

Example: Solar Water Heating System

The first example describes a typical solar domestic hot water system with a collector-tank heat exchanger; shown in Figure 2.2.11.2. When the solar radiation is available, a fluid is circulated through the solar collector at a constant mass flowrate. The collected energy is transferred through a heat exchanger to a storage tank. 100 kg of water constantly flows into the bottom of the storage tank. Hot water removed from the upper part of the tank is mixed with cold supply water in the mixing tee-piece at a mass flowrate sufficient to maintain the temperature of the water going to the load below a specified maximum.

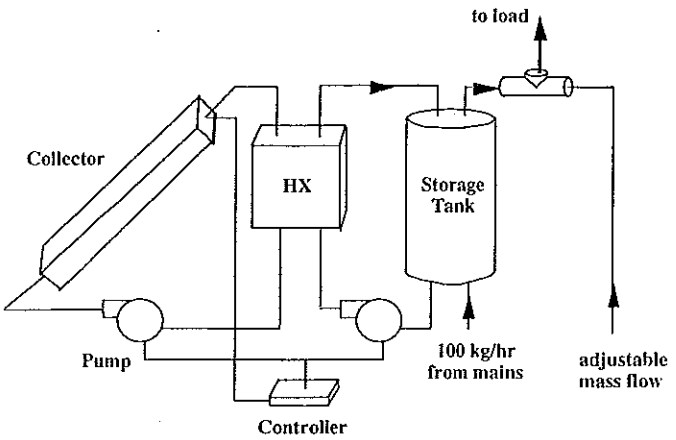

Fig. 2.2.11.2: Simple water heating system with variable water flowrate

All of the components required to simulate this system exist in the standard TRNSYS library. However, the tee-piece component model has been developed to calculate the outlet temperature and mass flowrate given known mass flowrates and temperature for the inlet streams. In this example, it is necessary to calculate the mass flowrate of the mains supply stream in order to control the water outlet temperature. This is an example of a 'backwards' problem in which an OUTPUT is known and it is necessary to calculate an INPUT. This problem could not easily be solved with the existing tee-piece model in the TRNSYS library with previous versions of TRNSYS. Results of the simulation with TRNSYS 14 for a day in August in Sacramento, CA are shown in Figure 2.2.11.3 for controlled water outlet temperatures of 45°C.

The ability provided in TRNSYS to solve backwards problems is very convenient since it does not require any modification to the existing components. In many cases, however, backward problems of this type do not have a solution or they may have a solution for only certain hours of the day. In the system shown in Figure 2.2.11.2, for example, it is possible to keep the outlet water temperature above 45°C only between 8:00 am and 6:00 pm, as seen in the simulation results in Figure 2.2.10.3. A solution to the backwards problem does not exist during the remainder of the day. If TRNSYS 14 can not find a solution to the backward problem, it will issue a warning message and proceed to solve the forward problem in which the INPUTS are known. In this case the values of the unknown INPUTS are set to their initial values and the OUTPUTS are calculated corresponding to these inputs values. The result of this computational scheme is evident in Figure 2.2.11.3 in which the mains water flowrate through the tee-piece was set to 30 kg/hr outside of the interval between 8:00 am and 6:00 pm resulting in a total mass flowrate of 130 kg/hr.

A portion of the TRNSYS input for the above example is given to demonstrate the required backsolving formulation:

```
SOLVER 1
SIMULATION 0 24 0.25
```

```
NOCHECK 4
2,1   2,2   2,3   2,4

UNIT 2 TYPE 2 CONTROLLER
PARAMETERS 5
*MODE  UDB  LDB  TMAX  TRESET
0  5  1  100  95
INPUTS 4
1,1  4,1  4,3  2,1
75.  50.  50.  0.
* Use the new control strategy (MODE 0) since backsolving is employed
* Provide reasonable guess values for initial values

* SET THE OUTLET TEMPERATURE TO A CONSTANT 45 C
EQUATIONS 1
[11,1] = 45
* SET COLD FLOW TO 30 IF NO SOLUTION EXISTS
* OUTPUTS FROM TYPE 11 ARE THE TEMPERATURE AND MASS FLOW RATE

UNIT 11 TYPE 11 TEE PIECE
PARAMETERS 1
* MODE 1 - TEE PIECE
1
INPUTS  4
* TTANK  MTANK  TCOLD    MCOLD
   4,3     4,4    TMAINS   -1,0
   70.     100.   20.0     30.
* SET COLD FLOW TO 30 IF NO SOLUTION
*OUTPUTS ARE TEMP AND MASS FLOW
```

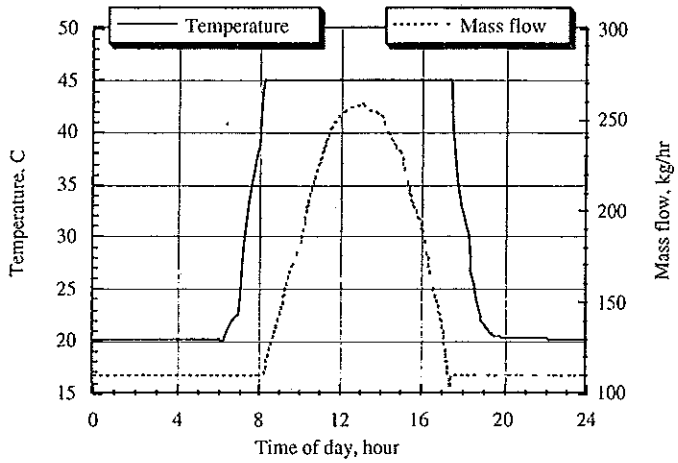

Fig. 2.2.11.3: Calculated flow rate and temperature for the system shown in Fig. 2.2.11.2.

Example: Photovoltaic System

A photovoltaic array directly coupled to a resistance load results in a rather difficult problem for TRNSYS because the system has no storage capacity and therefore no differential equations; resulting in a set of highly coupled non-linear algebraic equations. In addition, the algebraic equations are highly non-linear. The photovoltaic array is modeled using Eqns 23.2.6 - 23.2.8 from Duffie and Beckman (12). Current-voltage plots for the PV array at several radiation levels are shown in Figure 2.2.11.4. Also shown in Figure 2.2.11.4 are the straight-line current-voltage relations for resistance loads.

At a specified time, and therefore at known solar radiation and ambient temperature, photovoltaic array current-voltage characteristics are known. It is now necessary to find the current and voltage at the intersection point of the load and array curves. At high insolation and/or loads, this algebraic system becomes very nonlinear, as evident in Figure 2.2.11.4. Photovoltaic systems have been simulated previously in TRNSYS 13 (13). However, in earlier studies, it has been necessary to add convergence-enhancing code to the component models, or to combine the array and load models into a single component. In this example, however, the photovoltaic array and the load are each modeled as separate components using only the governing current-voltage equations. The results of a simulation for a 24 hour period with 0.1 hour time steps on June 1 in Madison, WI are shown in Figure 2.2.10.5 using both TRNSYS 13 and TRNSYS 14.

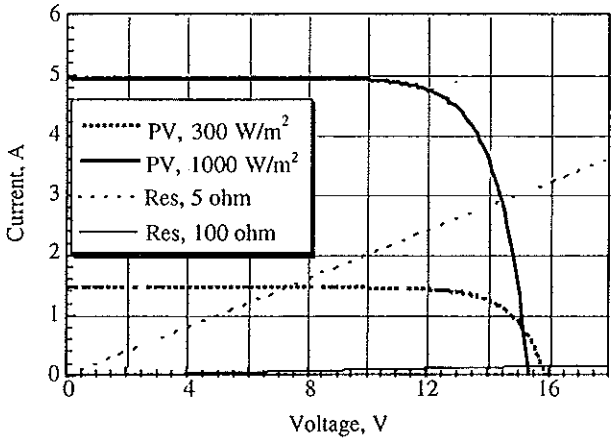

Fig. 2.2.11.4: Current-voltage curves for PV array and the resistance load.

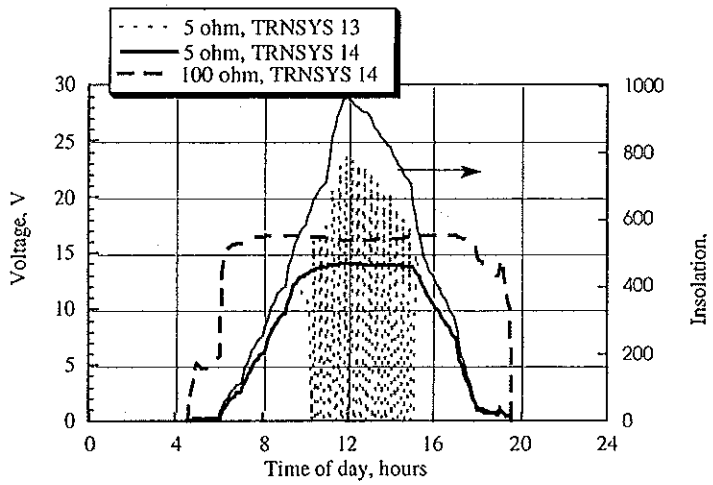

Fig. 2.2.11.5: The dependence of PV array voltage on time for different loads.

As the insolation increases, TRNSYS 13 is unable to solve this problem even for small loads. The calculated performance is unstable, oscillating above and below the true solution every other time step. TRNSYS version 14 and beyond, however, experience no difficulties in solving this problem for a wide range of insolation, ambient temperature, and loads. It is this type of problem for which the alternative equation solver in TRNSYS 14 was designed.

2.2.12 THE ASSIGN STATEMENT AND LOGICAL UNIT NUMBERS

The ASSIGN statement allows the assignment of files to logical unit numbers from within the TRNSYS input file. The format for this statement is

```
ASSIGN  filename  lu
```

where

filename	is the full name of the desired file (including path, if necessary); filename must be less than or equal to 160 characters in length. Spaces are allowed in pathnames as long as the entire path is contained in quotes. Quote marks are not necessary if there are no spaces in the pathname.
lu	is the logical unit number to which filename is to be assigned

As many ASSIGN statements as allowed in Windows may be used in a TRNSYS input file and they may be placed anywhere before the END statement. The general TRNSYS output should be directed to logical unit 6 for every simulation. The ASSIGN statement assigning logical unit number 6 to a file should be placed before any other control statements (otherwise the input file listing and error messages may be incomplete). If the logical unit 6 is ASSIGNED at the beginning of the file, some part of the listing may be written to a file called TRNSYS.OUT which resides in the \trnsys15 directory.

TRNSYS uses several logical unit numbers[*]:

LU	device or file
4	TRNSYS input file
5	keyboard
6	listing of input file and error messages (normally written to screen unless otherwise assigned)
7	UNITS.LAB
8	ASHRAE.COF (for use with TYPE 19 only)
9	CNFGTR.TRN

Additionally, the user must specify logical unit numbers when configuring output components, TYPE 9, and components that need to read data files. The re-use of the above listed logical unit numbers should be avoided at all costs. In general, weather and output files should be directed to LUs greater than 10.

For better compatibility with the TRNSYS utility program such as TRNSHELL and TRNSED, the following convention should be used when ASSIGNing the listing, output, and plot files within the input file (.dck). The listing, output, and plot files should all have the same name as the input file. For example if the input file is called TEST.DCK and located in the \trnsys15\ test directory, then the following ASSIGN statements should be used and placed within the input file:

---

[*]The input deck, keyboard, deck and error output logical unit numbers can be reassigned in the BLOCKDATA subroutine. The ASHRAE.COF logical unit number can be reassigned in the TYPE 19 subroutine.

```
ASSIGN\ TRNSYS15\ TEST\ TEST.LST        6
ASSIGN\ TRNSYS15\ TEST\ TEST.OUT        10
ASSIGN\ TRNSYS15\ TEST\ TEST.PLT        11
```

2.2.13 THE INCLUDE STATEMENT

In certain situations it is advantageous to have part of the TRNSYS input file (for example, equations passed from another program) in a separate file and include it with an INCLUDE statement into the main file. In this way, it is possible to change items or rewrite the include file without changing the main TRNSYS input file. The syntax is:

```
INCLUDE c:\trnsys15\file.inc
```

When the TRNSYS input file reader reaches this line, it opens the new file (file.inc). TRNSYS reads in any valid TRNSYS statements in this second file, closes the second file and returns to the main TRNSYS input file. TRNSYS continues reading the input file from the point of the INCLUDE statement.

The list file contains all of the statements in this file and the main input file. It is not possible, with the release of TRNSYS 15, to do recursive INCLUDE statements.

2.2.14 THE END STATEMENT

The END statement must be the last line of a TRNSYS input file. It signals the TRNSYS processor that no more control statements follow and that the simulation may begin. The END statement has the simple form:

```
END
```

Any data, such as data lines read by the TYPE 9 data reader described in Chapter 4, which will be read during the simulation process from the same file as that containing the simulation input file must immediately follow the END statement.

2.3 COMPONENT CONTROL STATEMENTS

A description of a modular system includes all of the information contained in the information flow diagram of the system as well as the number of INPUTS, PARAMETERS and differential equations (DERIVATIVES) employed in each component model, the type of components being used, the values of their PARAMETERS, and the initial values of their INPUTS and time-dependent variables. All of this information is conveyed to TRNSYS through the use of six component control statements, followed by required data.

The control statements are

1. UNIT-TYPE
2. PARAMETERS
3. INPUTS
4. DERIVATIVES
5. TRACE
6. FORMAT (for TYPES 25 and 28 only)

Each component in the system model is identified by its UNIT-TYPE statement. Following each UNIT-TYPE statement are the PARAMETERS, INPUTS and DERIVATIVES control statements (and possibly the FORMAT and TRACE statements); each followed by the supplementary data they require. The next six sections describe in detail the format of each of these control statements.

2.3.1 THE UNIT-TYPE STATEMENT

The information about each component in the system begins with a UNIT-TYPE statement, which has the following format.

UNIT n TYPE m Comment

where

n is the UNIT number of the component. Allowable UNIT numbers are integers between 1 and n, where n is set in param.inc. See Section 6.5 for details.*

m is the TYPE number of the component. Allowable TYPE numbers are integers between 1 and 99.*

Comment is an optional comment of up to 46 characters. The comment is reproduced on the output but is otherwise disregarded. Its function is primarily to help the user associate the UNIT and TYPE numbers with a particular component in the system.

Every system component must have and begin with a UNIT-TYPE statement. The UNIT number specified on this statement must be unique. Two UNITS cannot share the same number.

Examples:

UNIT 6 TYPE 15 EXAMPLE COMPONENT

UNIT 26 TYPE 26 PLOTTER

---

*These and other limits may be changed by modifying the TRNSYS processor as described in Section 6.5. The given limits should be sufficient for most users.

UNIT 1 TYPE 4 TANK

2.3.2 THE PARAMETERS STATEMENT

The PARAMETERS of a component may be specified by the PARAMETERS control statement and supplementary data following it. The PARAMETERS control statement has the following format:

PARAMETERS n

where

n is the number of PARAMETERS to follow on the next line(s). Typically this is the number of parameters required by the component, but may be less if more than one PARAMETERS statement is used for a given component.

The next non-comment line in the input file must contain the values of the n PARAMETERS in their proper order. Any format is allowable; a comma or one or more blanks must separate each value. The values may be placed on more than 1 line, if necessary. These lines have the general form:

$V_1, V_2, \ldots, V_i, \ldots, V_n$

where

V_i is the value of the i^{th} PARAMETER, or a name defined by a CONSTANTS or EQUATIONS command

The PARAMETERS statement defines the PARAMETERS for the UNIT specified on the previous UNIT-TYPE statement. (INPUTS or DERIVATIVES statements for the given unit may precede the PARAMETERS statement.) Any number of PARAMETERS statements are allowed for each UNIT (except for units of TYPES 24-29, which allow only one per unit).

TRNSYS has no provision for default values. All n values of the PARAMETERS must be specified on the supplementary data lines in the order they are expected by the component model. If a component model requires no PARAMETERS, the PARAMETERS control statement and the supplementary data following it should be omitted.

Example: The TYPE 3 pump model described in Chapter 4 requires 4 PARAMETERS:

```
UNIT 1 TYPE 3 PUMP
PARAMETERS 4
100. 4.19 100. 0.2
```

Some component descriptions require a large number of parameters that are logically organized as separate lists. This is the case for the TYPE 19 Zone and TYPE 40 microprocessor components. In this situation, it is advantageous to utilize more than one PARAMETERS

statements for one component description. As always, a data line (or lines) containing the number of values specified by the PARAMETERS statement must follow each PARAMETERS statement line.

```
UNIT 1  TYPE 4 TANK
PARAMETERS 3
2  .42  4.19
PARAMETERS 3
1000  1.44  -1.69
```

Internal limits in TRNSYS restrict the number of parameters/unit to 500 and the number of parameters/simulation to n, which is set in param.inc. See Section 6.5 for details. This limit may be changed by an experienced TRNSYS user. Refer to the TRNSYS error handling section in Chapter 6 for more details.

2.3.3 THE INPUTS STATEMENT

In general, the INPUT values for a component are OUTPUT values from other components in the system model. It is thus necessary to specify the appropriate UNIT and OUTPUT variable number for each INPUT variable of each component. In addition, TRNSYS requires that an initial value be specified for each INPUT. This information is conveyed by the INPUTS control statement and at least two supplementary data lines following it. As with the parameters, any number of INPUTS statements are allowed for each UNIT (except for units of TYPES 24-29). The INPUTS control statement has the following format:

```
INPUTS n
```

where

n is the number of INPUTS to follow on the next line(s). Typically this is the number of inputs required by the component, but may be less if more than one INPUTS statement is used for a given component.

The first non-comment line following the INPUTS control statement specifies the UNIT and position numbers of the OUTPUT variables that are to be the INPUT variables to the component. This line has the following format:

$u_1,o_1 \quad u_2,o_2 \ldots u_i,o_i \ldots u_n,o_n$

where

u_i is an integer number referencing the number of the UNIT to which the ith INPUT is connected.

o_i is an integer number indicating to which OUTPUT (i.e., the 1st, 2nd, etc.) of UNIT number ui the ith INPUT is connected.

A u_i,o_i pair may be replaced by an EQUATION-defined variable name. If the initial value (see below) of the input variable is to be taken as the input variable value throughout the simulation, the user should specify $u_i = 0$ and $o_i = 0$ or use the word 'CONST' (see Section 2.2.5). This feature is useful when it is desired to hold one or more INPUTS to a component constant while investigating the effects of the variations of other INPUTS.

It is recommended u_i be separated from o_i by a comma, and o_i be separated from u_{i+1} by several blanks. This format, although not required, improves the readability. More than one line may be used if necessary.

There are two forms of the second data line. For all components except TYPE 25 printers, TYPE 26 plotters, TYPE 27 histogram plotters and TYPE 65 online plotters this line specifies the initial values of the n INPUT variables in the following format:

$$V_1, V_2, \ldots, V_i, \ldots, V_n$$

where

V_i is the initial value of the ith INPUT variable.*

There is no second data line for TYPE 28 simulation summaries.

All n values must be specified and must appear in the order expected by the component TYPE. No default values are assumed. More than 1 line may be used if necessary.

The other form of this data line is only for the TYPE 25 Printer, the TYPE 26 Plotter, the TYPE 27 Histogram Plotter and the TYPE 65 Online Plotter components. For these component TYPES, the second data line must contain a label for each of the n INPUTS. The labels are used to identify the printed or plotted INPUTS. (See the description of TYPES 25, 26, 27 and 65) The format of this form of the 2nd data line is

$$\text{Label}_1, \text{Label}_2, \ldots \text{Label}_i, \ldots \text{Label}_n$$

where

Label_i is the printer or plotter label for the ith INPUT.

Labels may be up to 6 characters in length and must not contain blanks or commas. Labels must be separated by at least one blank or comma.

Example 1: A TYPE 2 component has 3 INPUTS. The first INPUT is the 2nd OUTPUT from UNIT 17, the 2nd INPUT is the 4th OUTPUT from UNIT 3, and the 3rd INPUT is the 1st OUTPUT of UNIT 7. The initial values of the three inputs are 0, 100, and .5 respectively. The 3 lines needed to convey this information to TRNSYS are

---

*This value may be a number, or may be represented by a CONSTANT-defined variable name or a constant EQUATION-defined variable name.

```
INPUTS 3
17,2  3,4   7,1
0.0   100.0 .5
```

Example 2: A TYPE 25 printer is to print the values of two system OUTPUTS as the simulation progresses. These two outputs are INPUTS to the printer. The first INPUT is the third OUTPUT from UNIT 17. The second INPUT is the 1st OUTPUT from UNIT 20. The two printed values are to be identified with the labels TAMB and TSOL, respectively. The following 3 lines communicate this information

```
INPUTS 2
17,3      20,1
TAMB      TSOL
```

Example 3: Continuing the example from the last two sections, the inputs to the UNIT 1 TYPE 4 tank could be as follows:

INPUT #1 is connected to UNIT 2, OUTPUT 1
INPUT #2 is connected to UNIT 2, OUTPUT 2
INPUT #3 is connected to UNIT 3, OUTPUT 1
INPUT #4 is connected to UNIT 3, OUTPUT 2
INPUT #5 is to be temporarily held constant with a value of 60

Initial values of these INPUTS are to be 60., 0.0, 21., 0.0, and 60., respectively. The control lines for this component are now:

```
UNIT 1  TYPE 4  TANK
PARAMETERS 6
2  .42  4.19  1000  1.44  -1.69
INPUTS 5
2,1  2,2  3,1  3,2  0,0*
60.  0.0  21.  0.0  60.*
```

Internal limits in TRNSYS restrict the total number of inputs/simulation to a value which is set in the file param.inc and defaults to 750 and the number of inputs/unit to 500. For listing purposes, only 50 inputs/unit will be printed.

2.3.4 THE DERIVATIVES STATEMENT

---

*Note the use of the constant input feature.

The number of numerically solved time-dependent differential equations involved in the mathematical model of a system component, and the initial values of the corresponding dependent variables are specified by the DERIVATIVES control statement and a data line following it. The DERIVATIVES statement has the following format:

DERIVATIVES n

where

n is the number of numerically solved time-dependent differential equations in the component model.

The data line following the DERIVATIVES statement contains the initial values of the n dependent variables in their proper order. The initial values may be placed on more than 1 line if necessary. If the component model does not use the DTDT array (see Section 3.3.2) to solve time-dependent differential equations, the DERIVATIVES control statement and the data line following it should be omitted. Only one DERIVATIVES statement is allowed per UNIT.

Example 1: A hypothetical TYPE 73 component numerically solves 3 differential equations in its mathematical description of the component. The initial values of the three time-dependent solutions of the differential equations are 100.0, 80.0, and 60.0, respectively. The following two lines convey this information:

```
DERIVATIVES 3
100.0, 80.0, 60.0
```

Example 2: Continuing the example from the previous sections, the one segment TYPE 4 tank uses one differential equation. The initial value of the time-dependent solution, in this case the initial temperature of the tank, is to be 60. The control statements for this component are now:

```
UNIT 1 TYPE 4 TANK
PARAMETERS 6
2 .42 4.19 1000 1.44 -1.69
INPUTS 5
2,1 2,2 3,1 3,2 0,0
60. 0.0 21. 0.0 60.
DERIVATIVES 1
60.
```

Internal limits in TRNSYS restrict the number of derivatives per simulation. This limit is set in the include file called param.inc and defaulted to 100. See Section 6.5 for more details.

2.3.5 THE TRACE STATEMENT

The use of the TRACE control statement is optional. When included among the control statements for a component, it causes the values of the PARAMETERS, INPUTS, OUTPUTS and DERIVATIVES of that component to be printed out whenever that component is called by TRNSYS during a simulation. This feature is useful in debugging user-written TYPE subroutines (explained in Chapter 3) and for investigating problems encountered in TRNSYS simulations. As TRACE can produce voluminous output, two specifications must be made on the TRACE statement: the times in the simulation at which TRACE output is to start and stop. The TRACE statement has the following format:

TRACE t_{on} t_{off}

where

t_{on} is the TIME in the simulation at which TRACE output is to begin
t_{off} is the TIME in the simulation at which TRACE output is to stop

As with the PARAMETERS, INPUTS, and DERIVATIVES statements, the TRACE statement refers to the previous UNIT-TYPE statement. The ordering of these control statements, following the UNIT-TYPE statement, is optional. Only one TRACE statement may be present for each component.

Example 1:. TRACE a component for 24 hours starting at the beginning of the simulation.

```
.
.
.
TRACE 0 24
.
.
```

Example 2: TRACE the UNIT 1 TYPE 4 TANK defined in previous examples from noon until 2 pm the first day of a simulation.

```
UNIT 1 TYPE 4 TANK
PARAMETERS 6
2 .42 4.19 1000 1.44 -1.69
INPUTS 5
2,1 2,2 3,1 3,2 0,0
60. 0.0 21. 0.0 60.
DERIVATIVES 1
60
TRACE 12. 14.
```

An automatic TRACE will be performed on any component which is called, during one

timestep, more than the number of times specified in the LIMITS command (see Section 2.2.3).

2.3.6 THE FORMAT STATEMENT

An optional FORMAT statement can be used with the TYPE 25 printer and the TYPE 28 Simulation Summarizer to control the format of output which is directed to a file.

The format statement format is simply

FORMAT
(valid FORTRAN Format Specification)

The FORTRAN format specification must have an open parenthesis in column 1 and must be no longer than 80 characters. TRNSYS prints only real numbers so the format specifications for numbers must be either F or E formats.

The FORMAT statement only affects output for TYPE 25 for which $L_{unit}>0$ (set by a PARAMETER) and additionally, for TYPE 28, only for Output Mode =1; otherwise it is ignored. Note that when $L_{unit}>0$, both TYPE 25 and TYPE 28 output tabular columns of numbers with the simulation time in the first column. A format specification must be provided for time, as well as for the outputs of the printing unit.

2.4 LISTING CONTROL STATEMENTS

Listing control statements are used to specify the format of the TRNSYS output listing. They are:

1.WIDTH
2.NOLIST
3.LIST
4.MAP

These control statements are all optional and default values are assumed if they are not present in the TRNSYS input file. The format and use of each of these control statements is defined in the next four sections. Only one WIDTH and one MAP card are allowed in a TRNSYS input file.

2.4.1 THE WIDTH STATEMENT

The WIDTH statement is an optional control statement is used to set the number of characters to be allowed on a line of TRNSYS output. The format of the WIDTH statement is as

follows:

WIDTH n

where

n is the number of characters per printed line; n must be between 72 and 132.

The WIDTH specification affects both the output of the TRNSYS processor and the output of the output-producing components described in Chapter 4. Suggested values of n are as follows:

video screen 72
line printer 120 (or 132 if available)

If a WIDTH statement is not included in a TRNSYS input file, the number of printed characters per line is assumed to be 120 (mainframe) or 72 (personal computer).

Example: If a 132 character/line line printer is available, then a WIDTH statement of the following form may be included in the TRNSYS input file.

WIDTH 132

The WIDTH specification of 132 characters will allow the entire width of the printer to be used.

2.4.2 THE NOLIST STATEMENT

The NOLIST statement is used to turn off the listing of the TRNSYS input file. It has the simple form:

NOLIST

The NOLIST control statement affects only the output of the TRNSYS processor and not the operation of the TYPES 25-29 printers and plotter. It is useful in reducing TRNSYS output. The NOLIST statement does not override the printing of ERROR messages. As many NOLIST statements as desired may be placed in a TRNSYS input file. The NOLIST statement may be placed before the SIMULATION statement. It is best not to use the NOLIST statement until the input file has been thoroughly debugged.

2.4.3 THE LIST STATEMENT

The LIST statement is used to turn on the TRNSYS processor listing after it has been

turned off by a NOLIST statement. It has the form:

LIST

The listing is assumed to be on at the beginning of a TRNSYS input file. As many LIST cards as desired may appear in a TRNSYS input file and may be located anywhere in the input file.

Example: The TRNSYS processor listing is to be turned off at the beginning of the input file, turned on for one set of component control cards and then turned back off for the rest of the input file. The following sequence of control cards is then used:

```
SIMULATION 0 24 .25
NOLIST
  .
  .
  .
LIST
UNIT 1 TYPE 4
  .
  .
  .
NOLIST
  .
END
```

.4 THE MAP STATEMENT

The MAP statement is an optional control statement that is used to obtain a component map listing which is particularly useful in debugging component interconnections. The atement has the form:

P

nition of the MAP control statement causes TRNSYS to print each component by its YPE numbers, followed by a list of its connected outputs and the UNIT, TYPE, and ber to which each is connected. This printout occurs only after the EXEC subroutine TRNSYS kernel routines) has been called and therefore will not appear if the terminated by errors recognized by the input file processor. A sample map appears

TRNSYS COMPONENT OUTPUT MAP

UNIT 1	TYPE 9	UNIT/TYPE/INPUT		
OUTPUT	4	3	1	5
OUTPUT	5	10	16	1
OUTPUT	6	3	1	3
		4	12	3
UNIT 10	TYPE 16	UNIT/TYPE/INPUT		
OUTPUT	1	3	1	4

etc.

2.5 COMMENT LINES

Comment lines may be included at any location in the TRNSYS input file or data* A comment line has the following format:

* Comment

The '*' must appear in column one of the line and any line with a '*' in column one will be interpreted as a comment line. The entire line is printed without modification.

Example:
```
*THIS IS AN EXAMPLE OF A COMMENT LINE
*THIS IS ANOTHER ONE
*ETC.
```

2.6 CONTROL STATEMENT EXAMPLE

Assume a system is to be modeled using two components, a simple tank and pump. A printer will be included to print out the temperature and flow rate of the fluid coming out of the pump.

```
*24 HOUR SIMULATION WITH 1/4 HOUR TIME-STEPS
SIMULATION  0 24 .25
ASSIGN  TEST.PLT  21

*REFER TO CHAPTER IV FOR THE TYPE 4 PUMP
UNIT 1  TYPE 4  TANK
```

Comment lines after the END statement are not printed. Comment lines in this "data" part of the deck may be useful for separation of data groups.

```
PARAMETERS 20
2 .42 4.19 1000 1.44 -1.69 1
1 1 55 3 16200
2 2 55 3 16200
0. 20. 100.
INPUTS 7
2,1 2,2 1,3 1,4 0,0 0,0 0,0
60. 0.0 60. 0.0 60. 1.0 1.0
DERIVATIVES 2
60. 60.
*REFER TO CHAPTER IV FOR THE TYPE 3 PUMP
UNIT 2 TYPE 3 PUMP
PARAMETERS 4
350 4.190 100. 0.
INPUTS 3
1,1 1,2 0,0
60. 350. 1.0

UNIT 3 TYPE 25 PRINTER
PARAMETERS 5
1. 0 24 21 2
*THE PRINTER PRINTS EVERY HOUR
INPUTS 2
2,1 2,2
FLOW TEMP

END
```

2.7 DATA ECHO AND CONTROL STATEMENT ERRORS

The information contained for each statement of the TRNSYS input file described in the preceding sections is processed by TRNSYS. Provided there are no errors on the line and that the listing has not been turned off by a NOLIST statement, it is reproduced in a neat format on the output before the simulation begins. With a few exceptions, any errors detected in the input file will terminate the attempted simulation after all the lines in the input file have been processed. TRNSYS is programmed to respond to most input errors with appropriate error messages, however, the error checking facility of TRNSYS is not foolproof. Section 6.3 discusses the TRNSYS error and warning messages in more depth.

2.8 SUMMARY

A TRNSYS simulation is defined and controlled by a set of control statements and data

lines as described in this chapter. A SIMULATION statement must appear in each TRNSYS input file, usually near the beginning of the input file. TOLERANCES, LIMITS, CONSTANTS, EQUATIONS, ACCELERATE, LOOP-REPEAT, DFQ, NOCHECK, SOLVER and ASSIGN statements are optional. These control statements are discussed in Section 2.2.

Each component in the system model is identified by a UNIT-TYPE statement, followed by statements that assign values to component parameters, identify sources for all component inputs, and assign initial values to inputs and to time-dependent differential equations (if any). An optional TRACE statement is available to aid in debugging user-written TYPE subroutines (see Chapter 3) and for investigating problems encountered in TRNSYS simulations. An optional FORMAT statement can be used to specify the format of output from TYPE 25 (printer) and TYPE 28 (simulation summary). These component control statements are discussed in Section 2.3

Several optional statements are available which influence the TRNSYS output listing. These are discussed in Section 2.4. The MAP control statement is particularly useful in debugging component interconnections.

Comment lines and blank lines may also be included in the TRNSYS input file. The control input file must end with an END statement.

The TRNSYS processor requires only the first three characters of each control statement. Thus SIM can be used for SIMULATION, TOL for TOLERANCES, etc.

References

1. SOLMET, Volume 2 - Final Report, "Hourly Solar Radiation Surface Meteorological Observations", TD-9724 (1979).
2. SOLMET Typical Meteorological Year, Tape 9734, National Oceanic and Atmospheric Administration, Environmental Data Service, National Climatic Center, Asheville, North Carolina.
3. Steven C. Chapra and Raymond P. Canale, Numerical Methods for Engineers with Personal Computer Application (New York: McGray-Hill, 1985), p. 488.
4. Steven C. Chapra and Raymond P. Canale, p. 483.
5. Steven C. Chapra and Raymond P. Canale, p. 530.
6. Gerald C.F., Wheatley P.O. (1984), Applied Numerical Analysis, Addison-Wesley, p. 190.
7. Broyden C.G. (1965), "A Class of Methods for Solving Nonlinear Simultaneous equations", Math, Comp., 19, 577-593
8. Gear C.W. (1967), "The Numerical Integration of Ordinary Differential Equations", Math. Comp., 21, 22-49
9. Duff I.S., Erisman A.M., Reid J.K. (1986), Direct Methods for Sparce Matrices, Oxford Science Publications, Claredon Press.
10. Powell M.J.D. (1970), "A hybrid method for non-linear equations", Computer Journal,
11. Powell M.J.D., (1970), "A Fortran Subroutine for Solving Systems of Nonlinear Algebraic Equations", Computer Journal, 12, 115-161.
12. Duffie J.A., Beckman W.A. (1991), Solar Engineering of Thermal Processes, - 2nd edition., Wiley-Interscience publication, New York

13. Eckstein, J.H., "Detailed Modeling of Photovoltaic System Components", M.S. Thesis, Dept. of Mechanical Engineering, University of Wisconsin - Madison, 1990