CSE User's Manual

California Simulation Engine

Contents

1 Introduction **4**
1.1 Greetings 4
1.2 Manual Organization 5
1.3 Installation 5
1.3.1 Hardware and Software Requirements 5
1.3.2 Installation Procedure 5

2 Operation **6**
2.1 Command Line 6
2.2 Locating Files 6
2.3 Output File Names 7
2.4 Errorlevel 7

3 Input Structure **7**
3.1 Introduction 7
3.2 Form of the CSE Data 8
3.3 Overview of CSE Input Language 9
3.3.1 Statements – Overview 9
3.3.2 Nested Objects 10
3.3.3 Expressions – Overview 10
3.3.4 The Preprocessor – Overview 11
3.4 The Preprocessor 12
3.4.1 Line splicing 12
3.4.2 Macro definition and expansion 13
3.4.3 File inclusion 14
3.4.4 Conditional inclusion of text 14
3.4.5 Preprocessor examples 15
3.5 CSE Input Language Statements 16
3.5.1 Object Statements 16
3.5.1.1 Object Names 16
3.5.1.2 ALTER 17
3.5.1.3 DELETE 18
3.5.1.4 LIKE clause 18
3.5.1.5 COPY clause 18
3.5.1.6 USETYPE clause 19
3.5.1.7 DEFTYPE 19
3.5.1.8 END and ENDxxxx 20
3.5.2 Member Statements 20
3.5.2.1 UNSET 20
3.5.2.2 REQUIRE 20
3.5.2.3 FREEZE 21
3.5.3 Action Commands 21
3.5.3.1 RUN 21
3.5.3.2 CLEAR 21
3.6 Expressions 21
3.6.1 Expression Types 22
3.6.2 Constants 23
3.6.3 Operators 24
3.6.4 System Variables 25
3.6.5 Built-in Functions 26
3.6.5.1 brkt 26
3.6.5.2 fix 27

3.6.5.3 toFloat . . . 27
3.6.5.4 min . . . 27
3.6.5.5 max . . . 27
3.6.5.6 choose . . . 27
3.6.5.7 choose1 . . . 27
3.6.5.8 select . . . 28
3.6.5.9 hourval . . . 29
3.6.5.10 abs . . . 29
3.6.5.11 sqrt . . . 29
3.6.5.12 exp . . . 29
3.6.5.13 logE . . . 29
3.6.5.14 log10 . . . 29
3.6.5.15 sin . . . 29
3.6.5.16 sind . . . 30
3.6.5.17 asin . . . 30
3.6.5.18 asind . . . 30
3.6.5.19 cos . . . 30
3.6.5.20 cosd . . . 30
3.6.5.21 acos . . . 30
3.6.5.22 acosd . . . 30
3.6.5.23 tan . . . 30
3.6.5.24 tand . . . 31
3.6.5.25 atan . . . 31
3.6.5.26 atand . . . 31
3.6.5.27 atan2 . . . 31
3.6.5.28 atan2d . . . 31
3.6.5.29 pow . . . 31
3.6.5.30 enthalpy . . . 31
3.6.5.31 wFromDbWb . . . 31
3.6.5.32 wFromDbRh . . . 31
3.6.5.33 import . . . 32
3.6.5.34 importStr . . . 32
3.6.5.35 contin . . . 32
3.6.5.36 stepped . . . 32
3.6.6 User-defined Functions . . . 32
3.6.7 Probes . . . 33
3.6.8 Variation Frequencies Revisited . . . 35

4 Input Data **36**
4.1 TOP Members . . . 36
4.1.1 TOP General Data Items . . . 36
4.1.2 TOP Daylight Saving Time Items . . . 40
4.1.3 TOP Model Control Items . . . 41
4.1.4 TOP Weather Data Items . . . 43
4.1.5 TOP TDV (Time Dependent Value) Items . . . 46
4.1.6 TOP Report Data Items . . . 47
4.1.7 TOP Autosizing . . . 48
4.1.8 TOP Debug Reporting . . . 49
4.2 HOLIDAY . . . 50
4.3 MATERIAL . . . 52
4.4 CONSTRUCTION . . . 54
4.5 LAYER . . . 54
4.6 GLAZETYPE . . . 56
4.7 METER . . . 58

4.8 DHWMETER . . . 60
4.9 ZONE . . . 61
4.9.1 ZONE General Members . . . 61
4.9.2 ZONE Infiltration . . . 64
4.9.3 ZONE Exhaust Fan . . . 65
4.10 GAIN . . . 67
4.11 SURFACE . . . 70
4.12 WINDOW . . . 77
4.13 SHADE . . . 82
4.14 SGDIST . . . 84
4.15 DOOR . . . 86
4.16 PERIMETER . . . 90
4.17 IZXFER . . . 90
4.18 RSYS . . . 96
4.19 DUCTSEG . . . 105
4.20 DHWDAYUSE . . . 108
4.21 DHWUSE . . . 110
4.22 DHWSYS . . . 111
4.23 DHWHEATER . . . 115
4.24 DHWTANK . . . 121
4.25 DHWPUMP . . . 123
4.26 DHWLOOP . . . 123
4.27 DHWLOOPPUMP . . . 125
4.28 DHWLOOPSEG . . . 125
4.29 DHWLOOPBRANCH . . . 127
4.30 PVARRAY . . . 129
4.31 SHADEX . . . 133
4.32 BATTERY . . . 134
4.33 REPORTFILE . . . 137
4.34 REPORT . . . 139
4.35 REPORTCOL . . . 144
4.36 EXPORTFILE . . . 145
4.37 EXPORT . . . 146
4.38 EXPORTCOL . . . 149
4.39 IMPORTFILE . . . 151

5 Output Reports 153
5.1 Units . . . 153
5.2 Time . . . 153
5.3 METER Reports . . . 153
5.4 Energy Balance Report . . . 153
5.5 Air Handler Load Report . . . 154
5.6 Air Handler Report . . . 155

6 Probe Definitions 156
6.1 @ahRes[1..]. . . . 156
6.2 @airHandler[1..]. . . . 160
6.3 @boiler[1..]. (owner: heatPlant) . . . 181
6.4 @chiller[1..]. (owner: coolPlant) . . . 183
6.5 @construction[1..]. . . . 189
6.6 @coolPlant[1..]. . . . 190
6.7 @DHWDayUse[1..]. . . . 198
6.8 @DHWHeater[1..]. (owner: DHWSys) . . . 198
6.9 @DHWLoop[1..]. (owner: DHWSys) . . . 200

6.10 @DHWLoopBranch[1..]. (owner: DHWLoopSeg) . . . 201
6.11 @DHWLoopPump[1..]. (owner: DHWLoop) . . . 202
6.12 @DHWLoopSeg[1..]. (owner: DHWLoop) . . . 202
6.13 @DHWmeter[1..]. . . . 203
6.14 @DHWPump[1..]. (owner: DHWSys) . . . 203
6.15 @DHWSys[1..]. . . . 204
6.16 @DHWTank[1..]. (owner: DHWSys) . . . 207
6.17 @DHWUse[1..]. (owner: DHWDayUse) . . . 207
6.18 @door[1..]. (owner: surface) . . . 208
6.19 @DuctSeg[1..]. (owner: RSYS) . . . 219
6.20 @export[1..]. (owner: exportFile) . . . 222
6.21 @exportCol[1..]. (owner: export) . . . 222
6.22 @exportFile[1..]. . . . 223
6.23 @gain[1..]. (owner: zone) . . . 223
6.24 @glazeType[1..]. . . . 224
6.25 @heatPlant[1..]. . . . 224
6.26 @holiday[1..]. . . . 230
6.27 @impFileFldNames[1..]. . . . 231
6.28 @importFile[1..]. . . . 231
6.29 @izXfer[1..]. . . . 232
6.30 @layer[1..]. (owner: construction) . . . 235
6.31 @mass[1..]. . . . 236
6.32 @material[1..]. . . . 238
6.33 @meter[1..]. . . . 238
6.34 @perimeter[1..]. (owner: zone) . . . 241
6.35 @PVArray[1..]. . . . 242
6.36 @report[1..]. (owner: reportFile) . . . 243
6.37 @reportCol[1..]. (owner: report) . . . 244
6.38 @reportFile[1..]. . . . 244
6.39 @RSYS[1..]. . . . 245
6.40 @RSYSRes[1..]. . . . 261
6.41 @sgdist[1..]. (owner: window) . . . 262
6.42 @shade[1..]. (owner: window) . . . 262
6.43 @surface[1..]. (owner: zone) . . . 262
6.44 @terminal[1..]. (owner: zone) . . . 274
6.45 @top. . . . 281
6.46 @towerPlant[1..]. . . . 292
6.47 @weather. . . . 299
6.48 @weatherFile. . . . 299
6.49 @weatherNextHour. . . . 301
6.50 @window[1..]. (owner: surface) . . . 301
6.51 @xsurf[1..]. . . . 313
6.52 @zhx[1..]. (owner: zone) . . . 324
6.53 @znRes[1..]. . . . 326
6.54 @zone[1..]. . . . 341

1 Introduction

1.1 Greetings

The purpose of this manual is to document the California Simulation Engine computer program, CSE. CSE is an hourly building and HVAC simulation program which calculates annual energy requirements for building space conditioning and lighting. CSE is specifically tailored for use as internal calculation machinery for compliance with the California building standards.

CSE is a batch driven program which reads its input from a text file. It is not intended for direct use by people seeking to demonstrate compliance. Instead, it will be used within a shell program or by technically oriented users. As a result, this manual is aimed at several audiences:

1. People testing CSE during its development.
2. Developers of the CSE shell program.
3. Researchers and standards developers who will use the program to explore possible conservation opportunities.

Each of these groups is highly sophisticated. Therefore this manual generally uses an exhaustive, one-pass approach: while a given topic is being treated, *everything* about that topic is presented with the emphasis on completeness and accuracy over ease of learning.

Please note that CSE is under development and will be for many more months. Things will change and from time to time this manual may be inconsistent with the program.

1.2 Manual Organization

This Introduction covers general matters, including program installation.

Operation documents the operational aspects of CSE, such as command line switches, file naming conventions, and how CSE finds files it needs.

Input Structure documents the CSE input language in general.

Input Data describes all of the specific input language statements.

Output Reports will describe the output reports.

Lastly, Probe Definitions lists all available probes.

1.3 Installation

1.3.1 Hardware and Software Requirements

CSE is a 32-bit Microsoft Windows console application. That is, it runs at the command prompt on Windows XP, Windows Vista, and Windows 7. Memory and disk space requirements depend on the size of projects being modeled, but are generally modest.

To prepare input files, a text editor is required. Notepad will suffice, although a text editor intended for programming is generally more capable. Alternatively, some word processors can be used in "ASCII" or "text" or "non-document" mode.

1.3.2 Installation Procedure

Create a directory on your hard disk with the name \CSE or some other name of your choice. Copy the files into that directory. Add the name of the directory to the PATH environment setting unless you intend to use CSE only from the CSE directory.

2 Operation

2.1 Command Line

CSE is invoked from the command prompt or from a batch file using the following command:

```
CSE *inputfile* {*switches*}
```

where:

inputfile specifies the name of the text input file for the run(s). If the filename has an extension other than ".cse" (the default), it must be included. The name of the file with weather data for the simulation(s) is given in this file (wfName= statement, see Weather Data Items).

{switches} indicates zero or more of the following:

- -D*name* defines the preprocessor symbol *name* with the value "". Useful for testing with `#ifdef name`, to invoke variations in the simulation without changing the input file. The CSE preprocessor is described "The Preprocessor".
- -D*name*=*value* defines the preprocessor symbol *name* with the specified *value*. *Name* can then be used in the input file to allow varying the simulation without changing the input file – see "The Preprocessor" for more information. The entire switch should be enclosed in quotes if it contains any spaces – otherwise the command processor will divide it into two arguments and CSE will not understand it.
- -b batch mode: CSE will never stop for a response from the user when an error occurs. Error messages may thus scroll off the screen, but will all be in the error message file.
- -p display all the class and member names that can be "probed" or accessed in CSE expressions. "Probes" are described in "Probes". Use with command processor redirection operator ">" to obtain a report in a file. *Inputfile* may be given or omitted when -p is given.
- -q similar to -p, but displays additional member names that cannot be probed or would not make sense to probe in an input file (development aid).
- -x specifies report test prefix; see TOP repTestPfx. The -x command line setting takes precedence over the input file value, if any.

2.2 Locating Files

As with any program, in order to invoke CSE, the directory containing CSE.EXE must be the current directory, or that directory must be on the operating system path, or you must type the directory path before CSE.

A CSE simulation requires a weather file. The name of the weather file is given in the CSE input file (`wfName=` statement, see Weather Data Items). The weather file must be in one of the same three places: current directory, directory containing CSE.EXE, or a directory on the operating system path; or, the directory path to the file must be given in the `wfName=` statement in the usual pathName syntax. ?? Appears that file must be in current directory due to file locating bugs 2011-07

The CSE input file, named on the CSE command line, must be in the current directory or the directory path to it must be included in the command line.

Included input files, named in `#include` preprocessor directives (see "The Preprocessor") in the input file, must be in the current directory or the path to them must be given in the `#include` directive. In particular, CSE will NOT automatically look for included files in the directory containing the input file. The default extension for included files is ".CSE".

Output files created by default by CSE (error message file, primary report and export files) will be in the same directory as the input file; output files created by explicit command in the input file (additional report and/or export files) will be in the current directory unless another path is explicitly specified in the command creating the file.

2.3 Output File Names

If any error or warning messages are generated, CSE puts them in a file with the same name and path as the input file and extension .ERR, as well as on the screen and, usually, in the primary (default) report file. The exception is errors in the command line: these appear only on the screen. If there are no error or warning messages, any prior file with this name is deleted.

By default, CSE generates an output report file with the same name and path as the input file, and extension ".REP". This file may be examined with a text editor and/or copied to an ASCII printer. If any exports are specified, they go by default into a file with the same name and path as the input file and extension ".EXP".

In response to specifications in the input file, CSE can also generate additional report and export files with user-specified names. The default extensions for these are .REP and .CSV respectively and the default directory is the current directory; other paths and extensions may be specified. For more information on report and export files, see REPORTFILE and EXPORTFILE in "Input Data".

2.4 Errorlevel

CSE sets the command processor ERRORLEVEL to 2 if any error occurs in the session. This should be tested in batch files that invoke CSE, to prevent use of the output reports if the run was not satisfactory. The ERRORLEVEL is NOT set if only warning messages that do not suppress or abort the run occur, but such messages DO create the .ERR file.

3 Input Structure

DRAFT: In the following, any text annotated with ?? indicates areas of uncertainty or probable change. As the program and input language develop, these matters will be resolved.

3.1 Introduction

The CSE Input Language is the fundamental interface to the CSE program. The language has been designed with three objectives in mind:

1. Providing direct access to all program features (including ones included for self-testing), to assist in program development.
2. Providing a set of parametric and expression evaluation capabilities useful for standards development and program testing.
3. Providing a means for other programs, such as an interactive user interface, to transmit input data and control data to the program.

Thus, the language is not intended to be used by the average compliance or simulation user. Instead, it will be used during program development for testing purposes and subsequently for highly technical parametric studies, such as those conducted for research and standards development. In all of these situations, power, reproducibility, and thorough input documentation take precedence over user-friendliness.

CSE reads its input from a file. The file may be prepared by the user with a text editor, or generated by some other program.

3.2 Form of the CSE Data

The data used by CSE consists of *objects*. Each object is of a *class*, which determines what the object represents. For example, objects of class ZONE represent thermally distinct regions of the building; each thermally distinct region has its own ZONE object. An object's class determines what data items or *members* it contains. For instance, a ZONE object contains the zone's area and volume. In addition, each object can have a *name*.

The objects are organized in a hierarchy, or tree-like structure. For example, under each ZONE object, there can be SURFACE objects to represent the walls, floors, and ceilings of the ZONE. Under SURFACEs there can be WINDOW objects to represent glazings in the particular wall or roof. SURFACE is said to be a *subclass* of the class ZONE and WINDOW a subclass of SURFACE; each individual SURFACE is said to be a *subobject* of its particular ZONE object. Conversely, each individual SURFACE is said to be *owned by* its zone, and the SURFACE class is said to be owned by the ZONE class.

The hierarchy is rooted in the one *top-level object* (or just *Top*). The top level object contains information global to the entire simulation, such as the start and end dates, as well as all of the objects that describe the building to be simulated and the reports to be printed.

Objects and their required data must be specified by the user, except that Top is predefined. This is done with input language *statements*. Each statement begins an object (specifying its class and object name) or gives a value for a data member of the object being created. Each object is specified with a group of statements that are usually given together, and the objects must be organized according to the hierarchy. For example, SURFACEs must be specified within ZONEs and WINDOWs within SURFACEs. Each SURFACE belongs to (is a subobject of) the ZONE within which it is specified, and each WINDOW is a subobject of its SURFACE.

The entire hierarchy of CSE classes can be represented as follows, using indentation to indicate subclasses:

TODO: review hierarchy

```
TOP (Top-level class; object of this class supplied automatically by CSE)
    HOLIDAY
    MATERIAL
    CONSTRUCTION
        LAYER
    METER
    DHWMETER
    IZXFER
    DHWDAYUSE
        DHWUSE
    DHWSYS
        DHWHEATER
        DHWTANK
        DHWPUMP
        DHWLOOP
        DHWLOOPPUMP
        DHWLOOPSEG
            DHWLOOPBRANCH
   ZONE
        GAIN
        SURFACE
            WINDOW
                SHADE
                SGDIST
            DOOR
    REPORTFILE
```

```
REPORT
REPORTCOL
EXPORTFILE
EXPORT
EXPORTCOL
```

3.3 Overview of CSE Input Language

The CSE Input Language consists of *commands*, each beginning with a particular word and, preferably, ending with a semicolon. Each command is either an *action-command*, which specifies some action such as starting a simulation run, or a *statement*, which creates or modifies an *object* or specifies a value for a *member* of an object.

3.3.1 Statements – Overview

A statement that creates an object consists basically of the *class name* followed by your name for the object to be created. (The name can be omitted for most classes; optional modifying clauses will be described later.) For example,

```
ZONE "north";
```

begins an object of class ZONE; the particular zone will be named "north". This zone name will appear in reports and error messages, and will be used in other statements that operate on the zone. As well as creating the ZONE, this statement sets CSE to expect statements specifying ZONE data members or ZONE subobjects to follow.

A statement specifying a data member consists of the data member's name, an = sign, an *expression* specifying the value, and a terminating semicolon. An expression is a valid combination of operands and operators as detailed later; commonly it is just a number, name, or text enclosed in quotes. For example,

```
znVol = 100000;
```

specifies that the zone has a volume of 100000 cubic feet. (If the statement occurs outside of the description of a ZONE, an error message occurs.) All of the member names for each class are described in the input data section; most of them begin with an abbreviation of the class name for clarity.

The description of a zone or any object except Top can be terminated with the word "END"; but this is not essential; CSE will assume the ZONE ends when you start another ZONE or any object not a subobject of ZONE, or when you specify a member of a higher level class (Top for ZONE), or give an action-command such as RUN.

Statements are free-form; several can be put on a line, or a single statement can occupy several lines. Indentation according to class hierarchy will help make your input file readable. Spaces may be used freely except in the middle of a word or number. Tab characters may be used. Each statement should end with a semicolon. If the semicolon is omitted and the statement and the following statement are both correctly formed, CSE will figure out your intent anyway. But when there is an error, CSE gives clearer error messages when the statements are delimited with semicolons.

Capitalization generally does not matter in input language statements; we like to capitalize class names to make them stand out. Words that differ only in capitalization are NOT distinct to CSE.

Comments (remarks) may be interspersed with commands. Comments are used to make the input file clearer to humans; they are ignored by CSE. A comment introduced with "//" ends at the end of the line; a comment introduced with "/*" continues past the next "*/", whether on the same line, next line, or many lines down. Additional input language may follow the */ on the same line.

3.3.2 Nested Objects

The following is a brief CSE input file, annotated with comments intended to exemplify how the input language processor follows the object hierarchy when decoding input describing objects and their subobjects.

```
// short example file
                        // initially, the current object is Top.
wfName = "CZ12RV2.CEC"; // give weather file name, a Top member
begDay = Jan 1;         // start and ...
endDay = Dec 31;        // ...end run dates: Top members.

MATERIAL carpet;        // create object of class MATERIAL
matThk = .296;          // specify 'matThk' member of MATERIAL 'carpet'
matCond = 1./24;        // give value of 'matCond' for 'carpet'

CONSTRUCTION slab140C;  /* create object of class CONSTRUCTION, named
                           slab140C. Terminates MATERIAL, because
                           CONSTRUCTION is not a subclass of material
                           in the hierarchy shown in another section** */
  LAYER                 /* start an unnamed object of class LAYER.
                           Since LAYER is a subclass of CONSTRUCTION,
                           this will be a subobject of slab140C. */
    lrMat = carpet;     /* member of the LAYER. Note use of name of
                           MATERIAL object. */
  // (additional layers would be here)

METER Elec;             /* create METER named Elec;
                           since METER is a subobject of Top,
                           this ends slab140C and its LAYER. */

ZONE North;             // start a ZONE named North.  Ends METER.
  znArea = 1000;        // specify data members of ZONE North.
  znVol = 10;           // (you don't have to capitalize these as shown.)
  GAIN NorthLights      /* create GAIN object named NorthLights.
                           Creates a subobject of ZONE North. */
    gnPower = 0.01;     // member of NorthLights -- numeric value
    gnMeter = Elec;     // member of NorthLights -- object name value

  znCAir = 3.5;         /* processor knows that znCAir is a member of ZONE;
                           thus this statement terminates the GAIN
                           subobject & continues ZONE 'North'. */

  /*lrMat = ...            would be an error here, because the current
                           object is not a LAYER nor a subobject of LAYER */

RUN;                    /* initiate simulation run with data given.
                           Terminates ZONE North, since action-commands
                           terminate all objects being constructed. */
```

** See Form of the CSE Data

3.3.3 Expressions – Overview

Expressions are the parts of statements that specify values – numeric values, string values, object name values, and choice values. Expressions are composed of operators and operands, in a manner similar to many programming languages. The available operators and operands will be described in the section on operators.

Unlike most programming languages, CSE expressions have Variation. *Variation* is how often a value changes during the simulation run – hourly, daily, monthly, yearly (i.e. does not change during run), etc. For instance, the operand `$hour` represents the hour of the day and has "hourly" variation. An expression has the variation of its fastest-varying component.

Each data member of each object (and every context in which an expression may be used) has its allowed *variability*, which is the fastest variation it will accept. Many members allow no variability. For example, `begDay`, the date on which the run starts, cannot meaningfully change during the run. On the other hand, a thermostat setting can change hourly. Thermostat settings and other scheduled values are specified in CSE with expressions that often make use of variability; there is no explicit SCHEDULE class.

For example, a heating setpoint that was 68 during business hours and 55 at night might be expressed as

```
select( $hour > 8 && $hour < 18, 68, default 55)
```

An example of a complete statement containing the above expression is:

```
tuTH = select( $hour > 8 && $hour < 18, 68, default 55);
```

The preceding is valid a statement if used in a TERMINAL description. The following:

```
begDay = select( $hour > 8 && $hour < 18, 68, default 55);
```

would always get an error message, because `begDay` (the starting day of the run) will not accept hourly variation, and the expression varies hourly, since it contains `$hour`. The expression's variation is considered "hourly" even though it changes only twice a day, since CSE has no variation category between hourly and daily.

CSE's expression capability may be used freely to make input clearer. For example,

```
znVol = 15 * 25 * 8;
```

meaning that the zone volume is 15 times 25 times 8 is the same to CSE as

```
znVol = 3000;
```

but might be useful to you to tersely indicate that the volume resulted from a width of 15, a length of 25, and a height of 8. Further, if you wished to change the ceiling height to 9 feet, the edit would be very simple and CSE would perform the volume calculation for you.

CSE computes expressions only as often as necessary, for maximum simulation speed. For example,

```
tuTH = 68;
```

causes 68 to be stored in the heating setpoint once at the start of the run only, even though tuTH will accept expressions with variability up to hourly. Furthermore, constant inner portions of variable expressions are pre-evaluated before the run begins.

CSE statements and expressions do not (yet) have user-settable variables in the usual programming language sense. They do, however, have user-defined functions to facilitate using the same computation several places, and preprocessor macros, to facilitate using the same text several places, specifying parametric values in a separate file, etc.

3.3.4 The Preprocessor – Overview

The preprocessor scans and processes input file text before the language processor sees the text. The preprocessor can include (embed) additional files in the input, include sections of input conditionally, and define and expand macros.

Macros are a mechanism to substitute a specified text for each occurrence of a word (the macro name). For example,

```
#define ZNWID 20
#define ZNLEN 30
. . .

znArea = ZNWID * ZNLEN;
znVol  = ZNWID * ZNLEN * 8;
```

The first line above says that all following occurrences of "ZNWID" are to be replaced with "20" (or whatever follows ZNWID on the same line). The effect of the above is that the zone width and length are specified only one place; if the single numbers are editing, both the zone area and zone volume change to match.

Macros can be especially powerful when combined with the file inclusion feature; the generic building description could be in one file, and the specific values for multiple runs supplied by another file. By also using conditional compilation, the values-specifying file can select from a range of features available in the building description file.

The preprocessor is similar to that of the C programming language, and thus will be familiar to C programmers.

The next section describes the preprocessor in detail. The preprocessor description is followed by sections detailing statements, then expressions.

3.4 The Preprocessor

Note: The organization and wording of this section is based on section A12 of Kernigan and Richie [1988]. The reader is referred to that source for a somewhat more rigorous presentation but with the caution that the CSE input language preprocessor does not ***completely*** *comply to ANSI C specifications.*

The preprocessor performs macro definition and expansion, file inclusion, and conditional inclusion/exclusion of text. Lines whose first non-whitespace character is `#` communicate with the preprocessor and are designated *preprocessor directives*. Line boundaries are significant to the preprocessor (in contrast to the rest of the input language in which a newline is simply whitespace), although adjacent lines can be spliced with \, as discussed below. The syntax of preprocessor directives is separate from that of the rest of the language. Preprocessor directives can appear anywhere in an input file and their effects last until the end of the input file. The directives that are supported by the input language preprocessor are the following:

```
#if
#else
#elif
#endif
#ifndef

#define
#redefine
#undef

#include
```

3.4.1 Line splicing

If the last character on a line is the backslash \, then the next line is spliced to that line by elimination of the backslash and the following newline. Line splicing occurs *before* the line is divided into tokens.

Line splicing finds its main use in defining long macros:

```
// hourly light gain values:
#define LIGHT_GAIN       .024, .022, .021, .021, .021, .026, \
```

```
.038, .059, .056, .060, .059, .046, \
.045, .5  , .5  , .05 , .057, .064, \
.064, .052, .050, .055, .044, .027
```

3.4.2 Macro definition and expansion

A directive of the form

```
#define _identifier_ _token-sequence_
```

is a macro definition and causes the preprocessor to replace subsequent instances of the identifier with the given token sequence. Note that the token string can be empty (e.g. `#define FLAG`).

A line of the form

```
#define _identifier_( _identifier-list_) _token-sequence_
```

where there is no space between the identifier and the (, is a macro with parameters given by the identifier list. The expansion of macros with parameters is discussed below.

Macros may also be defined *on the CSE command line*, making it possible to vary a run without changing the input files at all. As described in the command line section, macros are defined on the CSE command line using the `-D` switch in the forms

```
-D_identifier_

-D_identifier_=_token-sequence_
```

The first form simply defines the name with no token-sequence; this is convenient for testing with `#ifdef`, `#ifndef`, or `defined()`, as described in the section on conditional inclusion of tex. The second form allows an argument list and token sequence. The entire command line argument must be enclosed in quotes if it contains any spaces.

A macro definition is forgotten when an `#undef` directive is encountered:

```
#undef _identifier_
```

It is not an error to `#undef` an undefined identifier.

A macro may be re-`#defined` without a prior `#undef` unless the second definition is identical to the first. A combined `#undef`/`#define` directive is available to handle this common case:

```
#redefine _identifier_ _token-sequence_

#redefine _identifier_( _identifier-list_) _token-sequence_
```

When a macro is `#redefined`, it need not agree in form with the prior definition (that is, one can have parameters even if the other does not). It is not an error to `#redefine` an undefined identifier.

Macros defined in the second form (with parameters) are expanded whenever the preprocessor encounters the macro identifier followed by optional whitespace and a comma-separated parameter list enclosed in parentheses. First the comma separated token sequences are collected; any commas within quotes or nested parentheses do not separate parameters. Then each unquoted instance of the each parameter identifier in the macro definition is replaced by the collected tokens. The resulting string is then repeatedly re-scanned for more defined identifiers. The macro definition and reference must have the same number of arguments.

It is often important to include parentheses within macro definitions to make sure they evaluate properly in all situations. Suppose we define a handy area macro as follows:

```
#define AREA(w, h) w*h        // WRONG
```

Consider what happens when this macro is expanded with arguments 2+3 and 4+1. The preprocessor substitutes the arguments for the parameters, then the input language processor processes the statement containing the macro expansion without regard to the beginning and end of the arguments. The expected result is 25, but as defined, the macro will produce a result of 15. Parentheses fix it:

```
#define AREA(w, h) ((w)*(h))  // RIGHT
```

The outer enclosing set of parentheses are not strictly needed in our example, but are good practice to avoid evaluation errors when the macro expands within a larger expression.

Note 1: The CSE preprocessor does not support the ANSI C stringizing (#) or concatenation (##) operators.

Note 2: Identifiers are case *insensitive* (unlike ANSI C). For example, the text "myHeight" will be replaced by the `#defined` value of MYHEIGHT (if there is one).

The preprocessor examples at the end of this section illustrate macro definition and expansion.

3.4.3 File inclusion

Directives of the form

`#include "filename"` and

`#include <filename>`

cause the replacement of the directive line with the entire contents of the referenced file. If the filename does not include an extension, a default extension of .INP is assumed. The filename may include path information; if it does not, the file must be in the current directory.

`#include`s may be nested to a depth of 5.

For an example of the use `#include`s, please see the preprocessor examples at the end of this section.

3.4.4 Conditional inclusion of text

Conditional text inclusion provides a facility for selectively including or excluding groups of input file lines. The lines so included or excluded may be either CSE input language text *or other preprocessor directives*. The latter capability is very powerful.

Several conditional inclusion directive involve integer constant expressions. Constant integer expressions are formed according the rules discussed in the section on expressions with the following changes:

1. Only constant integer operands are allowed.
2. All values (including intermediate values computed during expression evaluation) must remain in the 16 bit range (-32768 - 32767). The expression processor treats all integers as signed values and requires signed decimal constants – however, it requires unsigned octal and hexadecimal constants. Thus decimal constants must be in the range -32768 - 32767, octal must be in the range 0 - 0o177777, and hexadecimal in the range 0 - 0xffff. Since all arithmetic comparisons are done assuming signed values, 0xffff < 1 is true (unhappily). Care is required when using the arithmetic comparison operators (<, <=, >=, >).
3. The logical relational operators && and || are not available. Nearly equivalent function can be obtained with & and |.
4. A special operand defined( ) is provided; it is described below.

Macro expansion *is* performed on constant expression text, so symbolic expressions can be used (see examples below).

The basic conditional format uses the directive

```
#if _constant-expression_
```

If the constant expression has the value 0, all lines following the `#if` are dropped from the input stream (the preprocessor discards them) until a matching `#else`, `#elif`, or `#endif` directive is encountered.

The defined(*identifier*) operand returns 1 if the identifier is the name of a defined macro, otherwise 0. Thus

```
#if defined( _identifier_ )
```

can be used to control text inclusion based on macro flags. Two `#if` variants that test whether a macro is defined are also available. `#ifdef` *identifier* is equivalent to `#if` defined(*identifier*) and `#ifndef` *identifier* is equivalent to `#if` !defined(*identifier*).

Defined(), `#ifdef`, and `#ifndef` consider a macro name "defined" even if the body of its definition contains no characters; thus a macro to be tested with one of these can be defined with just

```
#define _identifier_
```

or with just "-D*identifier*" on the CSE command line.

Conditional blocks are most simply terminated with `#endif`, but `#else` and `#elif` *constant-expression* are also available for selecting one of two or more alternative text blocks.

The simplest use of `#if` is to "turn off" sections of an input file without editing them out:

```
#if 0This text is deleted from the input stream.#endif
```

Or, portions of the input file can be conditionally selected:

```
#define FLRAREA 1000   // other values used in other runs
#if FLRAREA <= 800
    CSE input language for small zones
#elif FLRAREA <= 1500
    CSE input language for medium zones
#else
    CSE input language for large zones
#endif
```

Note that if a set of `#if` … `#elif` … `#elif` conditionals does not contain an `#else`, it is possible for all lines to be excluded.

Finally, it is once again important to note that conditional directives *nest*, as shown in the following example (indentation is included for clarity only):

```
#if 0
    This text is NOT included.
    #if 1
        This text is NOT included.
    #endif
#else
    This text IS included.
#endif
```

3.4.5 Preprocessor examples

This section shows a few combined examples that demonstrate the preprocessor's capabilities.

The simplest use of macros is for run parameterization. For example, a base file is constructed that derives values from a macro named FLRAREA. Then multiple runs can be performed using `#include`:

```
// Base file
... various input language statements ...

ZONE main
    znArea = FLRAREA
```

```
    znVol  = 8*FLRAREA
    znCAir = 2*FLRAREA ...
    ... various other input language statements ...

RUN

CLEAR
```

The actual input file would look like this:

```
// Run with zone area = 500, 1000, and 2000 ft2
#define FLRAREA 500
#include "base."
#redefine FLRAREA 1000
#include "base."
#redefine FLRAREA 2000
#include "base."
```

Macros are also useful for encapsulating standard calculations. For example, most U-values must be entered *without* surface conductances, yet many tabulated U-values include the effects of the standard ASHRAE winter surface conductance of 6.00 Btuh/ft^2-°F. A simple macro is very helpful:

```
#define UWinter(u)  ( 1/(1/(u)-1/6.00) )
```

This macro can be used whenever a U-value is required (e.g. SURFACE … sfU=UWinter(.11) …).

3.5 CSE Input Language Statements

This section describes the general form of CSE input language statements that define objects, assign values to the data members of objects, and initiate actions. The concepts of objects and the class hierarchy were introduced in the section on form of CSE data. Information on statements for specific CSE input language classes and their members is the subject of the input data section.

3.5.1 Object Statements

As we described in a previous section, the description of an object is introduced by a statement containing at least the class name, and usually your chosen name for the particular object. In addition, this section will describe several optional qualifiers and modifying clauses that permit defining similar objects without repeating all of the member details, and reopening a previously given object description to change or add to it.

Examples of the basic object-beginning statement:

```
ZONE "North";

METER "Electric - Cooling";

LAYER;
```

As described in the section on nested objects, such a statement is followed by statements giving the object's member values or describing subobjects of the object. The object description ends when you begin another object that is not of a subclass of the object, or when a member of an embedding (higher level) object previously begun is given, or when END is given.

3.5.1.1 Object Names

An object name consists of up to 63 characters. If you always enclose the name in quotation marks, punctuation and spaces may be used freely; if the name starts with a letter or dollar sign and consists only of letters, digits, underscore, and dollar sign, and is different from all of the words already defined in CSE input language (as listed below in this section), you may omit the quotes. Capitalization, and Leading and trailing spaces and tabs, are always disregarded by input language processor. Names of 0 length, and names containing control characters (ASCII codes 0-31) are not allowed.

Examples of valid names that do not require quotes:

```
North
gas_meter
slab140E
```

The following object names are acceptable if always enclosed in quotes:

```
"Front Door"
"M L King Day"
"123"
"3.5-inch wall"
```

We suggest always quoting object names so you won't have to worry about disallowed words and characters.

Duplicate names result in error messages. Object names must be distinct between objects of the same class which are subobjects of the same object. For example, all ZONE names must be distinct, since all ZONEs are subobjects of Top. It is permissible to have SURFACEs with the same name in different ZONEs – but it is a good idea to keep all of your object names distinct to minimize the chance of an accidental mismatch or a confusing message regarding some other error.

For some classes, such as ZONE, a name is required for each object. This is because several other statements refer to specific ZONEs, and because a name is needed to identify ZONEs in reports. For other classes, the name is optional. The specific statement descriptions in the Input Data Section 5 say which names are required. We suggest always using object names even where not required; one reason is because they allow CSE to issue clearer error messages.

The following *reserved words will not work as object names unless enclosed in quotes*:

(this list needs to be assembled and typed in)

3.5.1.2 ALTER

ALTER is used to reopen a previously defined object when it is not possible or desired to give the entire description contiguously.

ALTER could be used if you wish to order the input in a special way. For example, SURFACE objects are subobjects of ZONE and are normally described with the ZONE they are part of. However, if you wanted to put all roofs together, you could use input of the general form:

```
ZONE "1";  . . .  (zone 1 description)
ZONE "2";  . . .
. . .
ALTER ZONE "1";              // revert to specifying zone 1
    SURFACE "Roof1";  . . .  (describe roof of zone 1)
ALTER ZONE "2";
    SURFACE "Roof2";  . . .
```

ALTER can be used to facilitate making similar runs. For example, to evaluate the effect of a change in the size of a window, you might use:

```
ZONE "South";
    SURFACE "SouthWall";
    ...
```

```
            WINDOW "BigWindow";
                wnHeight = 6;  wnWidth = 20;
    . . .
    RUN;          // perform simulation and generate reports
    // data from simulation is still present unless CLEAR given
    ALTER ZONE "South";
        ALTER SURFACE "SouthWall";
            ALTER WINDOW "BigWindow";
                wnHeight = 4;  wnWidth = 12;  // make window smaller
    RUN;          // perform simulation and print reports again
```

ALTER also lets you access the predefined "Primary" REPORTFILE and EXPORTFILE objects which will be described in the Input Data Section:

```
    ALTER REPORTFILE "Primary";    /* open description of object automatically
                                      supplied by CSE -- no other way to access */
        rfPageFmt = NO;            /* Turn off page headers and footers --
                                      not desired when reports are to be
                                      reviewed on screen. */
```

3.5.1.3 DELETE

DELETE followed by a class name and an object name removes the specified object, and any subobjects it has. You might do this after RUN when changing the data for a similar run (but to remove all data, CLEAR is handier), or you might use DELETE after COPYing (below) an object if the intent is to copy all but certain subobjects.

3.5.1.4 LIKE clause

LIKE lets you specify that an object being defined starts with the same member values as another object already defined. You then need give only those members that are different. For Example:

```
    MATERIAL "SheetRock";         // half inch gypsum board
        matCond = .0925;          // conductivity per foot
        matSpHt = .26;            // specific heat
        matDens = 50;             // density
        matThk = 0'0.5;           // thickness 1/2 inch
    MATERIAL "5/8 SheetRock" LIKE "SheetRock"; // 5/8" gypsum board
        matThk = 0'0.625;         // thickness 5/8 inch
        // other members same as "SheetRock", need not be repeated
```

The object named after LIKE must be already defined and must be of the same class as the new object.

LIKE copies only the member values; it does not copy any subobjects of the prototype object. For example, LIKEing a ZONE to a previously defined ZONE does not cause the new zone to contain the surfaces of the prototype ZONE. If you want to duplicate the surfaces, use COPY instead of LIKE.

3.5.1.5 COPY clause

COPY lets you specify that the object being defined is the same as a previously defined object including all of the subobjects of that object. For example,

```
    . . .
    ZONE "West" COPY "North";
        DELETE WALL "East";
        ALTER WALL "South";
            sfExCnd = ambient;
```

Specifies a ZONE named "West" which is the same as ZONE North except that it does not contain a copy of West's East wall, and the South wall has ambient exposure.

3.5.1.6 USETYPE clause

USETYPE followed by the type name is used in creating an object of a type previously defined with DEFTYPE (next section). Example:

```
SURFACE "EastWall" USETYPE "IntWall";     // use interior wall TYPE (below)
    sfAzm = 90;                           // this wall faces to the East
    sfArea = 8 * 30;                      // area of each wall is different
    sfAdjZn = "East";                     // zone on other side of wall
```

Any differences from the type, and any required information not given in the type, must then be specified. Any member specified in the type may be respecified in the object unless FROZEN (see this section) in the type (normally, a duplicate specification for a member results in an error message).

3.5.1.7 DEFTYPE

DEFTYPE is used to begin defining a TYPE for a class. When a TYPE is created, no object is created; rather, a partial or complete object description is stored for later use with DEFTYPE. TYPES facilitate creating multiple similar objects, as well as storing commonly used descriptions in a file to be #included in several different files, or to be altered for multiple runs in comparative studies without changing the including files. Example (boldface for emphasis only):

```
DEFTYPE SURFACE "BaseWall"                // common characteristics of all walls
    sfType = WALL;                        // walls are walls, so say it once
    sfTilt = 90;                          // all our walls are vertical;
                                          //  but sfAzm varies, so it is not in TYPE.
    sfU = .83;                            // surf conductance; override if different
    sfModel = QUICK;

DEFTYPE SURFACE "ExtWall" USETYPE "BaseWall";
    sfExCnd = AMBIENT;                    // other side of wall is outdoors
    sfExAbs = 0.5;                        // member only needed for exterior walls

DEFTYPE SURFACE "IntWall" USETYPE "BaseWall";   // interior wall
    sfExCnd = ADJZN;                      // user must give sfAdjZn.
```

In a TYPE as much or as little of the description as desired may be given. Omitting normally-required members does not result in an error message in the type definition, though of course an error will occur at use if the member is not given there.

At use, member values specified in the TYPE can normally be re specified freely; to prevent this, "freeze" the desired member(s) in the type definition with

```
FREEZE *memberName*;
```

Alternately, if you wish to be sure the user of the TYPE enters a particular member even if it is normally optional, use

```
REQUIRE *memberName*
```

Sometimes in the TYPE definition, member(s) that you do not want defined are defined – for example, if the TYPE definition were itself initiated with a statement containing LIKE, COPY, or USETYPE. In such cases the member specification can be removed with

```
UNSET *memberName*;
```

3.5.1.8 END and ENDxxxx

END, optionally followed by an object name, can be used to unequivocally terminate an object. Further, as of July 1992 there is still available a specific word to terminate each type of object, such as ENDZONE to terminate a ZONE object. If the object name is given after END or ENDxxxx, an additional check is performed: if the name is not that of an object which has been begun and not terminated, an error message occurs. Generally, we have found it is not important to use END or ENDxxxx, especially since the member names in different classes are distinct.

3.5.2 Member Statements

As introduced in the section on statements, statements which assign values to members are of the general form:

```
*memberName* = *expression*;
```

The specific member names for each class of objects are given in Section 5; many have already been shown in examples.

Depending on the member, the appropriate type for the expression giving the member value may be numeric (integer or floating point), string, object name, or multiple-choice. Expressions of all types will be described in detail in the section on expressions.

Each member also has its *variability* (also given in the input data section), or maximum acceptable *variation*. This is how often the expression for the value can change during the simulation – hourly, daily, monthly, no change (constant), etc. The "variations" were introduced in the expressions overview section and will be further detailed in a section on variation frequencies.

Three special statements, UNSET, REQUIRE, and FREEZE, add flexibility in working with members.

3.5.2.1 UNSET

UNSET followed by a member name is used when it is desired to delete a member value previously given. UNSETing a member resets the object to the same internal state it was in before the member was originally given. This makes it legal to specify a new value for the member (normally, a duplicate specification results in an error message); if the member is required (as specified in the input data section), then an error message will occur if RUN is given without re specifying the member.

Situations where you really might want to specify a member, then later remove it, include:

- After a RUN command has completed one simulation run, if you wish to specify another simulation run without CLEARing and giving all the data again, you may need to UNSET some members of some objects in order to re specify them or because they need to be omitted from the new run. In this case, use ALTER(s) to reopen the object(s) before UNSETing.
- In defining a TYPE (see this section), you may wish to make sure certain members are not specified so that the user must give them or omit them if desired. If the origin of the type (possibly a sequence of DEFTYPEs, LIKEs, and/or COPYs) has defined unwanted members, get rid of them with UNSET.

Note that UNSET is only for deleting *members* (names that would be followed with an = and a a value when being defined). To delete an entire *object*, use DELETE (see this section).

3.5.2.2 REQUIRE

REQUIRE followed by a member name makes entry of that member mandatory if it was otherwise optional; it is useful in defining a TYPE (see this section) when you desire to make sure the user enters a particular member, for example to be sure the TYPE is applied in the intended manner. REQUIRE by itself does not delete any previously entered value, so if the member already has a value, you will need to UNSET it. ?? *verify*

3.5.2.3 FREEZE

FREEZE followed by a member name makes it illegal to UNSET or redefine that member of the object. Note that FREEZE is unnecessary most of the time since CSE issues an error message for duplicate definitions without an intervening UNSET, unless the original definition came from a TYPE (see this section). Situations where you might want to FREEZE one or more members include:

- When defining a TYPE (see this section). Normally, the member values in a type are like defaults; they can be freely overridden by member specifications at each use. If you wish to insure a TYPE is used as intended, you may wish to FREEZE members to prevent accidental misuse.
- When your are defining objects for later use or for somebody else to use (perhaps in a file to be included) and you wish to guard against misuse, you may wish to FREEZE members. Of course, this is not foolproof, since there is at present no way to allow use of predefined objects or types without allowing access to the statements defining them.

3.5.3 Action Commands

CSE has two action commands, RUN and CLEAR.

3.5.3.1 RUN

RUN tells `CSE` to do an hourly simulation with the data now in memory, that is, the data given in the preceding part of the input file.

Note that CSE does NOT automatically run the simulator; an input file containing no RUN results in no simulation (you might nevertheless wish to submit an incomplete file to CSE to check for errors in the data already entered). The explicit RUN command also makes it possible to do multiple simulation runs in one session using a single input file.

When RUN is encountered in the input file, CSE checks the data. Many error messages involving inconsistencies between member values or missing required members occur at this time. If the data is good, CSE starts the simulation. When the simulation is complete and the reports have been output, CSE continues reading the input file. Statements after the first run can add to or change the data in preparation for another RUN. Note that the data for the first run is NOT automatically removed; if you wish to start over with complete specifications, use CLEAR after RUN.

3.5.3.2 CLEAR

CLEAR removes all input data (objects and all their members) from CSE memory. CLEAR is normally used after RUN, when you wish to perform another simulation run and wish to start clean. If CLEAR is not used, the objects from the prior run's input remain in memory and may be changed or added to produce the input data for the next simulation run.

3.6 Expressions

Probably the CSE input language's most powerful characteristic is its ability to accept expressions anywhere a single number, string, object name, or other value would be accepted. Preceding examples have shown the inputting zone areas and volumes as numbers (some defined via preprocessor macros) with *'s between them to signify multiplication, to facilitate changes and avoid errors that might occur in manual arithmetic. Such expressions, where all operands are constants, are acceptable *anywhere* a constant of the same type would be allowed.

But for many object members, CSE accepts *live expressions* that *vary* according to time of day, weather, zone temperatures, etc. (etc., etc., etc.!). Live expressions permit simulation of many relationships without special-purpose features in the language. Live expressions support controlling setpoints, scheduling HVAC system operation, resetting air handler supply temperature according to outdoor temperature, and other necessary and foreseen functions without dedicated language features; they will also support many unforeseen user-generated functionalities that would otherwise be unavailable.

Additional expression flexibility is provided by the ability to access all of the input data and much of the internal data as operands in expressions (*probes*, see this section).

As in a programming language, CSE expressions are constructed from operators and operands; unlike most programming languages, CSE determines how often an expression's operands change and automatically compute and store the value as often as necessary.

Expressions in which all operands are known when the statement is being decoded (for example, if all values are constants) are *always* allowed, because the input language processor immediately evaluates them and presents the value to the rest of the program in the same manner as if a single number had been entered. *Most* members also accept expressions that can be evaluated as soon as the run's input is complete, for example expressions involving a reference to another member that has not been given yet. Expressions that vary during the run, say at hourly or daily intervals, are accepted by *many* members. The *variability* or maximum acceptable variation for each member is given in the descriptions in the input data section, and the *variation* of each non-constant expression component is given in its description in this section.

Interaction of expressions and the preprocessor: Generally, they don't interact. The preprocessor is a text processor which completes its work by including specified files, deleting sections under false #if's, remembering define definitions, replacing macro calls with the text of the definition, removing preprocessor directives from the text after interpreting them, etc., *then* the resulting character stream is analyzed by the input language statement compiler. However, the if statement takes an integer numeric expression argument. This expression is similar to those described here except that it can only use constant operands, since the preprocessor must evaluate it before deciding what text to feed to the input statement statement compiler.

3.6.1 Expression Types

The type of value to which an expression must evaluate is specified in each member description (see the input data section) or other context in which an expression can be used. Each expression may be a single constant or may be made up of operators and operands described in the rest of this section, so long as the result is the required type or can be converted to that type by CSE, and its variation is not too great for the context. The possible types are:

float	A real number (3.0, 5.34, -2., etc.). Approximately 7 digits are carried internally. If an int is given where a real is required, it is automatically converted.
int	An integer or whole number (-1, 0, 1, 2 etc.). If a real is given, an error may result, but we should change it to convert it (discarding any fractional part).
Boolean	Same as int; indicates that a 0 value will be interpreted as "false" and any non-0 value will be interpreted as "true".
string	A string of characters; for example, some text enclosed in quotes.
object name	Name of an object of a specified class. Differs from *string* in that the name need not be enclosed in quotes if it consists only of letters, digits, _, and $, begins with a non-digit, and is different from all reserved words now in or later added to the language (see Object Names). The object may be defined after it is referred to. An expression using conditional operators, functions, etc. may be used provided its value is known when the RUN action command is reached.; no members requiring object names accept values that vary during the simulation.

choice	One of several choices; a list of the acceptable values is given wherever a *choice* is required. The choices are usually listed in CAPITALS but may be entered in upper or lower case as desired. As with object names, quotes are allowed but not required. Expressions may be used for choices, subject to the variability of the context.

date	May be entered as a 3-letter month abbreviation followed by an *int* for the day of the month, or an *int* for the Julian day of the year (February is assumed to have 28 days). Expressions may be used subject to variability limitations. Examples:
	`Jan 23   // January 23`
	`23       // January 23`
	`32       // February 1`

These words are used in following descriptions of contexts that can accept more than one basic type:

numeric	*float* or *int*. When floats and ints are intermixed with the same operator or function, the result is float.
anyType	Any type; the result is the same type as the argument. If floats and ints are intermixed, the result is float. If strings and valid choice names are intermixed, the result is *choice*. Other mixtures of types are generally illegal, except in expressions for a few members that will accept either one of several choices or a numeric value.

The next section describes the syntax of constants of the various data types; then, we will describe the available operators, then other operand types such as system variables and built-in functions.

3.6.2 Constants

This section reviews how to enter ordinary non-varying numbers and other values.

int	optional - sign followed by digits. Don't use a decimal point if your intent is to give an *int* quantity – the decimal point indicates a *float* to CSE. Hexadecimal and Octal values may be given by prefixing the value with 0x and 0O respectively (yes, that really is a zero followed by an 'O').
float	optional - sign, digits and decimal point. Very large or small values can be entered by following the number with an "e" and a power of ten. Examples; `1.0  1.  .1  -5534.6  123.e25  4.56e-23` The decimal point indicates a float as opposed to an int. Generally it doesn't matter as CSE converts ints to floats as required, but be careful when dividing: CSE interprets "2/3" as integer two divided by integer 3, which will produce an integer 0 before CSE notices any need to convert to *float*. If you mean .6666667, say 2./3, 2/3., or .6666667.
feet and inches	Feet and inches may be entered where a *float* number of feet is required by typing the feet (or a 0 if none), a single quote ', then the inches. (Actually this is an operator meaning "divide the following value by 12 and add it to the preceding value", so expressions can work with it.) Examples: `3'6  0'.5 (10+20)'(2+3)`

string	"Text" – desired characters enclosed in double quotes. Maximum length 80 characters (make 132??). To put a " within the "'s, precede it with a backslash. Certain control codes can be represented with letters preceded with a backslash as follows:
	`\\e` `escape`
	`\\t` `tab`
	`\\f` `form feed`
	`\\r` `carriage return`
	`\\n` `newline or line feed`

object name	Same as *string*, or without quotes if name consists only of letters, digits, _, and $, begins with a non-digit, and is different from all reserved words now in or later added to the language (see Object Names). Control character codes (ASCII 0-31) are not allowed.
choice	Same as string; quotes optional on choice words valid for the member. Capitalization does not matter.
date	Julian day of year (as *int* constant), or month abbreviation `Jan Feb Mar Apr May Jun Jul Aug Sep Oct Nov De c` followed by the *int* day of month. (Actually, the month names are operators implemented to add the starting day of the month to the following *int* quantity).

3.6.3 Operators

For *floats* and *ints*, the CSE input language recognizes a set of operators based closely on those found in the C programming language. The following table describes the available numeric operators. The operators are shown in the order of execution (precedence) when no ()'s are used to control the order of evaluation; thin lines separate operators of equal precedence.

Operator	Name	Notes and Examples
'	Feet-Inches Separator	a ' b yields a + b/12; thus 4'6 = 4.5.
+	Unary plus	The familiar "positive", as in +3. Does nothing; rarely used.
-	Unary minus	The familiar "minus", as in -3. -(-3) = +3 etc.
!	Logical NOT	Changes 0 to 1 and any non-0 value to 0. !0 = 1, !17 = 0.
~	One's complement	Complements each bit in an *int* value.
*	Multiplication	Multiplication, e.g. 3*4 = 12; 3.24*18.54 = 60.07
/	Division	Division, e.g. 4/2 = 2, 3.24/1.42 = 2.28. Integer division truncates toward 0 (e.g. 3/2 = 1, 3/-2 = -1, -3/2 = -1, 2/3 = 0) CAUTION!
%	Modulus	Yields the remainder after division, e.g. 7%2 = 1. The result has the same sign as the left operand (e.g.(-7)%2 = -1). % is defined for both integer and floating point operands (unlike ANSI C).
+	Addition	Yields the sum of the operands, e.g. 5+3 = 8
-	Subtraction	Yields the difference of the operands, e.g. 5-3 = 2
>>	Right shift	a >> b yields a shifted right b bit positions, e.g. 8>>2 = 2

<table>
<tr><th>Operator</th><th>Name</th><th>Notes and Examples</th></tr>
<tr><td><<</td><td>Left shift</td><td>a << b yields a shifted left b bit positions, e.g. 8<<2 = 32</td></tr>
<tr><td><</td><td>Less than</td><td>a < b yields 1 if a is less than b, otherwise 0</td></tr>
<tr><td><=</td><td>Less than or equal</td><td>a <= b yields 1 if a is less than or equal to b, otherwise 0</td></tr>
<tr><td>>=</td><td>Greater than or equal</td><td>a >= b yields 1 if a is greater than or equal to b, otherwise 0</td></tr>
<tr><td>></td><td>Greater than</td><td>a > b yields 1 if a is greater than b, otherwise 0</td></tr>
<tr><td>==</td><td>Equal</td><td>a == b yields 1 if a is exactly (bit wise) equal to b, otherwise 0</td></tr>
<tr><td>!=</td><td>Not equal</td><td>a != b yields 1 if a is not equal to b, otherwise 0</td></tr>
<tr><td>&</td><td>Bitwise and</td><td>a & b yields the bitwise AND of the operands, e.g. 6 & 2 = 2.</td></tr>
<tr><td>ˆ</td><td>Bitwise exclusive or</td><td>a ˆ b yields the bitwise XOR of the operands, e.g. 6 ˆ 2 = 4.</td></tr>
<tr><td>|</td><td>Bitwise inclusive or</td><td>a | b yields the bitwise IOR of the operands, e.g. 6 | 2 = 6.</td></tr>
<tr><td>&&</td><td>Logical AND</td><td>a && b yields 1 if both a and b are non-zero, otherwise 0. && guarantees left to right evaluation: if the first operand evaluates to 0, the second operand is not evaluated and the result is 0.</td></tr>
<tr><td>||</td><td>Logical OR</td><td>a || b yields 1 if either a or b is true (non-0), otherwise 0. || guarantees left to right evaluation: if the first operand evaluates to non-zero, the second operand in not evaluated and the result is 1.</td></tr>
<tr><td>? :</td><td>Conditional</td><td>a ? b : c yields b if a is true (non-0), otherwise c.</td></tr>
</table>

Dates are stored as *ints* (the value being the Julian day of the year), so all numeric operators could be used. The month abbreviations are implemented as operators that add the first day of the month to the following *int* value; CSE does not disallow their use in other numeric contexts.

For *strings*, *object names*, and *choices*, the CSE input language currently has no operators except the ?: conditional operator. A concatenation operator is being considered. Note, though, that the choose, choose1, select, and hourval functions described below work with strings, object names, and choice values as well as numbers.

3.6.4 System Variables

System Variables are built-in operands with useful values. To avoid confusion with other words, they begin with a $. Descriptions of the CSE system variables follow. Capitalization shown need not be matched. Most system variables change during a simulation run, resulting in the *variations* shown; they cannot be used where the context will not accept variation at least this fast. (The Input Data Section gives the *variability*, or maximum acceptable variation, for each object member.)

$dayOfYear	Day of year of simulation, 1 - 365; 1 corresponds to Jan-1. (Note that this is not the day of the simulation unless begDay is Jan-1.) **Variation:** daily.

$month	Month of year, 1 - 12. **Variation**: monthly.
$dayOfMonth	Day of month, 1 - 31. **Variation**: daily.
$hour	Hour of day, 1 - 24; 1 corresponds to midnight - 1 AM. **Variation**: hourly.
$dayOfWeek	Day of week, 1 - 7; 1 corresponds to Sunday, 2 to Monday, etc. **Variation:** daily.
$DOWH	Day of week 1-7 except 8 on every observed holiday. **Variation**: daily.
$isHoliday	1 on days that a holiday is observed (regardless of the true date of the holiday); 0 on other days. **Variation**: daily.
$isHoliTrue	1 on days that are the true date of a holiday, otherwise 0. **Variation**: daily.
$isWeHol	1 on weekend days or days that are observed as holidays. **Variation:** daily.
$isWeekend	1 on Saturday and Sunday, 0 on any day from Monday to Friday. **Variation:** daily.
$isWeekday	1 on Monday through Friday, 0 on Saturday and Sunday. **Variation:** daily.
$isBegWeek	1 for any day immediately following a weekend day or observed holiday that is neither a weekend day or an observed holiday. **Variation:** daily.
$isWorkDay	1 on non-holiday Monday through Friday, 0 on holidays, Saturday and Sunday. **Variation:** daily.
$isNonWorkDay	1 on Saturday, Sunday and observed holidays, 0 on non-holiday Monday through Friday. **Variation:** daily.
$isBegWorkWeek	1 on the first workday after a non-workday, 0 all other days. **Variation:** daily.
$isDT	1 if Daylight Saving time is in effect, 0 otherwise. **Variation:** hourly.
$autoSizing	1 during autosizing calculations, 0 during main simulation. **Variation:** for each phase.
$dsDay	Design day type, 0 during main simulation, 1 during heating autosize, 2 during cool autosize. **Variation:** daily.

Weather variables: the following allow access to the current hour's weather conditions in you CSE expressions. Units of measure are shown in parentheses. All have **Variation**: hourly.

$radBeam	Solar beam irradiance (on a sun-tracking surface) this hour (Btu/ft2)
$radDiff	Solar diffuse irradiance (on horizontal surface) this hour (Btu/ft2)
$tDbO	Outdoor drybulb temperature this hour (degrees F)
$tWbO	Outdoor wetbulb temperature this hour (degrees F)
$wO	Outdoor humidity ratio this hour (lb H2O/lb dry air)
$windDirDeg	Wind direction (compass degrees)
$windSpeed	Wind speed (mph)

3.6.5 Built-in Functions

Built-in functions perform a number of useful scheduling and conditional operations in expressions. Built-in functions have the combined variation of their arguments; for *hourval*, the minimum result variation is hourly. For definitions of *numeric* and *anyType*, see Expression Types.

3.6.5.1 brkt

Function	limits a value to be in a given range
Syntax	*numeric* **brkt**(*numeric min, numeric val, numeric max*)

Remark	If *val* is less than *min*, returns *min*; if *val* is greater than *max,* returns *max*; if *val* is in between, returns *val*.
Example	In an AIRHANDLER object, the following statement would specify a supply temperature equal to 130 minus the outdoor air temperature, but not less than 55 nor greater than 80: `ahTsSp = brkt( 55, 130 - $tDbO, 80);` This would produce a 55-degree setpoint in hot weather, an 80-degree setpoint in cold weather, and a transition from 55 to 70 as the outdoor temperature moved from 75 to 50.

3.6.5.2 fix

Function	converts *float* to *int*
Syntax	*int* **fix**(*float val*)
Remark	*val* is converted to *int* by truncation – **fix**(1.3) and **fix**(1.99) both return 1. **fix**(-4.4) returns -4.

3.6.5.3 toFloat

Function	converts *int* to *float*
Syntax	*float* **toFloat**(*int val*)

3.6.5.4 min

Function	returns the lowest quantity from a list of values.
Syntax	*numeric* **min**(*numeric value1, numeric value2, … numeric valuen*)
Remark	there can be any number of arguments separated by commas; if floats and ints are intermixed, the result is float.

3.6.5.5 max

Function	returns the highest quantity from a list of values.
Syntax	*numeric* **max** (*numeric value1, numeric value2, … numeric valuen*)

3.6.5.6 choose

Function	returns the nth value from a list. If *arg0* is 0, *value0* is returned; for 1, *value1* is returned, etc.
Syntax	*anyType* **choose** (*int arg0, anyType value0, anyType value1, … anyType valuen*) or *anyType* **choose** (*int arg0, anyType value0, … anyType valuen*, **default** *valueDef*)
Remarks	Any number of *value* arguments may be given. If **default** and another value is given, this value will be used if *arg0* is less than 0 or too large; otherwise, an error will occur.

3.6.5.7 choose1

Function	same as choose except *arg0* is 1-based. Choose1 returns the second argument *value1* for *arg0* = 1, the third argument *value2* when *arg0* = 2, etc.
Syntax	*anyType* **choose1** (*int arg0, anyType value1, anyType value2, … anyType valuen*) or *anyType* **choose1** (*int arg0, anyType value1, … anyType valuen,* **default** *valueDef*)
Remarks	**choose1** is a function that is well suited for use with daily system variables. For example, if a user wanted to denote different values for different days of the week, the following use of **choose1** could be implemented: `tuTC = choose1(\$dayOfWeek, MonTemp, TueTemp, ...)` Note that for hourly data, the **hourval** function would be a better choice, because it doesn't require the explicit declaration of the $hour system variable.

3.6.5.8 select

Function

contains Boolean-value pairs; returns the value associated with the first Boolean that evaluates to true (non-0).

Syntax

anyType (*Boolean arg1, anyType value1, Boolean arg2, anyType value2, …* **default** *anyType*) (the **default** part is optional)

Remark

select is a function that simulates if-then logic during simulation (for people familiar with C, it works much like a series of imbedded conditionals: (a?b:(a?b:c))).

Examples

Select can be used to simulate a **dynamic** (run-time) **if-else statement**:

```
gnPower = select( $isHoliday, HD_GAIN,  // if ($isHolida y)
default WD_GAIN)  // else
```

This technique can be combined with other functions to schedule items on a hourly and daily basis. For example, an internal gain that has different schedules for holidays, weekdays, and weekends could be defined as follows:

```
// 24-hour lighting power schedules for weekend, weekday ,
holiday:
#define WE_LIGHT  hourval( .024, .022, .021, .021, .021, . 026, \
        .038, .059, .056, .060, .059, .046, \
        .045, .005, .005, .005, .057, .064, \
        .064, .052, .050, .055, .044, .027 )
#define WD_LIGHT  hourval( .024, .022, .021, .021, .021, . 026, \
        .038, .059, .056, .060, .059, .046, \
        .045, .005, .005, .005, .057, .064, \
        .064, .052, .050, .055, .044, .027 )
#define HD_LIGHT  hourval( .024, .022, .021, .021, .021, . 026, \
        .038, .059, .056, .060, .059, .046, \
        .045, .005, .500, .005, .057, .064, \
        .064, .052, .050, .055, .044, .027 )
// set power member of zone's GAIN object for lighting
gnPower = BTU_Elec( ZAREA*0.1 ) *          // .1 kW/ft2 ti mes...
   select( $isHoliday, HD_LIGHT,   // Holidays
     $isWeekend, WE_LIGHT,   // Saturday & Sunday
     default     WD_LIGHT ); // Week Days
```

In the above, three subexpressions using **hourval** (next) are first defined as macros, for ease of reading and later change. Then, gnPower (the power member of a GAIN object) is set, using **select** to choose the appropriate one of the three **hourval** calls for the type of day. The expression for gnPower is a *live expression* with hourly variation, that is, CSE will evaluate it an set gnPower to the latest value each hour of the simulation. The variation comes from **hourval**, which varies hourly (also, $isHoliday and $isWeekend vary daily, but the faster variation determines the variation of the result).

3.6.5.9 hourval

Function	from a list of 24 values, returns the value corresponding to the hour of day.
Syntax	*anyType hourval (anyType value1, anyType value2, … anyType value24)* *anyType hourval (anyType value1, anyType value2*, … **default** *anyType*)
Remark	**hourval** is evaluated at runtime and uses the hour of the day being simulated to choose the corresponding value from the 24 suppplied values. If less than 24 *value* arguments are given, **default** and another value (or expression) should be supplied to be used for hours not explicitly specified.
Example	see **select**, just above.

3.6.5.10 abs

Function	converts numeric to its absolute value
Syntax	numeric **abs**(numeric val)

3.6.5.11 sqrt

Function	Calculates and returns the positive square root of *val* (*val* must be ≥ 0).
Syntax	*float* **sqrt** (*float val*)

3.6.5.12 exp

Function	Calculates and returns the exponential of *val* ($= e^{val}$)
Syntax	*float* **exp**(*float val*)

3.6.5.13 logE

Function	Calculates and returns the base e logarithm of *val* (*val* must be ≥ 0).
Syntax	*float* **logE**(*float val*)

3.6.5.14 log10

Function	Calculates and returns the base 10 logarithm of *val* (*val* must be ≥ 0).
Syntax	*float* **log10**(*float val*)

3.6.5.15 sin

Function	Calculates and returns the sine of *val* (val in radians)
Syntax	*float* **sin**(*float val*)

3.6.5.16 sind

Function	Calculates and returns the sine of *val* (val in degrees)
Syntax	*float* **sind**(*float val*)

3.6.5.17 asin

Function	Calculates and returns (in radians) the arcsine of *val*
Syntax	*float* **asin**(*float val*)

3.6.5.18 asind

Function	Calculates and returns (in degrees) the arcsine of *val*
Syntax	*float* **asind**(*float val*)

3.6.5.19 cos

Function	Calculates and returns the cosine of *val* (val in radians)
Syntax	*float* **cos**(*float val*)

3.6.5.20 cosd

Function	Calculates and returns the cosine of *val* (val in degrees)
Syntax	*float* **cosd**(*float val*)

3.6.5.21 acos

Function	Calculates and returns (in radians) the arccosine of *val*
Syntax	*float* **acos**(*float val*)

3.6.5.22 acosd

Function	Calculates and returns (in degrees) the arccosine of *val*
Syntax	*float* **acosd**(*float val*)

3.6.5.23 tan

Function	Calculates and returns the tangent of *val* (val in radians)
Syntax	*float* **tan**(*float val*)

3.6.5.24 tand

Function	Calculates and returns the tangent of *val* (val in degrees)
Syntax	*float* **tand**(*float val*)

3.6.5.25 atan

Function	Calculates and returns (in radians) the arctangent of *val*
Syntax	*float* **atan**(*float val*)

3.6.5.26 atand

Function	Calculates and returns (in degrees) the arctangent of *val*
Syntax	*float* **atand**(*float val*)

3.6.5.27 atan2

Function	Calculates and returns (in radians) the arctangent of y/x (handling x = 0)
Syntax	*float* **atan2**(*float y, float x*)

3.6.5.28 atan2d

Function	Calculates and returns (in degrees) the arctangent of y/x (handling x = 0)
Syntax	*float* **atan2d**(*float y, float x*)

3.6.5.29 pow

Function	Calculates and returns *val* raised to the xth power (= val^x). *val* and x cannot both be 0. If $val < 0$, x must be integral.
Syntax	*float* **pow**(*float val, numeric x*)

3.6.5.30 enthalpy

Function	Returns enthalpy of moist air (Btu/lb) for dry bulb temperature (F) and humidity ratio (lb/lb)
Syntax	*float* **enthalpy**(*float tDb, float w*)

3.6.5.31 wFromDbWb

Function	Returns humidity ratio (lb/lb) of moist air from dry bulb and wet bulb temperatures (F)
Syntax	*float* **wFromDbWb**(*float tDb, float tWb*)

3.6.5.32 wFromDbRh

Function	Returns humidity ratio (lb/lb) of moist air from dry bulb temperature (F) and relative humidity (0 – 1)
Syntax	*float* **wFromDbRh**(*float tDb, float rh*)

3.6.5.33 import

Function	Returns *float* read from an import file.
Syntax	*float* **import**(*string importFile, string colName*) *float* **import**(*string importFile, int colN*)
Remark	Columns can be referenced by name or 1-based index. See IMPORTFILE for details on use of import()

3.6.5.34 importStr

**Function	Returns *string* read from an import file.
Syntax	*string* **importStr**(*string importFile, string colName*) *string* **importStr**(*string importFile, int colN*)
Remark	See IMPORTFILE for details on use of importStr()

3.6.5.35 contin

Function	Returns continuous control value, e.g. for lighting control
Syntax	*float* **contin**(*float* mpf, *float* mlf, *float* sp, *float* val)
Remark	**contin** is evaluated at runtime and returns a value in the range 0 – 1 ???
Example	–

3.6.5.36 stepped

Function	Returns stepped reverse-acting control value, e.g. for lighting control
Syntax	*float* **stepped**(*int* nsteps, *float* sp, *float* val)
Remark	**stepped** is evaluated at runtime and returns a value in the range 0 – 1. If val <= 0, 1 is returned; if val >= sp, 0 is returned; otherwise, a stepped intermediate value is returned (see example)

example:

stepped(3, 12, val) returns

val	*result*
val $<$ 4	1
$4 \leq$ val < 8	.667
$8 \leq$ val < 12	.333
val ≥ 12	0

3.6.6 User-defined Functions

User defined functions have the format:

```
type FUNCTION name ( arg decls ) = expr ;
```

Type indicates the type of value the function returns, and can be:

```
INTEGER
FLOAT
STRING
DOY       (day of year date using month name and day; actually same as integer).
```

Arg decls indicates zero or more comma-separated argument declarations, each consisting of a *type* (as above) and the name used for the argument in *expr*.

Expr is an expression of (or convertible to) *type*.

The tradeoffs between using a user-defined function and a preprocessor macro (`#define`) include:

1. Function may be slightly slower, because its code is always kept separate and called, while the macro expansion is inserted directly in the input text, resulting in inline code.
2. Function may use less memory, because only one copy of it is stored no matter how many times it is called.
3. Type checking: the declared types of the function and its arguments allow CSE to perform additional checks.

Note that while macros require line-splicing ("\")to extend over one line, functions do not require it:

```
// Function returning number of days in ith month of year:
DOY FUNCTION MonthLU (integer i) = choose1 ( i , Jan 31, Feb 28, Mar 31,
                                                 Apr 30, May 31, Jun 30,
                                                 Jul 31, Aug 31, Sep 30,
                                                 Oct 31, Nov 30, Dec 31 ) ;
// Equivalent preprocessor macro:
#define MonthLU (i) = choose1 ( i , Jan 31, Feb 28, Mar 31,  \
                                    Apr 30, May 31, Jun 30,  \
                                    Jul 31, Aug 31, Sep 30,  \
                                    Oct 31, Nov 30, Dec 31 ) ;
```

3.6.7 Probes

Probes provide a universal means of referencing data within the simulator. Probes permit using the inputtable members of each object, as described in the Input Data Section, as operands in expressions. In addition, most internal members can be probed; we will describe how to find their names shortly.

Three general ways of using probes are:

1. During input, to implement things like "make this window's width equal to 10% of the zone floor area" by using the zone's floor area in an expression:

```
wnWidth = @zone[1].znArea * 0.1;
```

Here "`@zone[1].znArea`" is the probe.

2. During simulation. Probing during simulation, to make inputs be functions of conditions in the building or HVAC systems, is limited because most of the members of interest are updated *after* CSE has evaluated the user's expressions for the subhour or other time interval – this is logically necessary since the expressions are inputs. (An exception is the weather data, but this is also available through system variables such as $tDbO.)

However, a number of *prior subhour* values are available for probing, making it possible to implement relationships like "the local heat output of this terminal is 1000 Btuh if the zone temperature last subhour was below 65, else 500":

```
tuMnLh = @znres["North"].S.prior.tAir < 65 ? 1000 : 500;
```

3. For output reports, allowing arbitrary data to be reported at subhourly, hourly, daily, monthly, or annual intervals. The REPORT class description describes the user-defined report type (UDT), for which you write the expression for the value to be reported. With probes, you can thus report almost any datum within CSE – not just those values chosen for reporting when the program was designed. Even values calculated during the current subhour simulation can be probed and reported, because expressions for reports are evaluated after the subhour's calculations are performed.

Examples:

```
colVal = @airHandler["Hot"].ts;     // report air handler supply temp
colVal = @terminal[NorthHot].cz;    // terminal air flow to zone (Btuh/F)
```

The general form of a probe is

@ *className* [*objName*] . *member*

The initial @ is always necessary. And don't miss the period after the].

className is the CLASS being probed

objName	is the name of the specific object of the class; alternately, a numeric subscript is allowed. Generally, the numbers correspond to the objects in the order created. [*objName*] can be omitted for the TOP class, which has only one member, Top.
member	is the name of the particular member being probed. This must be exactly correct. For some inputtable members, the probe name is not the same as the input name given in the Input Data Section, and there are many probe-able members not described in the Input Data section.

How do you find out what the probe-able member names are? CSE will display the a list of the latest class and member names if invoked with the -p switch. Use the command line

```
CSE -p >probes.txt
```

to put the displayed information into the file PROBES.TXT, then print the file or examine it with a text editor.

A portion of the -p output looks like:

```
@exportCol[1..].        I   R                    owner: export
                name    I   R   string             constant
             colHead    I   R   string             input time
              colGap    I   R   integer number     input time
              colWid    I   R   integer number     input time
              colDec    I   R   integer number     input time
             colJust    I   R   integer number     constant
              colVal    I   R   un-probe-able      end of each subhour
              nxColi    I   R   integer number     constant

@holiday[1..].          I
                name    I       string             constant
          hdDateTrue    I       integer number     constant
           hdDateObs    I       integer number     constant
          hdOnMonday    I       integer number     constant
```

In the above "exportCol" and "holiday" are class names, and "name", "colHead", "colGap", . . . are member names for class exportCol. Some members have multiple names separated by .'s, or they may contain an additional subscript. To probe one of these, type all of the names and punctuation exactly as shown (except capitalization may differ); if an additional subscript is shown, give a number in the specified range. An "I" designates an "input" parameter, an R means "runtime" parameter. The "owner" is the class of which this class is a subclass.

The data type and variation of each member is also shown. Note that *variation*, or how often the member changes, is shown here. (*Variability*, or how often an expression assigned to the member may change, is given for the input table members in the Input Data Section). Members for which an "end of" variation is shown can be probed only for use in reports. A name described as "un-probe-able" is a structure or something not convertible to an integer, float, or string.

surface[].sgdist[].f[]: f[0] is winter solar coupling fraction; f[1] is summer.

3.6.8 Variation Frequencies Revisited

At risk of beating the topic to death, we're going to review once more the frequencies with which a CSE value can change (*variations)*, with some comments on the corresponding *variabilities*.

subhourly	changes in each "subhour" used in simulation. Subhours are commonly 15-minute intervals for models using znModel=CNE or 2-minute intervals for CSE znModels.
hourly	changes every simulated hour. The simulated weather and many other aspects of the simulation change hourly; it is customary to schedule setpoint changes, HVAC system operation, etc. in whole hours.
daily	changes at each simulated midnite.
monthly	changes between simulated months.
monthly-hourly, or "hourly on first day of each month"	changes once an hour on the first day of each month; the 24 hourly values from the first day of the month are used for the rest of the month. This variation and variability is used for data dependent on the sun's position, to save calculation time over computing it every hour of every day.
run start time	value is derived from other inputs before simulation begins, then does not change. Members that cannot change during the simulation but which are not needed to derive other values before the simulation begins have "run start time" *variability*.
input time	value is known before CSE starts to check data and derive "run start time" values. Expressions with "input time" variation may be used in many members that cannot accept any variation during the run. Many members documented in the Input Data Section as having "constant" variability may actually accept expressions with "input time" variation; to find out, try it: set the member to an expression containing a proposed probe and see if an error message results. "Input time" differs from "constant" in that it includes object names (forward references are allowed, and resolved just before other data checks) and probes that are forward references to constant values.
constant	does not vary. But a "constant" member of a class denoted as R (with no I) in the probes report produced by CSE -p is actually not available until run start time.

Also there are end-of varieties of all of the above; these are values computed during simulation: end of each hour, end of run, etc. Such values may be reported (using a probe in a UDT report), but will produce an error message if probed in an expression for an input member value.

4 Input Data

This section describes the input for each CSE class (object type). For each object you wish to define, the usual input consists of the class name, your name for the particular object (usually), and zero or more member value statements of the form *name=expression*. The name of each subsection of this section is a class name (HOLIDAY, MATERIAL, CONSTRUCTION, etc.). The object name, if given, follows the class name; it is the first thing in each description (hdName, matName, conName, etc.). Exception: no statement is used to create or begin the predefined top-level object "Top" (of class TOP); its members are given without introduction.

After the object name, each member's description is introduced with a line of the form *name=type*. *Type* indicates the appropriate expression type for the value:

- *float*
- *int*
- *string*
- ______*name* (object name for specified type of object)
- *choice*
- *date*

These types discussed in the section on expression types.

Each member's description continues with a table of the form:

Units	Legal Range	Default	Required	Variability
ft^2	x > 0	wnHeight * wnWidth	No	constant

where the column headers have the following meaning:

Units	units of measure (lb., ft, Btu, etc.) where applicable
Legal Range	limits of valid range for numeric inputs; valid choices for *choice* members, etc.
Default	value assumed if member not given; applicable only if not required
Required	YES if you must give this member
Variability	how often the given expression can change: hourly, daily, etc. See sections on expressions, statements, and variation frequencies

4.1 TOP Members

The top-level data items (TOP members) control the simulation process or contain data that applies to the modeled building as a whole. No statement is used to begin or create the TOP object; these statements can be given anywhere in the input (they do, however, terminate any other objects being specified – ZONEs, REPORTs, etc.).

4.1.1 TOP General Data Items

doMainSim=*choice*

Specifies whether the simulation is performed when a Run command is encountered. See also doAutoSize.

Units	Legal Range	Default	Required	Variability
	NO,YES	YES	No	constant

begDay=*date*

Date specifying the beginning day of the simulation performed when a Run command is encountered. See further discussion under endDay (next).

Units	Legal Range	Default	Required	Variability
	date	Jan 1	No	constant

endDay=*date*

Date specifying the ending day of the simulation performed when a Run command is encountered.

The program simulates 365 days at most. If begDay and endDay are the same, 1 day is simulated. If begDay precedes endDay in calendar sequence, the simulation is performed normally and covers begDay through endDay inclusive. If begDay follows endDay in calendar sequence, the simulation is performed across the year end, with Jan 1 immediately following Dec 31.

Units	Legal Range	Default	Required	Variability
	date	Dec 31	No	constant

jan1DoW=*choice*

Day of week on which January 1 falls.

Units	Legal Range	Default	Required	Variability
	SUN MON TUE WED THU FRI SAT	THU	No	constant

workDayMask=*int* TODO

Units	Legal Range	Default	Required	Variability
		Mon-fri?	No	constant

wuDays=*int*

Number of "warm-up" days used to initialize the simulator. Simulator initialization is required because thermal mass temperatures are set to arbitrary values at the beginning of the simulation. Actual mass temperatures must be established through simulation of a few days before thermal loads are accumulated. Heavier buildings require more warm-up; the default values are adequate for conventional construction.

Units	Legal Range	Default	Required	Variability
	$x \geq 0$	7	No	constant

nSubSteps=*int*

Number of subhour steps used per hour in the simulation. 4 is the time-honored value for models using CNE zones. A value of 30 is typically for CSE zone models.

Units	Legal Range	Default	Required	Variability
	$x > 0$	4	No	constant

tol=*float*

Endtest convergence tolerance for internal iteration in CNE models (no effect for CSE models) Small values for the tolerance cause more accurate simulations but slower performance. The user may wish to use a high number during the initial design process (to quicken the runs) and then lower the tolerance for the final design (for better accuracy). Values other than .001 have not been explored.

Units	Legal Range	Default	Required	Variability
	$x > 0$	.001	No	constant

humTolF=*float*

Specifies the convergence tolerance for humidity calculations in CNE models (no effect in for CSE models), relative to the tolerance for temperature calculations. A value of .0001 says that a humidity difference of .0001 is about as significant as a temperature difference of one degree. Note that this is multiplied internally by "tol"; to make an overall change in tolerances, change "tol" only.

Units	Legal Range	Default	Required	Variability
	$x > 0$	.0001	No	

ebTolMon=*float*

Monthly energy balance error tolerance for internal consistency checks. Smaller values are used for testing the internal consistency of the simulator; values somewhat larger than the default may be used to avoid error messages when it is desired to continue working despite a moderate degree of internal inconsistency.

Units	Legal Range	Default	Required	Variability
	$x > 0$	0.0001	No	constant

ebTolDay=*float*

Daily energy balance error tolerance.

Units	Legal Range	Default	Required	Variability
	$x > 0$	0.0001	No	constant

ebTolHour=*float*

Hourly energy balance error tolerance.

Units	Legal Range	Default	Required	Variability
	$x > 0$	0.0001	No	constant

ebTolSubhr=*float*

Sub-hourly energy balance error tolerance.

Units	Legal Range	Default	Required	Variability
	$x > 0$	0.0001	No	constant

humMeth=*choice*

Developmental zone humidity computation method choice for CNE models (no effect for CSE models).

ROB	Rob's backward difference method. Works well within limitations of backward difference approach.
PHIL	Phil's central difference method. Should be better if perfected, but initialization at air handler startup is unresolved, and ringing has been observed.

Units	Legal Range	Default	Required	Variability
	ROB, PHIL	ROB	No	constant

dflExH=*float*

Default exterior surface (air film) conductance used for opaque and glazed surfaces exposed to ambient conditions in the absence of explicit specification.

Units	Legal Range	Default	Required	Variability
Btuh/ft^2-^{o}F	$x > 0$	2.64	No	constant

bldgAzm=*float*

Reference compass azimuth (0 = north, 90 = east, etc.). All zone orientations (and therefore surface orientations) are relative to this value, so the entire building can be rotated by changing bldgAzm only. If a value outside the range $0^o \leq x < 360^o$ is given, it is normalized to that range.

Units	Legal Range	Default	Required	Variability
^{o} (degrees)	unrestricted	0	No	constant

elevation=*float*

Elevation of the building site. Used internally for the computation of barometric pressure and air density of the location.

Units	Legal Range	Default	Required	Variability
ft	$x \geq 0$	0 (sea level)	No	constant

runTitle=*string*

Run title for the simulation. Appears in report footers, export headers, and in the title lines to the INP, LOG, and ERR built-in reports (these appear by default in the primary report file; the ERR report also appears in the error message file, if one is created).

Units	Legal Range	Default	Required	Variability
	63 characters	blank (no title)	No	constant

runSerial=*int*

Run serial number for the simulation. Increments on each run in a session; appears in report footers.

Units	Legal Range	Default	Required	Variability
	$0 \leq x \leq 999$	0	No	constant

4.1.2 TOP Daylight Saving Time Items

Daylight savings starts by default at 2:00 a.m. of the second Sunday in March. Internally, hour 3 (2:00-3:00 a.m.) is skipped and reports for this day show only 23 hours. Daylight savings ends by default at 2:00 a.m. of the first Sunday of November; for this day 25 hours are shown on reports. CSE fetches weather data using standard time but uses daylight savings time to calculate variable expressions (and thus all schedules).

DT=*choice*

Whether Daylight Savings Time is to be used for the current run.

Units	Legal Range	Default	Required	Variability
	YES, NO	YES	No	constant

DTbegDay=*date*

Start day for daylight saving time (assuming DT=Yes)

Units	Legal Range	Default	Required	Variability
	date	*second Sunday in March*	No	constant

DTendDay=*date*

End day for daylight saving time (assuming DT=Yes)

Units	Legal Range	Default	Required	Variability
	date	*first Sunday in November*	No	constant

4.1.3 TOP Model Control Items

ventAvail=*choice*

Indicates availability of outdoor ventilation strategies. CSE cannot model simultaneously-operating alternative ventilation strategies. For example, an RSYS central fan integrated (CFI) OAV system is never modeled while whole house fan ventilation is available. ventAvail controls which ventilation mode, if any, is available for the current hour. Note that mode availability means that the strategy could operate but may not operate due to other control assumptions.

Choice	Ventilation Strategy Available
NONE	None
WHOLEBUILDING	IZXFER (window and whole-house fan)
RSYSOAV	RSYS central fan integrated (CFI) outside air ventilation (OAV)

As noted, ventAvail is evaluated hourly, permitting flexible control strategy modeling. The following example specifies that RSYSOAV (CFI) ventilation is available when the seven day moving average temperature is above 68 °F, otherwise whole building ventilation is available between 7 and 11 PM, otherwise no ventilation.

```
ventAvail = (@weather.taDbAvg07 > 68)    ? RSYSOAV
          : ($hour >= 19 && $hour <= 23) ? WHOLEBUILDING
          :                                NONE
```

Units	Legal Range	Default	Required	Variability
	Choices above	WHOLEBUILDING	No	hourly

dhwModel=*choice*

Modifies aspects of DHW calculations.

Choice	Effect
T24DHW	Matches results from T24DHW.DLL
2013	Corrected CEC 2013 methods with 2016 updates

Units	Legal Range	Default	Required	Variability
	Choices above	2013	No	constant

exShadeModel=*choice*

Specifies advanced exterior shading model used to evaluate shading of PVARRAYs by SHADEXs or other PVARRAYs. Advanced shading is not implemented for building surfaces and this setting has no effect on walls or windows.

Choice	Effect
PENUMBRA	Calculate shading using the Penumbra model
NONE	Disable advanced shading calculations

Units	Legal Range	Default	Required	Variability
	Choices above	PENUMBRA	No	constant

ANTolAbs=*float*

AirNet absolute convergence tolerance. Ideally, calculated zone air pressures should be such that the net air flow into each zone is 0 – that is, there should be a perfect mass balance. The iterative AirNet solution techniques are deemed converged when netAirMassFlow < max(ANTolAbs, ANTolRel*totAirMassFlow).

Units	Legal Range	Default	Required	Variability
lbm/sec	x > 0	0.00125 (about 1 cfm)	No	constant

ANTolRel=*float*

AirNet relative convergence tolerance. See AnTolAbs just above.

Units	Legal Range	Default	Required	Variability
	x > 0	.0001	No	constant

The ASHWAT complex fenestration model used when WINDOW wnModel=ASHWAT yields several heat transfer results that are accurate over local ranges of conditions. Several values control when these value are recalculated. If any of the specified values changes more than the associated threshold, a full ASHWAT calculation is triggered. Otherwise, prior results are used. ASHWAT calculations are computationally expensive and conditions often change only incrementally between time steps.

AWTrigT=*float*

ASHWAT temperature change threshold – full calculation is triggered by a change of either indoor or outdoor environmental (combined air and radiant) temperature that exceeds AWTrigT.

Units	Legal Range	Default	Required	Variability
°F	x > 0	1	No	constant

AWTrigSlr=*float*

ASHWAT solar change threshold – full calculation is triggered by a fractional change of incident solar radiation that exceeds AWTrigSlr.

Units	Legal Range	Default	Required	Variability
	x > 0	.05	No	constant

AWTrigH=*float*

ASHWAT convection coefficient change threshold – full calculation is triggered by a fractional change of inside surface convection coefficient that exceeds AWTrigH.

Units	Legal Range	Default	Required	Variability
	x > 0	.1	No	constant

4.1.4 TOP Weather Data Items

The following system variables (4.6.4) are determined from the weather file for each simulated hour:

$radBeam	beam irradiance on tracking surface (integral for hour, Btu/ft^2).
$radDiff	diffuse irradiance on a horizontal surface (integral for hour, Btu/ft^2).
$tDbO	dry bulb temp (°F).
$tWbO	wet bulb temp (°F).
$wO	humidity ratio
$windDirDeg	wind direction (degrees, NOT RADIANS; 0=N, 90=E).
$windSpeed	wind speed (mph).

The following are the terms determined from the weather file for internal use, and can be referenced with the probes shown.

`@Top.depressWbWet bulb depression (F).`

`@Top.windSpeedSquaredWind speed squared (mph2).`

wfName=*string*

Weather file path name for simulation. The file should be in the current directory, in the directory CSE.EXE was read from, or in a directory on the operating system PATH. Weather file formats supported are CSW, EPW, and ET1. Only full-year weather files are supported.

Note: Backslash (\) characters in path names must be doubled to work properly (e.g. "\\wthr\\mywthr.epw"). Forward slash (/) may be used in place of backslash without doubling.

Units	Legal Range	Default	Required	Variability
	file name,path optional		Yes	constant

Units	Legal Range	Default	Required	Variability
	file name,path optional		Yes	constant

skyModel=*choice*

Selects sky model used to determine relative amounts of direct and diffuse irradiance.

ISOTROPIC	traditional isotropic sky model
ANISOTROPIC	Hay anisotropic model

Units	Legal Range	Default	Required	Variability
	choices above	ANISOTROPIC	No	constant

skyModelLW=*choice*

Selects the model used to derive sky temperature used in long-wave (thermal) radiant heat exchange calculations for SURFACEs exposed to ambient conditions. See the RACM alorithms documentation for technical details.

Choice	Description
DEFAULT	Default: tSky from weather file if available else Berdahl-Martin
BERDAHLMARTIN	Berdahl-Martin (tSky depends on dew point, cloud cover, and hour)
DRYBULB	tSky = dry-bulb temperature (for testing)
BLAST	Blast model (tSky depends on dry-bulb)

Units	**Legal Range**	**Default**	**Required**	**Variability**
	choices above	DEFAULT	No	constant

The reference temperature and humidity are used to calculate a humidity ratio assumed in air specific heat calculations. The small effect of changing humidity on the specific heat of air is generally ignored in the interests of speed, but the user can control the humidity whose specific heat is used through the refTemp and refRH inputs.

refTemp=*float*

Reference temperature (see above paragraph).

Units	**Legal Range**	**Default**	**Required**	**Variability**
°F	$x \geq 0$	60°	No	constant

refRH=*float*

Reference relative humidity (see above).

Units	**Legal Range**	**Default**	**Required**	**Variability**
	$0 \leq x \leq 1$	0.6	No	constant

grndRefl=*float*

Global ground reflectivity, used except where other value specified with sfGrndRefl or wnGrndRefl. This reflectivity is used in computing the reflected beam and diffuse radiation reaching the surface in question.

Units	**Legal Range**	**Default**	**Required**	**Variability**
	$0 \leq x \leq 1$	0.2	No	Monthly-Hourly

The following values modify weather file data, permitting varying the simulation without making up special weather files. For example, to simulate without the effects of wind, use windF = 0; to halve the effects of diffuse solar radiation, use radDiffF = 0.5. Note that the default values for windSpeedMin and windF result in modification of weather file wind values unless other values are specified.

windSpeedMin=*float*

Minimum value for wind speed

Units	**Legal Range**	**Default**	**Required**	**Variability**
mph	$x \geq 0$	0.5	No	constant

windF=*float*

Wind Factor: multiplier for wind speeds read from weather file. windF is applied *after* windSpeedMin. Note that windF does *not* effect infiltration rates calculated by the Sherman-Grimsrud model (see e.g. ZONE.infELA). However, windF does modify AirNet flows (see IZXFER).

Units	Legal Range	Default	Required	Variability
	$x \geq 0$	0.25	No	constant

terrainClass=*int*

Specifies characteristics of ground terrain in the project region.

1	ocean or other body of water with at least 5 km unrestricted expanse
2	flat terrain with some isolated obstacles (buildings or trees well separated)
3	rural areas with low buildings, trees, etc.
4	urban, industrial, or forest areas
5	center of large city

Units	Legal Range	Default	Required	Variability
	$1 \leq x \leq 5$	4	No	constant

radBeamF=*float*

Multiplier for direct normal (beam) irradiance

Units	Legal Range	Default	Required	Variability
	$x \geq 0$	1	No	constant

radDiffF=*float*

Multiplier for diffuse horizonal irradiance.

Units	Legal Range	Default	Required	Variability
	$x \geq 0$	1	No	constant

soilDiff=*float*

Soil diffusivity, used in derivation of ground temperature. CSE calculates a ground temperature at 10 ft depth for each day of the year using dry-bulb temperatures from the weather file and soilDiff. Ground temperature is used in heat transfer calculations for SURFACEs with sfExCnd=GROUND. Note that derivation of mains water temperature for DHW calculations involves a ground temperature based on soil diffusivity = 0.025 and does not use this soilDiff.

Units	Legal Range	Default	Required	Variability
ft^2/hr	x > 0	0.025	No	constant

4.1.5 TOP TDV (Time Dependent Value) Items

CSE supports an optional comma-separated (CSV) text file that provides hourly TDV values for electricity and fuel. TDV values are read along with the weather file and the values merged with weather data. Several daily statistics are calculated for use via probes. The file has no other effect on the simulation. Only full-year TDV files are supported.

The format of a TDV file is the same as an IMPORTFILE with the proviso that the 4 line header is not optional and certain header items must have specified values. In the following table, non-italic items must be provided as shown (with optional quotes).

Line	Contents	Notes
1	TDV Data (TDV/Btu), *runNumber*	*runNumber* is not checked
2	*timestamp*	optionally in quotes accessible via @TOP.TDVFileTimeStamp
3	*title*, hour	*title* (in quotes if it contains commas) accessible via @TOP.TDVFileTitle
4	tdvElec, tdvFuel	comma separated column names (optionally in quotes) not checked
5 ..	*valElec*,*valFuel*	comma separated numerical values (8760 or 8784 rows) tdvElec is always in column 1, tdvFuel always in column 2 column names in row 4 do not determine order

Example TDV file –

```
"TDV Data (TDV/Btu)","001"
"Wed 14-Dec-16  12:30:29 pm"
"BEMCmpMgr 2019.0.0 RV (758), CZ12, Fuel NatGas", Hour
"tdvElec","tdvFuel"
7.5638,2.2311
7.4907,2.2311
7.4478,2.2311
7.4362,2.2311
7.5255,2.2311
7.5793,2.2311
7.6151,2.2311
7.6153,2.2311
7.5516,2.2311
(... continues for 8760 or 8784 data lines ...)
```

Note: additional columns can be included and are ignored.

The following probes are available for accessing TDV data in expressions –

Probe	Variability	Description
@Weather.tdvElec	Hour	current hour electricity TDV
@Weather.tdvFuel	Hour	current hour fuel TDV
@Weather.tdvElecPk	Day	current day peak electricity TDV (includes future hours)
@Weather.tdvElecAvg	Day	current day average electricity TDV (includes future hours)
@Weather.tdvElecPvPk	Day	previous day peak electricity TDV
@Weather.tdvElecAvg01	Day	previous day average electricity TDV
@weatherFile.tdvFileTimeStamp	Constant	TDV file timestamp (line 2 of header)

Probe	Variability	Description
@weatherFile.tdvFileTitle	Constant	TDV file title (line 3 of header)
@Top.tdvFName	Constant	TDV file full path

TDVfName=*string*

Note: Backslash (\) characters in path names must be doubled to work properly (e.g. "\\data\\mytdv.tdv"). Forward slash (/) may be used in place of backslash without doubling.

Units	**Legal Range**	**Default**	**Required**	**Variability**
	file name, path optional	(no TDV file)	No	constant

4.1.6 TOP Report Data Items

These items are used in page-formatted report output files. See REPORTFILE, Section 5.245.21, and REPORT, Section 5.25.

repHdrL=*string*

Report left header. Appears at the upper left of each report page unless page formatting (rfPageFmt) is OFF. If combined length of repHdrL and repHdrR is too large for the page width, one or both will be truncated.

Units	**Legal Range**	**Default**	**Required**	**Variability**
		blank	No	constant??

repHdrR=*string*

Report right header. Appears at the upper right of each report page unless page formatting (rfPageFmt) is OFF. If combined length of repHdrL and repHdrR is too large for the page width, one or both will be truncated.

Units	**Legal Range**	**Default**	**Required**	**Variability**
		blank(no right header)	No	constant??

repLPP=*int*

Total lines per page to be assumed for reports. Number of lines used for text (including headers and footers) is repLPP - repTopM - repBotM.

Units	**Legal Range**	**Default**	**Required**	**Variability**
lines	$x \geq 50$	66	No	constant??

repTopM=*int*

Number of lines to be skipped at the top of each report page (prior to header).

Units	**Legal Range**	**Default**	**Required**	**Variability**
lines	$0 \geq x \geq 12$	3	No	constant

repBotM=*int*

Number of lines reserved at the bottom of each report page. repBotM determines the position of the footer on the page (blank lines after the footer are not actually written).

Units	Legal Range	Default	Required	Variability
lines	$0 \leq x \leq 12$	3	No	constant

repCPL=*int*

Characters per line for report headers and footers, user defined reports, and error messages. CSE writes simple ASCII files and assumes a fixed (not proportional) spaced printer font. Many of the built-in reports now (July 1992) assume a line width of 132 columns.

Units	Legal Range	Default	Required	Variability
characters	$78 \leq x \leq 132$	78	No	constant

repTestPfx=*string*

Report test prefix. Appears at beginning of report lines that are expected to differ from prior runs. This is useful for "hiding" lines from text comparison utilities in automated testing schemes. Note: the value specified with command line -x takes precedence over this input.

Units	Legal Range	Default	Required	Variability
		blank	No	constant??

4.1.7 TOP Autosizing

doAutoSize=*choice*

Controls invocation of autosizing phase prior to simulation.

Units	Legal Range	Default	Required	Variability
	YES, NO	NO, unless AUTOSIZE commands in input	No	constant

auszTol=*float*

Autosize tolerance. Sized capacity results are deemed final when successive design day calculations produce results within auszTol of the prior iteration.

Units	Legal Range	Default	Required	Variability
		.005	No	constant

heatDsTDbO=*float*

Heating outdoor dry bulb design temperature used for autosizing heating equipment.

Units	Legal Range	Default	Required	Variability
°F			if autosizing	hourly

heatDsTWbO=*float*

Heating outdoor Whether bulb design temperature used for autosizing heating equipment.

Units	Legal Range	Default	Required	Variability
°F		derived assuming RH=.7	No	hourly

coolDsDay=*list of up to 12 days*

Specifies cooling design days for autosizing. Each day will be simulated repeatedly using weather file conditions for that day.

Units	Legal Range	Default	Required	Variability
dates		*none*	No	constant

coolDsMo=*list of up to 12 months*

Deprecated method for specifying cooling autosizing days. Design conditions are taken from ET1 weather file header, however, the limited availale ET1 files do not contain design condition information.

Units	Legal Range	Default	Required	Variability
months		*none*	No	constant

4.1.8 TOP Debug Reporting

verbose=*int*

Controls verbosity of screen remarks. Most possible remarks are generated during autosizing of CNE models. Little or no effect in CSE models. TODO: document options

Units	Legal Range	Default	Required	Variability
	0 – 5 ?	1	No	constant

The following dbgPrintMask values provide bitwise control of addition of semi-formated internal results to the run report file. The values and format of debugging reports are modified as required for testing purposes.

dbgPrintMaskC=*int*

Constant portion of debug reporting control.

Units	Legal Range	Default	Required	Variability
		0	No	constant

dbgPrintMask=*int*

Hourly portion of debug reporting control (generally an expression that evaluates to non-0 only on days or hours of interest).

Units	Legal Range	Default	Required	Variability
		0	No	hourly

4.2 HOLIDAY

HOLIDAY objects define holidays. Holidays have no inherent effect, but input expressions can test for holidays via the $DOWH, $isHoliday, $isHoliTrue, $isWeHol, and $isBegWeek system variables (4.6.4).

Examples and the list of default holidays are given after the member descriptions.

hdName

Name of holiday: must follow the word HOLIDAY.

Units	Legal Range	Default	Required	Variability
	63 characters	*none*	Yes	constant

A holiday may be specified by date or via a rule such as "Fourth Thursday in November". To specify by date, give hdDateTrue, and also hdDateObs or hdOnMonday if desired. To specify by rule, give all three of hdCase, hdMon, and hdDow.

hdDateTrue=*date*

The true date of a holiday, even if not celebrated on that day.

Units	Legal Range	Default	Required	Variability
	date	*blank*	No	constant

hdDateObs=*date*

The date that a holiday will be observed. Allowed only if hdDateTrue given and hdOnMonday not given.

Units	Legal Range	Default	Required	Variability
	date	*hdDateTrue*	No	constant

hdOnMonday=*choice*

If YES, holiday is observed on the following Monday if the true date falls on a weekend. Allowed only if hdDateTrue given and hdDateObs not given.

Note: there is no provision to celebrate a holiday that falls on a Saturday on *Friday* (as July 4 was celebrated in 1992).

Units	Legal Range	Default	Required	Variability
	YES NO	YES	No	constant

hdCase=*choice*

Week of the month that the holiday is observed. hdCase, hdMon, and hdDow may be given only if hdDateTrue, hdDateObs, and hdOnMonday are not given.

Units	Legal Range	Default	Required	Variability
	FIRST SECOND THIRD FOURTH LAST	FIRST	No	constant

hdMon=*choice*

Month that the holiday is observed.

Units	Legal Range	Default	Required	Variability
	JAN, FEB, MAR, APR, MAY, JUN, JUL, AUG, SEP, OCT, NOV, DEC	*none*	required if hdCase given	constant

hdDow=*choice*

Day of the week that the holiday is observed.

Units	Legal Range	Default	Required	Variability
	SUNDAY, MONDAY, TUESDAY, WEDNESDAY, THURSDAY, FRIDAY, SATURDAY	MONDAY	required if hdCase given	constant

endHoliday

Indicates the end of the holiday definition. Alternatively, the end of the holiday definition can be indicated by "END" or simply by beginning another object.

Units	Legal Range	Default	Required	Variability
		N/A	No	constant

Examples of valid HOLIDAY object specifications:

- Holiday on May first, observed date moved to following Monday if the first falls on a weekend (hdOnMonday defaults Yes).

```
HOLIDAY MAYDAY;
    hdDateTrue = May 1;
```

- Holiday on May 1, observed on May 3.

```
HOLIDAY MAYDAY;
    hdDateTrue = May 1;
    hdDateObs  = May 3;
```

- Holiday observed on May 1 even if on a weekend.

```
HOLIDAY MAYDAY;
    hdDateTrue = May 1;
    hdOnMonday = No;
```

- Holiday observed on Wednesday of third week of March

```
HOLIDAY HYPOTHET;
    hdCase = third;
    hdDow  = Wed;
    hdMon  = MAR
```

As with reports, Holidays are automatically generated for a standard set of Holidays. The following are the default holidays automatically defined by CSE:

New Year's Day	*January 1
M L King Day	*January 15
President's Day	3rd Monday in February
Memorial Day	last Monday in May
Fourth of July	*July 4
Labor Day	1st Monday in September
Columbus Day	2nd Monday in October
Veterans Day	*November 11
Thanksgiving	4th Thursday in November
Christmas	*December 25

* *observed on the following Monday if falls on a weekend, except as otherwise noted:*

If a particular default holiday is not desired, it can be removed with a DELETE statement:

```
DELETE HOLIDAY Thanksgiving

DELETE HOLIDAY "Columbus Day"  // Quotes necessary (due to space)

DELETE HOLIDAY "VETERANS DAY"  // No case-sensitivity
```

Note that the name must be spelled *exactly* as listed above.

4.3 MATERIAL

MATERIAL constructs an object of class MATERIAL that represents a building material or component for later reference a from LAYER (see below). A MATERIAL so defined need not be referenced. MATERIAL properties are defined in a consistent set of units (all lengths in feet), which in some cases differs from units used in tabulated data. Note that the convective and air film resistances for the inside wall surface is defined within the SURFACE statements related to conductances.

matName

Name of material being defined; follows the word "MATERIAL".

Units	Legal Range	Default	Required	Variability
	63 characters	*none*	Yes	constant

matThk=*float*

Thickness of material. If specified, matThk indicates the discreet thickness of a component as used in construction assemblies. If omitted, matThk indicates that the material can be used in any thickness; the thickness is then specified in each LAYER using the material (see below).

Units	Legal Range	Default	Required	Variability
ft	$x > 0$	(omitted)	No	constant

matCond=*float*

Conductivity of material. Note that conductivity is *always* stated for a 1 foot thickness, even when matThk is specified; if the conductance is known for a specific thickness, an expression can be used to derive matCond.

Units	Legal Range	Default	Required	Variability
Btuh-ft/ft^2-°F	$x > 0$	*none*	Yes	constant

matCondT=*float*

Temperature at which matCond is rated. See matCondCT (next).

Units	Legal Range	Default	Required	Variability
°F	$x > 0$	70 °F	No	constant

matCondCT=*float*

Coefficient for temperature adjustment of matCond in the forward difference surface conduction model. Each hour (not subhour), the conductivity of layers using this material are adjusted as followslrCond = matCond * (1 + matCondCT*(T_{layer} – matCondT))

Units	Legal Range	Default	Required	Variability
°F^{-1}		0	No	constant

Note: A typical value of matCondCT for fiberglass batt insulation is 0.00418 F^{-1}

matSpHt=*float*

Specific heat of material.

Units	Legal Range	Default	Required	Variability
Btu/lb-°F	$x \geq 0$	0 (thermally massless)	No	constant

matDens=*float*

Density of material.

Units	Legal Range	Default	Required	Variability
lb/ft^3	$x \geq 0$	0 (massless)	No	constant

matRNom=*float*

Nominal R-value per foot of material. Appropriate for insulation materials only and *used for documentation only.* If specified, the current material is taken to have a nominal R-value that contributes to the reported nominal R-value for a construction. As with matCond, matRNom is *always* stated for a 1 foot thickness, even when matThk is specified; if the nominal R-value is known for a specific thickness, an expression can be used to derive matRNom.

Units	Legal Range	Default	Required	Variability
ft^2-°F/Btuh	$x > 0$	(omitted)	No	constant

endMaterial

Optional to indicate the end of the material. Alternatively, the end of the material definition can be indicated by "END" or simply by beginning another object.

Units	Legal Range	Default	Required	Variability
		N/A	No	constant

4.4 CONSTRUCTION

CONSTRUCTION constructs an object of class CONSTRUCTION that represents a light weight or massive ceiling, wall, floor, or mass assembly (mass assemblies cannot, obviously, be lightweight). Once defined, CONSTRUCTIONs can be referenced from SURFACEs (below). A defined CONSTRUCTION need not be referenced. Each CONSTRUCTION is optionally followed by LAYERs, which define the constituent LAYERs of the construction.

conName

Name of construction. Required for reference from SURFACE and DOOR objects, below.

Units	Legal Range	Default	Required	Variability
	63 characters	*none*	Yes	constant

conU=*float*

U-value for the construction (NOT including surface (air film) conductances; see SURFACE statements). If omitted, one or more LAYERs must immediately follow to specify the LAYERs that make up the construction. If specified, no LAYERs can follow.

Units	Legal Range	Default	Required	Variability
Btuh/ft^2- °F	$x > 0$	calculated from LAYERs	if omitted, LAYERs must follow	constant

endConstruction

Optional to indicates the end of the CONSTRUCTION. Alternatively, the end of the CONSTRUCTION definition can be indicated by "END" or by beginning another object If END or endConstruction is used, it should follow the construction's LAYER subobjects, if any.

Units	Legal Range	Default	Required	Variability
		N/A	No	constant

4.5 LAYER

LAYER constructs a subobject of class LAYER belonging to the current CONSTRUCTION. LAYER is not recognized except immediately following CONSTRUCTION or another LAYER. The members represent one layer (that optionally includes framing) within the CONSTRUCTION.

The layers should be specified in inside to outside order. A framed layer (lrFrmMat and lrFrmFrac given) is modeled by creating a homogenized material with weighted combined conductivity and volumetric heat capacity. Caution: it is generally preferable to model framed constructions using two separate surfaces (one with framing, one without). At most one framed layer (lrFrmMat and lrFrmFrac given) is allowed per construction.

The layer thickness may be given by lrThk, or matThk of the material, or matThk of the framing material if any. The thickness must be specified at least one of these three places; if specified in more than one place and not consistent, an error message occurs.

lrName

Name of layer (follows "LAYER"). Required only if the LAYER is later referenced in another object, for example with LIKE or ALTER; however, we suggest naming all objects for clearer error messages and future flexibility.

Units	Legal Range	Default	Required	Variability
	63 characters	*None*	No	constant

lrMat=*matName*

Name of primary MATERIAL in layer.

Units	Legal Range	Default	Required	Variability
	name of a *MATERIAL*	*none*	Yes	constant

lrThk=*float*

Thickness of layer.

Units	Legal Range	Default/Required	Variability
ft	$x > 0$	Required if *matThk* not specified in referenced *lrMat*	constant

lrFrmMat=*matName*

Name of framing MATERIAL in layer, if any. At most one layer with lrFrmMat is allowed per CONSTRUCTION. See caution above regarding framed-layer model.

Units	Legal Range	Default	Required	Variability
	name of a MATERIAL	*no framed layer*	No	constant

lrFrmFrac=*float*

Fraction of layer that is framing. Must be specified if frmMat is specified. See caution above regarding framed-layer model.

Units	Legal Range	Default	Required	Variability
	$0 \leq x \leq 1$	*no framed layer*	Required if *lrFrmMat* specified, else disallowed	constant

endLayer

Optional end-of-LAYER indicator; LAYER definition may also be indicated by "END" or just starting the definition of another LAYER or other object.

4.6 GLAZETYPE

GLAZETYPE constructs an object of class GLAZETYPE that represents a glazing type for use in WINDOWs.

gtName

Name of glazetype. Required for reference fromWINDOW objects, below.

Units	Legal Range	Default	Required	Variability
	63 characters	*none*	Yes	constant

gtModel=*choice*

Selects model to be used for WINDOWs based on this GLAZETYPE.

Units	Legal Range	Default	Required	Variability
	SHGC ASHWAT	SHGC	No	constant

gtU=*float*

Glazing conductance (U-factor without surface films, therefore not actually a U-factor but a C-factor). Used as wnU default; an error message will be issued if the U value is not given in the window (wnU) nor in the glazeType (gtU). Preferred Approach: To use accurately with standard winter rated U-factor from ASHRAE or NFRC enter as:

```
gtU = (1/((1/U-factor)-0.85)
```

Where 0.85 is the sum of the interior (0.68) and exterior (0.17) design air film resistances assumed for rating window U-factors. Enter wnInH (usually 1.5=1/0.68) instead of letting it default. Enter the wnExH or let it default. It is important to use this approach if the input includes gnFrad for any gain term. Using approach 2 below will result in an inappropriate internal gain split at the window.

Approach 2. Enter gtU=U-factor and let the wnInH and wnExH default. This approach systematically underestimates the window U-factor because it adds the wnExfilm resistance to 1/U-factor thereby double counting the exterior film resistance. This approach will also yield incorrect results for gnFrad internal gain since the high wnInH will put almost all the gain back in the space.

Units	Legal Range	Default	Required	Variability
Btuh/ft^2-^{o}F	$x > 0$	*none*	No	constant

gtUNFRC=*float*

Fenestration system (including frame) U-factor evaluated at NFRC heating conditions. For ASHWAT windows, a value for the NFRC U-factor is required, set via gtUNFRC or wnUNFRC.

Units	Legal Range	Default	Required	Variability
Btuh/ft^2-^{o}F	$x > 0$	*none*	No	constant

gtSHGC=*float*

Glazing Solar Heat Gain Coefficient: fraction of normal beam insolation which gets through glass to space inside. We recommend using this to represent the glass normal transmissivity characteristic only, before shading and framing effects

Units	Legal Range	Default	Required	Variability
fraction	*$0 \leq x \leq 1$*	*none*	Yes	Constant

gtSMSO=*float*

SHGC multiplier with shades open. May be overriden in the specific window input.

Units	Legal Range	Default	Required	Variability
fraction	*$0 \leq x \leq 1$*	1.0	No	Monthly - Hourly

gtSMSC=*float*

SHGC multiplier with shades closed. May be overriden in the specific window input.

Units	Legal Range	Default	Required	Variability
fraction	*$0 \leq x \leq 1$*	gtSMSO (no shades)	No	Monthly - Hourly

gtFMult=*float*

Framing multiplier used if none given in window, for example .9 if frame and mullions reduce the solar gain by 10%. Default of 1.0 implies frame/mullion effects allowed for in gtSHGC's or always specified in Windows.

Units	Legal Range	Default	Required	Variability
fraction	*$0 \leq x \leq 1$*	gtSHGCO	No	Monthly - Hourly

gtPySHGC =*float*

Four float values separated by commas. Coefficients for incidence angle SHGC multiplier polynomial applied to gtSHGC to determine beam transmissivity at angles of incidence other than 90 degrees. The values are coefficients for first through fourth powers of the cosine of the incidence angle; there is no constant part. An error message will be issued if the coefficients do not add to one. They are used in the following computation:

angle = incidence angle of beam radiation, measured from normal to glass.

cosI = cos(angle)

angMult = a*cosI + b*cosI^2 + c*cosI^3 + d*cosI^4

beamXmisvty = gtSHGCO * angMult (shades open)

Units	Legal Range	Default	Required	Variability
float	*any*	none	Yes	Constant

gtDMSHGC=*float*

SHGC diffuse multiplier, applied to gtSHGC to determine transmissivity for diffuse radiation.

Units	Legal Range	Default	Required	Variability
fraction	*$0 \leq x \leq 1$*	none	yes	Constant

gtDMRBSol=*float*

SHGC diffuse multiplier, applied to qtSHGC to determine transmissivity for diffuse radiation reflected back out the window. Misnamed as a reflectance. Assume equal to DMSHGC if no other data available.

Units	Legal Range	Default	Required	Variability
fraction	$0 \leq x \leq 1$	none	yes	Constant

gtNGlz=*int*

Number of glazings in the Glazetype (bare glass only, not including any interior or exterior shades).

Units	Legal Range	Default	Required	Variability
	$0 < x \leq 4$	2	no	Constant

gtExShd=*choice*

Exterior shading type (ASHWAT only).

Units	Legal Range	Default	Required	Variability
	NONE INSCRN	NONE	no	Constant

gtInShd=*choice*

Interior shade type (ASHWAT only).

Units	Legal Range	Default	Required	Variability
	NONE DRAPEMED	NONE	no	Constant

gtDirtLoss=*float*

Glazing dirt loss factor.

Units	Legal Range	Default	Required	Variability
fraction	$0 \leq x \leq 1$	0	no	Constant

endGlazeType

Optional to indicates the end of the Glazetype. Alternatively, the end of the GLAZETYPE definition can be indicated by "END" or by beginning another object

Units	Legal Range	Default	Required	Variability
		N/A	No	constant

4.7 METER

A METER object is a user-defined "device" that records energy consumption of equipment as simulated by CSE. The user defines METERs with the desired names, then assigns energy uses of specific equipment to the desired meters using commands described under each equipment type's class description (AIRHANDLER, TERMINAL, etc.). Additional energy use from equipment not simulated by CSE (except optionally for its effect on heating and cooling loads) can also be charged to METERs (see GAIN). The data accumulated by meters can be reported at hourly, daily, monthly, and annual (run) intervals by using REPORTs and EXPORTs of type MTR.

Meters account for energy use in the following pre-defined categories, called *end uses*. The abbreviations in parentheses are used in MTR report headings (and for gnMeter input, below).

- Total use
- Space cooling use (Clg)
- Space heating use - including heat pump compressor (Htg)
- Heat pump backup heating (HPHtg)
- Hot water heating (DHW)
- Hot water heating backup - detailed HPWH model only (DHWBU)
- Fans – AC and cooling ventilation (FanC)
- Fans – heating (FanH)
- Fans – IAQ venting (FanV)
- Fans – other (Fan)
- HVAC auxiliary - not including fans (Aux)
- Process energy (Proc)
- Lighting (Lit)
- Receptacles (Rcp)
- Exterior (Ext)
- Refrigeration (Refr)
- Dish washing (Dish)
- Clothes drying (Dry)
- Clothes washing (Wash)
- Cooking (Cook)
- User defined 1 (User1)
- User defined 2 (User2)
- Photovoltaic generation (PV)

The user has complete freedom over how many meters are defined and how equipment is assigned to them. At one extreme, a single meter "Electricity" could be defined and have all of electrical uses assigned to it. On the other hand, definition of separate meters "Elect_Fan1", "Elect_Fan2", and so forth allows accounting of the electricity use for individual pieces of equipment. Various groupings are possible: for example, in a building with several air handlers, one could separate the energy consumption of the fans from the coils, or one could separate the energy use by air handler, or both ways, depending on the information desired from the run.

The members that assign energy use to meters include:

- GAIN: gnMeter, gnEndUse
- ZONE: xfanMtr
- IZXFER: izfanMtr
- RSYS: rsElecMtr, rsFuelMtr
- DHWSYS: wsElecMtr, wsFuelMtr
- DHWHEATER: whElectMtr, whFuelMtr
- DHWPUMP: wpElecMtr
- DHWLOOPPUMP: wlpElecMtr
- PVARRAY: pvElecMeter
- TERMINAL: tuhcMtr, tfanMtr
- AIRHANDLER: sfanMtr, rfanMtr, ahhcMtr, ahccMtr, ahhcAuxOnMtr, ahhcAuxOffMtr, ahhcAuxFullOnMtr, ahhcAuxOnAtAllMtr, ahccAuxOnMtr, ahccAuxOffMtr, ahccAuxFullOnMtr, ahccAuxOnAtAllMtr
- BOILER: blrMtr, blrpMtr, blrAuxOnMtr, blrAuxOffMtr, blrAuxFullOnMtr, blrAuxOnAtAllMtr

- CHILLER: chMtr, chppMtr, chcpMtr, chAuxOnMtr, chAuxOffMtr, chAuxFullOnMtr, chAuxOnAtAllMtr
- TOWERPLANT: tpMtr

The end use can be specified by the user only for GAINs and PVARRAYs; in other cases it is hard-wired to Clg, Htg, FanC, FanH, FanV, Fan, or Aux as appropriate.

mtrName

Name of meter: required for assigning energy uses to the meter elsewhere.

Units	Legal Range	Default	Required	Variability
	63 characters	*none*	Yes	constant

endMeter

Indicates the end of the meter definition. Alternatively, the end of the meter definition can be indicated by the declaration of another object or by END.

Units	Legal Range	Default	Required	Variability
		N/A	No	constant

4.8 DHWMETER

A DHWMETER object is a user-defined "device" that records water consumption as simulated by CSE. The data accumulated by DHWMETERs can be reported at hourly, daily, monthly, and annual (run) intervals by using REPORTs and EXPORTs of type DHWMTR.

DHWMETERs account for water use in the following pre-defined end uses. The abbreviations in parentheses are used in DHWMTR report headings.

- Total water use (Total)
- Unknown end use (Unknown)
- Miscellaneous draws (Faucet)
- Shower (Shower)
- Bathtub (Bath)
- Clothes washer (CWashr)
- Dishwasher (DWashr)

DHWSYS items wsWHhwMtr and wsFXhwMtr specify the DHWMETER(s) to which water consumption is accumulated.

dhwMtrName

Name of meter: required for assigning water uses to the DHWMETER.

Units	Legal Range	Default	Required	Variability
	63 characters	*none*	Yes	constant

endDhwMeter

4.9 ZONE

ZONE constructs an object of class ZONE, which describes an area of the building to be modeled as having a uniform condition. ZONEs are large, complex objects and can have many subobjects that describe associated surfaces, shading devices, HVAC equipment, etc.

4.9.1 ZONE General Members

znName

Name of zone. Enter after the word ZONE; no "=" is used.

Units	Legal Range	Default	Required	Variability
	63 characters	*none*	Yes	constant

znModel=*choice*

Selects model for zone.

CNE	Older, central difference model based on original CALPAS methods. Not fully supported and not suitable for current compliance applications.
CZM	Conditioned zone model. Forward-difference, short time step methods are used.
UZM	Unconditioned zone model. Identical to CZM except heating and cooling are not supported. Typically used for attics, garages, and other ancillary spaces.

Units	Legal Range	Default	Required	Variability
	choices above	CNE	No	constant

znArea=*float*

Nominal zone floor area.

Units	Legal Range	Default	Required	Variability
ft^2	$x > 0$	*none*	Yes	constant

znVol=*float*

Nominal zone volume.

Units	Legal Range	Default	Required	Variability
ft^3	$x > 0$	*none*	Yes	constant

znAzm=*float*

Zone azimuth with respect to bldgAzm. All surface azimuths are relative to znAzm, so that the zone can be rotated by changing this member only. Values outside the range 0° to 360° are normalized to that range.

Units	Legal Range	Default	Required	Variability
degrees	unrestricted	0	No	constant

znFloorZ=*float*

Nominal zone floor height relative to arbitrary 0 level. Used re determination of vent heights

Units	Legal Range	Default	Required	Variability
ft	unrestricted	*0*	No	constant

znCeilingHt=*float*

Nominal zone ceiling height relative to zone floor (typically 8 – 10 ft).

Units	Legal Range	Default	Required	Variability
ft	$x > 0$	*znVol / znArea*	No	constant

znEaveZ=*float*

Nominal eave height above ground level. Used re calculation of local surface wind speed. This in turn influences outside convection coefficients in some surface models and wind-driven air leakage.

Units	Legal Range	Default	Required	Variability
ft	$x \geq 0$	*znFloorZ + infStories*8*	No	constant

znCAir=*float*

Zone "air" heat capacity: represents heat capacity of air, furniture, "light" walls, and everything in zone except surfaces having heat capacity (that is, non-QUICK surfaces).

Units	Legal Range	Default	Required	Variability
Btu/°F	$x \geq 0$	3.5 * *znArea*	No	constant

znHcAirX=*float*

Zone air exchange rate used in determination of interior surface convective coefficients. This item is generally used only for model testing.

Units	Legal Range	Default	Required	Variability
ACH	x ≥ 0	as modeled	No	subhourly

znHcFrcF=*float*

Zone surface forced convection factor. Interior surface convective transfer is modeled as a combination of forced and natural convection. hcFrc = znHcFrcF * znHcAirX^.8. See CSE Engineering Documentation.

Units	Legal Range	Default	Required	Variability
Btuh/ft^2-°F		.2	No	hourly

znHIRatio=*float*

Zone hygric inertia ratio. In zone moisture balance calculations, the effective dry-air mass = znHIRatio * (zone dry air mass). This enhancement can be used to represente the moisture storage capacity of zone surfaces and contents.

Units	Legal Range	Default	Required	Variability
	$x > 0$	1	No	constant

znSC=*float*

Zone shade closure. Determines insolation through windows (see WINDOW members *wnSCSO* and *wnSCSC*) and solar gain distribution: see SGDIST members *sgFSO* and *sgFSC*. 0 represents shades open; 1 represents shades closed; intermediate values are allowed. An hourly variable CSE expression may be used to schedule shade closure as a function of weather, time of year, previous interval HVAC use or zone temperature, etc.

Units	Legal Range	Default	Required	Variability
	$0 \leq x \leq 1$	1 when cooling was used in *previous* hour, else 0	No	hourly

znTH=*float*

Heating set point for znModel=CZM.

Units	Legal Range	Default	Required	Variability
°F	$x \geq 0$			hourly

znTD=*float*

Desired set point (temperature maintained with ventilation if possible) for znModel=CZM

Units	Legal Range	Default	Required	Variability
°F	$x \geq 0$			hourly

znTC=*float*

Cooling set point for znModel=CZM.

Units	Legal Range	Default	Required	Variability
°F	$x \geq 0$			Hourly

CZM zone heating and cooling is provided either via an RSYS HVAC system or by "magic" heat transfers specified by znQxxx items.

znRSys=*rsysName*

Name of RSYS providing heating, cooling, and optional central fan integrated ventilation to this zone.

Units	Legal Range	Default	Required	Variability
	RSYS name	(no RSYS)	No	constant

znQMxH=*float*

Heating capacity at current conditions

Units	Legal Range	Default	Required	Variability
Btuh	$x \geq 0$			hourly

znQMxHRated=*float*

Rated heating capacity

Units	Legal Range	Default	Required	Variability
Btuh	$x \geq 0$			constant

znQMxC=*float*

Cooling capacity at current conditions

Units	Legal Range	Default	Required	Variability
Btuh	$x \leq 0$			hourly

znQMxCRated=*float*

Rated cooling capacity

Units	Legal Range	Default	Required	Variability
Btuh	$x \leq 0$			constant

4.9.2 ZONE Infiltration

The following control a simplified air change plus leakage area model. The Sherman-Grimsrud model is used to derive air flow rate from leakage area and this rate is added to the air changes specified with infAC. Note that TOP.windF does *not* modify calculated infiltration rates, since the Sherman-Grimsrud model uses its own modifiers. See also AirNet models available via IZXFER.

infAC=*float*

Zone infiltration air changes per hour.

Units	Legal Range	Default	Required	Variability
1/hr	$x \geq 0$	0.5	No	hourly

infELA=*float*

Zone effective leakage area (ELA).

Units	Legal Range	Default	Required	Variability
in^2	$x \geq 0$	0.0	No	hourly

infShld=*int*

Zone local shielding class, used in derivation of local wind speed for ELA infiltration model, wind-driven AirNet leakage, and exterior surface coefficients. infShld values are –

1	no obstructions or local shielding
2	light local shielding with few obstructions
3	moderate local shielding, some obstructions within two house heights
4	heavy shielding, obstructions around most of the perimeter
5	very heavy shielding, large obstructions surrounding the perimeter within two house heights

Units	Legal Range	Default	Required	Variability
	$1 \leq x \leq 5$	3	No	constant

infStories=*int*

Number of stories in zone, used in ELA model.

Units	Legal Range	Default	Required	Variability
	$1 \leq x \leq 3$	1	No	constant

znWindFLkg=*floatTODO*

Units	Legal Range	Default	Required	Variability
		1	No	constant

4.9.3 ZONE Exhaust Fan

Presence of an exhaust fan in a zone is indicated by specifying a non-zero design flow value (xfanVfDs).

Zone exhaust fan model implementation is incomplete as of July, 2011. The current code calculates energy use but does not account for the effects of air transfer on room heat balance. IZXFER provides a more complete implementation.

xfanFOn=*float*

Exhaust fan on fraction. On/off control assumed, so electricity requirement is proportional to run time.

Units	Legal Range	Default	Required	Variability
fraction	0 ≤ x ≤ 1	1	No	hourly

Example: The following would run an exhaust fan 70% of the time between 8 AM and 5 PM:

```
xfanFOn = select( (\$hour >= 7 && \$hour < 5), .7,
```

```
default, 0 );
```

xfanVfDs=*float*

Exhaust fan design flow; 0 or not given indicates no fan.

Units	Legal Range	Default	Required	Variability
cfm	x 0	0 (no fan)	If fan present	constant

xfanPress=*float*

Exhaust fan external static pressure.

Units	Legal Range	Default	Required	Variability
inches H_2O	$0.05 \leq x \leq 1.0$	0.3	No	constant

Only one of xfanElecPwr, xfanEff, and xfanShaftBhp may be given: together with xfanVfDs and xfanPress, any one is sufficient for CSE to detemine the others and to compute the fan heat contribution to the air stream.

xfanElecPwr=*float*

Fan input power per unit air flow (at design flow and pressure).

Units	Legal Range	Default	Required	Variability
W/cfm	$x > 0$	derived from xfanEff and xfanShaftBhp	If xfanEff and xfanShaftBhp not present	constant

xfanEff=*float*

Exhaust fan/motor/drive combined efficiency.

Units	Legal Range	Default	Required	Variability
fraction	$0 \leq \text{x} \leq 1$	0.08	No	constant

xfanShaftBhp=*float*

Fan shaft power at design flow and pressure.

Units	Legal Range	Default	Required	Variability
BHP	$x > 0$	derived from xfanElecPwr and xfanVfDs	If xfanElecPwr not present	constant

xfanMtr=*mtrName*

Name of METER object, if any, by which fan's energy use is recorded (under end use category "fan").

Units	Legal Range	Default	Required	Variability
	name of a METER	*not recorded*	No	constant

endZone

Indicates the end of the zone definition. Alternatively, the end of the zone definition can be indicated by the declaration of another object or by "END". If END or endZone is used, it should follow the definitions of the ZONE's subobjects such as GAINs, SURFACEs, TERMINALs, etc.

Units	Legal Range	Default	Required	Variability
		N/A	No	constant

4.10 GAIN

A GAIN object adds sensible and/or latent heat to the ZONE, and/or adds arbitrary energy use to a METER. GAINs may be subobjects of ZONEs and are normally given within the input for their ZONE. As many GAINs as desired (or none) may be given for each ZONE. Alternatively, GAINs may be subobjects of TOP and specify gnZone to specify their associate zone.

Each gain has an amount of power (gnPower), which may optionally be accumulated to a METER (gnMeter). The power may be distributed to the zone, plenum, or return as sensible heat with an optionl fraction radiant, or to the zone as latent heat (moisture addition), or not.

gnName

Name of gain; follows the word GAIN if given.

Units	Legal Range	Default	Required	Variability
	63 characters	*none*	No	constant

gnZone=*znName*

Name of ZONE to which heat gains are added. Omitted when GAIN is given as a ZONE subobject. If a TOP subobject (i.e., not a ZONE subobject) and znZone is omitted, heat gains are discarded but energy use is still recorded to gnMeter. This feature can be used to represent energy uses that our outside of conditioned zones (e.g. exterior lighting).

Units	Legal Range	Default	Required	Variability
	name of ZONE	*parent zone if any*	No	constant

gnPower=*float*

Rate of heat addition/energy use. Negative gnPower values may be used to represent heat removal/energy generation. Expressions containing functions are commonly used with this member to schedule the gain power on a daily and/or hourly basis. Refer to the functions section in Section 4 for details and examples.

All gains, including electrical, are specified in Btuh units unless associated with DHW use (see gnCtrlDHWSYS), in which case gnPower is specified in Btuh/gal. Note that meter reporting of internal gain is in MBtu (millions of Btu) by default.

Units	Legal Range	Default	Required	Variability
Btuh	*no restrictions*	*none*	Yes	hourly

Units	Legal Range	Default	Required	Variability

gnMeter=*choice*

Name of meter by which this GAIN's gnPower is recorded. If omitted, gain is assigned to no meter and energy use is not accounted in CSE simulation reports; thus, gnMeter should only be omitted for "free" energy sources.

Units	Legal Range	Default	Required	Variability
	name of METER	*none*	No	constant

gnEndUse=*choice*

Meter end use to which the GAIN's energy use should be accumulated.

Clg	Cooling
Htg	Heating (includes heat pump compressor)
HPHTG	Heat pump backup heat
DHW	Domestic (service) hot water
DHWBU	Domestic (service) hot water heating backup (HPWH resistance)
FANC	Fans, AC and cooling ventilation
FANH	Fans, heating
FANV	Fans, IAQ venting
FAN	Fans, other purposes
AUX	HVAC auxiliaries such as pumps
PROC	Process
LIT	Lighting
RCP	Receptacles
EXT	Exterior lighting
REFR	Refrigeration
DISH	Dishwashing
DRY	Clothes drying
WASH	Clothes washing
COOK	Cooking
USER1	User-defined category 1
USER2	User-defined category 2
PV	Photovoltaic power generation

Units	Legal Range	Default	Required	Variability
	Codes listed above	*none*	Required if gnMeter is given	constant

The gnFrZn, gnFrPl, and gnFrRtn members allow you to allocate the gain among the zone, the zone's plenum, and the zone's return air flow. Values that total to more than 1.0 constitute an error. If they total less than 1, the unallocated portion of the gain is recorded by the meter (if specified) but not transferred into the building. By default, all of the gain not directed to the return or plenum goes to the zone.

gnFrZn=*float*

Fraction of gain going to zone. gnFrLat (below) gives portion of this gain that is latent, if any; the remainder is sensible.

Units	Legal Range	Default	Required	Variability
	$0 \leq x \leq 1$	1.	No	hourly

Gain Modeling in zones

The radiant internal gain is distributed to the surfaces in the zone, rather than going directly to the zone "air" heat capacity (znCAir). A simple model is used – all surfaces are assumed to be opaque and to have the same (infrared) absorptivity – even windows. Along with the assumption that the zone is spherical (implicit in the current treatment of solar gains), this allows distribution of gains to surfaces in proportion to their area, without any absorptivity or transmissivity calculations. The gain for windows and quick-model surfaces is assigned to the znCAir, except for the portion which conducts through the surface to the other side rather than through the surface film to the adjacent zone air; the gain to massive (delayed-model) surfaces is assigned to the side of surface in the zone with the gain.

Radiant internal gains are included in the IgnS (Sensible Internal Gain) column in the zone energy balance reports. (They could easily be shown in a separate IgnR column if desired.) Any energy transfer shows two places in the ZEB report, with opposite signs, so that the result is zero – otherwise it wouldn't be an energy balance. The rest of the reporting story for radiant internal gains turns out to be complex. The specified value of the radiant gain (gnPower * gnFrZn * gnFrRad) shows in the IgnS column. To the extent that the gain heats the zone, it also shows negatively in the Masses column, because the zone CAir is lumped with the other masses. To the extent that the gain heats massive surfaces, it also shows negatively in the masses column. To the extent that the gain conducts through windows and quick-model surfaces, it shows negatively in the Conduction column. If the gain conducts through a quick-model surface to another zone, it shows negatively in the Izone (Interzone) column, positively in the Izone column of the receiving zone, and negatively in the receiving zone's Masses or Cond column.

gnFrRad=*float*

Fraction of total gain going to zone (gnFrZn) that is radiant rather than convective or latent.

Units	Legal Range	Default	Required	Variability
	$0 \leq x \leq 1$	0.	No	hourly

gnFrLat=*float*

Fraction of total gain going to zone (gnFrZn) that is latent heat (moisture addition).

Units	Legal Range	Default	Required	Variability
	$0 \leq \mathrm{x} \leq 1$	0.	No	hourly

gnDlFrPow=*float*

Hourly power reduction factor, typically used to modify lighting power to account for daylighting.

Units	Legal Range	Default	Required	Variability
	$0 \leq x \leq 1$	1.	No	hourly

gnCtrlDHWSYS=*dhwsysName*

Name of a DHWSYS whose water use modulates gnPower. For example, electricity use of water-using appliances (e.g. dishwasher or clothes washer) can be modeled based on water use, ensuring that the uses are synchronized. When this feature is used, gnPower should be specified in Btuh/gal.

Units	Legal Range	Default	Required	Variability
	name of a DHWSYS	no DHWSYS/GAIN linkage	No	constant

gnCtrlDHWEndUse=*dhwEndUseName*

Name of the DHWSYS end use consumption that modulates gnPower. See DHWMETER for DHW end use definitions.

Units	Legal Range	Default	Required	Variability
	DHW end use	Total	No	constant

endGain

Optional to indicate the end of the GAIN definition. Alternatively, the end of the gain definition can be indicated by END or by the declaration of another object.

Units	Legal Range	Default	Required	Variability
		N/A	No	constant

4.11 SURFACE

Surface constructs a ZONE subobject of class SURFACE that represents a surrounding or interior surface of the zone. Internally, SURFACE generates a QUICK surface (U-value only), a DELAYED (massive) surface (using the finite-difference mass model), interzone QUICK surface, or interzone DELAYED surface, as appropriate for the specified construction and exterior conditions.

sfName

Name of surface; give after the word SURFACE.

Units	Legal Range	Default	Required	Variability
	63 characters	*none*	No	constant

sfType=*choice*

Type of surface:

FLOOR	Surface defines part or all of the "bottom" of the zone; it is horizontal with inside facing up. The outside of the surface is not adjacent to the current zone.
WALL	Surface defines a "side" of the zone; its outside is not adjacent to the current zone.
CEILING	Surface defines part or all of the "top" of the zone with the inside facing down. The outside of the surface is not adjacent to the current zone.

sfType is used extensively for default determination and input checking, but does not have any further internal effect. The Floor, Wall, and Ceiling choices identify surfaces that form boundaries between the zone and some other condition.

Units	Legal Range	Default	Required	Variability
	FLOOR WALL CEILING	*none*	Yes	constant

sfArea=*float*

Gross area of surface. (CSE computes the net area for simulation by subtracting the areas of any windows and doors in the surface.).

Units	Legal Range	Default	Required	Variability
ft^2	$x > 0$	*none*	Yes	constant

sfTilt=*float*

Surface tilt from horizontal. Values outside the range 0 to 360 are first normalized to that range. The default and allowed range depend on sfType, as follows:

sfType = FLOOR	*sfTilt*=180, default = 180 (fixed value)
sfType = WALL	60 < *sfTilt* < 180, default = 90
sfType = CEILING	0 ≤ *sfTilt* ≤ 60, default = 0

Units	Legal Range / Default	Required	Variability
degrees	Dependent upon *sfType. See above*	No	constant

sfAzm=*float*

Azimuth of surface with respect to znAzm. The azimuth used in simulating a surface is bldgAzm + znAzm + sfAzm; the surface is rotated if any of those are changed. Values outside the range 0 to 360 are normalized to that range. Required for non-horizontal surfaces.

Units	Legal Range	Default	Required	Variability
degrees	unrestricted	*none*	Required if *sfTilt* ≠ 0 and *sfTilt* ≠ 180	constant

sfModel=*choice*

Provides user control over how CSE models conduction for this surface.

QUICK	Surface is modeled using a simple conductance. Heat capacity effects are ignored. Either sfCon or sfU (next) can be specified.
DELAYED, DELAYED_HOUR, DELAYED_SUBHOUR	Surface is modeled using a multi-layer finite difference technique that represents heat capacity effects. If the time constant of the surface is too short to accurately simulate, a warning message is issued and the Quick model is used. The program **cannot** use the finite difference model if sfU rather than sfCon is specified.

AUTO	Program selects Quick or the appropriate Delayed automatically according to the time constant of the surface (if sfU is specified, Quick is selected).
FD (or FORWARD_DIFFERENCE)	Selects the forward difference model (used with short time steps and the CZM/UZM zone model)

Units	Legal Range	Default	Required	Variability
–	QUICK, DELAYED, DELAYED_HOUR, DE-LAYED_SUBOUR, AUTO, FD	AUTO	No	constant

Either sfU or sfCon must be specified, but not both.

sfU=*float*

Surface U-value (NOT including surface (air film) conductances). For surfaces for which no heat capacity is to be modeled, allows direct entry of U-value without defining a CONSTRUCTION.

Units	Legal Range	Default	Required	Variability
Btuh/ft^2-°F	$x > 0$	Determined from *sfCon*	if *sfCon* not given	constant

sfCon=*conName*

Name of CONSTRUCTION of the surface.

Units	Legal Range	Default	Required	Variability
	Name of a *CONSTRUCTION*	*none*	unless *sfU* given	constant

sfLThkF=*float*

Sublayer thickness adjustment factor for FORWARD_DIFFERENCE conduction model used with sfCon surfaces. Material layers in the construction are divided into sublayers as needed for numerical stability. sfLThkF allows adjustment of the thickness criterion used for subdivision. A value of 0 prevents subdivision; the default value (0.5) uses layers with conservative thickness equal to half of an estimated safe value. Fewer (thicker) sublayers improves runtime at the expense of accurate representation of rapid changes.

Units	Legal Range	Default	Required	Variability
	x $\geq$ 0	.5	No	constant

sfExCnd=*choice*

Specifies the thermal conditions assumed at surface exterior, and at exterior of any subobjects (windows or doors) belonging to current surface. The conditions accounted for are dry bulb temperature and incident solar radiation.

AMBIENT	Exterior surface is exposed to the "weather" as read from the weather file. Solar gain is calculated using solar geometry, sfAzm, sfTilt, and sfExAbs.
SPECIFIEDT	Exterior surface is exposed to solar radiation as in AMBIENT, but the dry bulb temperature is calculated with a user specified function (sfExT). sfExAbs can be set to 0 to eliminate solar effects.
ADJZN	Exterior surface is exposed to another zone, whose name is specified by sfAdjZn. Solar gain is 0 unless gain is targeted to the surface with SGDIST below.
ADIABATIC	Exterior surface heat flow is 0. Thermal storage effects of delayed surfaces are modeled.

sfExAbs=*float*

Surface exterior absorptivity.

Units	Legal Range	Default	Required	**Variabili ty
(none)	$0 \leq x \leq 1$	0.5	Required if *sfExCnd* = AMBIENT or *sfExCnd* = SPECIFIEDT	monthly-hourly

sfInAbs=*float*

Surface interior solar absorptivity.

Units	Legal Range	Default	Required	**Variabili ty
(none)	$0 \leq x \leq 1$	sfType = CEILING, 0.2; sfType = WALL, 0.6; sfType = FLOOR, 0.8	No	monthly-hourly

sfExEpsLW=*float*

Surface exterior long wave (thermal) emittance.

Units	Legal Range	Default	Required	Variability
(none)	$0 \leq x \leq 1$	0.9	No	constant

sfInEpsLW=*float*

Surface interior long wave (thermal) emittance.

Units	Legal Range	Default	Required	Variability
(none)	$0 \leq x \leq 1$	0.9	No	constant

sfExT=*float*

Exterior air temperature.

Units	Legal Range	Default	Required	Variability
°F	*unrestricted*	*none*	Required if *sfExCnd* = SPECIFIEDT	hourly

sfAdjZn=*znName*

Name of adjacent zone; used only when sfExCnd is ADJZN. Can be the same as the current zone.

Units	Legal Range	Default	Required	Variability
	name of a *ZONE*	*none*	Required when*sfExCnd* = ADJZN	constant

sfGrndRefl=*float*

Ground reflectivity for this surface.

Units	Legal Range	Default	Required	Variability
fraction	$0 \leq x \leq 1$	grndRefl	No	Monthly - Hourly

sfInH=*float*

Inside surface (air film) conductance. Ignored for sfModel = Forward_Difference. Default depends on the surface type.

sfType = FLOOR or CEILING	1.32
other	1.5

Units	Legal Range	Default	Required	Variability
Btuh/ft^2-°F	$x > 0$	*see above*	No	constant

sfExH=*float*

Outside combined surface (air film) conductance. Ignored for sfModel = Forward_Difference. The default value is dependent upon the exterior conditions:

sfExCnd = AMBIENT	dflExH (Top-level member, described above)
sfExCnd = SPECIFIEDT	dflExH (described above)
sfExCnd = ADJZN	1.5
sfExCnd = ADIABATIC	not applicable

Units	Legal Range	Default	Required	Variability
Btuh/ft^2-°F	$x > 0$	see above	No	constant

When sfModel = Forward_Difference, several models are available for calculating inside and outside surface convective coefficients. Inside surface faces can be exposed only to zone conditions. Outside faces may be exposed either to ambient conditions or zone conditions, based on sfExCnd. Only UNIFIED and INPUT are typically used. The other models were used during CSE development for comparison. For details, see CSE Engineering Documentation.

Model	Exposed to ambient	Exposed to zone
UNIFIED	default CSE model	default CSE model
INPUT	hc = sfExHcMult	hc = sfxxHcMult
AKBARI	Akbari model	n/a
WALTON	Walton model	n/a
WINKELMANN	Winkelmann model	n/a
MILLS	n/a	Mills model
ASHRAE	n/a	ASHRAE handbook values

sfExHcModel=*choice*

Selects the model used for exterior surface convection when sfModel = Forward_Difference.

Units	**Legal Range**	**Default**	**Required**	**Variability**
	choices above	UNIFIED	No	constant

sfExHcLChar=*float*

Characteristic length of surface, used in derivation of forced exterior convection coefficients in some models when outside surface is exposed to ambient. See sfExHcModel.

Units	**Legal Range**	**Default**	**Required**	**Variability**
ft	x > 0	10	No	constant

sfExHcMult=*float*

Exterior convection coefficient adjustment factor. When sfExHcModel=INPUT, hc=sfExHcMult. For other sfExHcModel choices, the model-derived hc is multiplied by sfExHcMult.

Units	**Legal Range**	**Default**	**Required**	**Variability**
		1	No	subhourly

sfExRf=*float*

Exterior surface roughness factor. Used only when surface is exposed to ambient (i.e. with wind exposure). Typical values:

Roughness Index	sfExRf	Example
1 (very rough)	2.17	Stucco
2 (rough)	1.67	Brick
3 (medium rough)	1.52	Concrete
4 (Medium smooth)	1.13	Clear pine
5 (Smooth)	1.11	Smooth plaster
6 (Very Smooth)	1	Glass

Units	Legal Range	Default	**Require d	Variabilit y
		sfExHcModel = WINKELMANN: 1.66 else 2.17	No	constant

sfInHcModel=*choice*

Selects the model used for the inside (zone) surface convection when sfModel = Forward_Difference.

Units	Legal Range	Default	Required	Variability
	choices above (see sfExHcModel)	UNIFIED	No	constant

sfInHcMult=*float*

Interior convection coefficient adjustment factor. When sfInHcModel=INPUT, hc=sfInHcMult. For other sfInHcModel choices, the model-derived hc is multiplied by sfInHcMult.

Units	Legal Range	Default	Required	Variability
		1	No	subhourly

The items below give values associated with CSE's model for below grade surfaces (sfExCnd=GROUND). See CSE Engineering Documentation for technical details.

sfDepthBG=*float*

Depth below grade of surface. For walls, sfDepthBG is measured to the lower edge. For floors, sfDepthBG is measured to the bottom face.

Units	Legal Range	Default	Required	Variability
ft	$x \geq 0$		No	constant

sfExCTGrnd=*float*

sfExCTaDbAvg07=*float*

sfExCTaDbAvg14=*float*

sfExCTaDbAvg31=*float*

sfExCTaDbAvgYr=*float*

Conductances from outside face of surface to the weather file ground temperature and the moving average outdoor dry-bulb temperatures for 7, 14, 31, and 365 days.

Units	Legal Range	Default	Required	Variability
Btuh/ft^2-^{o}F	$x \geq 0$	see above	No	constant

sfExRConGrnd=*float*

Resistance overall construction resistance. TODO: full documentation.

Units	**Legal Range**	**Default**	**Required**	**Variability**

Units	**Legal Range**	**Default**	**Required**	**Variability**
	$x \geq 0$		No	constant

endSURFACE

Optional to indicates the end of the surface definition. Alternatively, the end of the surface definition can be indicated by END, or by beginning another SURFACE or other object definition. If used, should follow the definitions of the SURFACE's subobjects – DOORs, WINDOWs, SHADEs, SGDISTs, etc.

Units	**Legal Range**	**Default**	**Required**	**Variability**
		N/A	No	constant

4.12 WINDOW

WINDOW defines a subobject belonging to the current SURFACE that represents one or more identical windows. The azimuth, tilt, and exterior conditions of the window are the same as those of the surface to which it belongs. The total window area ($wnHt \cdot wnWid \cdot wnMult$) is deducted from the gross surface area. A surface may have any number of windows.

Windows may optionally have operable interior shading that reduces the overall shading coefficient when closed.

wnName

Name of window: follows the word "WINDOW" if given.

Units	**Legal Range**	**Default**	**Required**	**Variability**
	63 characters	*none*	No	constant

wnHeight=*float*

Overall height of window (including frame).

Units	**Legal Range**	**Default**	**Required**	**Variability**
ft	$x > 0$	*none*	Yes	constant

wnWidth=*float*

Overall width of window (including frame).

Units	**Legal Range**	**Default**	**Required**	**Variability**
ft	$x > 0$	*none*	Yes	constant

wnArea=*float*

Overall area of window (including frame).

Units	Legal Range	Default	Required	Variability
ft^2	$x > 0$	*wnHeight* * *wnWidth*	No	constant

wnMult=*float*

Area multiplier; can be used to represent multiple identical windows.

Units	Legal Range	Default	Required	Variability
	$x > 0$	1	No	constant

wnModel=*choice*

Selects window model

Units	Legal Range	Default	Required	Variability
	SHGC, ASHWAT	SHGC	No	constant

wnGt=*choice*

GLAZETYPE for window. Provides many defaults for window properties as cited below.

wnU=*float*

Window conductance (U-factor without surface films, therefore not actually a U-factor but a C-factor).

Preferred Approach: To use accurately with standard winter rated U-factor from ASHRAE or NFRC enter as:

```
wnU = (1/((1/U-factor)-0.85)
```

Where 0.85 is the sum of the interior (0.68) and exterior (0.17) design air film resistances assumed for rating window U-factors. Enter wnInH (usually 1.5=1/0.68) instead of letting it default. Enter the wnExH or let it default. It is important to use this approach if the input includes gnFrad for any gain term. Using approach 2 below will result in an inappropriate internal gain split at the window.

Approach 2. Enter wnU=U-factor and let the wnInH and wnExH default. Tnormally this approach systematically underestimates the window U-factor because it adds the wnExfilm resistance to 1/U-factor thereby double counting the exterior film resistance. This approach will also yield incorrect results for gnFrad internal gain since the high wnInH will put almost all the gain back in the space.

Units	Legal Range	Default	Required	Variability
Btuh/ft^2-°F	$x > 0$	*none*	Yes	constant

wnUNFRC=*float*

Fenestration system (including frame) U-factor evaluated at NFRC heating conditions.

Units	Legal Range	Default	Required	Variability
Btuh/ft^2-°F	$x > 0$	gtUNFRC	Required when *wnModel* = ASHWAT	constant

wnExEpsLW=*float*

Window exterior long wave (thermal) emittance.

Units	Legal Range	Default	Required	Variability
(none)	$0 \leq x \leq 1$	0.84	No	constant

wnInEpsLW=*float*

Window interior long wave (thermal) emittance.

Units	Legal Range	Default	Required	Variability
(none)	$0 \leq x \leq 1$	0.84	No	constant

wnInH=*float*

Window interior surface (air film) conductance.

Preferred Approach: Enter the appropriate value for each window, normally:

```
wnInH = 1.5

where 1.5 = 1/0.68 the standard ASHRAE value.
```

The large default value of 10,000 represents a near-0 resistance, for the convenience of those who wish to include the interior surface film in wnU according to approach 2 above.

Units	Legal Range	Default	Required	Variability
Btuh/ft^2-^{o}F	$x > 0$	10000	No	constant

wnExH=*float*

Window exterior surface (air film) conductance.

Units	Legal Range	Default	Required	Variability
Btuh/ft^2-^{o}F	x > 0	same as owning surface	No	constant

Several models are available for calculating inside and outside surface convective coefficients. Inside surface faces can be exposed only to zone conditions. Outside faces may be exposed either to ambient conditions or zone conditions, based on wnExCnd. Only UNIFIED and INPUT are typically used. The other models were used during CSE development for comparison. For details, see CSE Engineering Documentation.

Model	Exposed to ambient	Exposed to zone
UNIFIED	default CSE model	default CSE model
INPUT	hc = wnExHcMult	hc = wnxxHcMult
AKBARI	Akbari model	n/a
WALTON	Walton model	n/a
WINKELMANN	Winkelmann model	n/a
MILLS	n/a	Mills model
ASHRAE	n/a	ASHRAE handbook values

wnExHcModel=*choice*

Selects the model used for exterior surface convection when wnModel = Forward_Difference.

Units	Legal Range	Default	Required	Variability
	choices above	UNIFIED	No	constant

wnExHcLChar=*float*

Characteristic length of surface, used in derivation of forced exterior convection coefficients in some models when outside face is exposed to ambient (i.e. to wind).

Units	Legal Range	Default	Required	Variability
ft	x > 0	10	No	constant

wnExHcMult=*float*

Exterior convection coefficient adjustment factor. When wnExHcModel=INPUT, hc=wnExHcMult. For other wnExHcModel choices, the model-derived hc is multiplied by wnExHcMult.

Units	Legal Range	Default	Required	Variability
		1	No	subhourly

wnInHcModel=*choice*

Selects the model used for the inside (zone) surface convection when wnModel = Forward_Difference.

Units	Legal Range	Default	Required	Variability
	choices above (see wnExHcModel)	UNIFIED	No	constant

wnInHcMult=*float*

Interior convection coefficient adjustment factor. When wnInHcModel=INPUT, hc=wnInHcMult. For other wnInHcModel choices, the model-derived hc is multiplied by wnInHcMult.

Units	Legal Range	Default	Required	Variability
		1	No	subhourly

wnSHGC=*float*

Rated Solar Heat Gain Coefficient (SHGC) for the window assembly.

Units	Legal Range	Default	Required	Variability
fraction	0 < x < 1	gtSHGC	No	constant

wnFMult=*float*

Frame area multiplier = areaGlaze / areaAssembly

Units	Legal Range	Default	Required	Variability
fraction	0 < x < 1	gtFMult or 1	No	constant

wnSMSO=*float*

SHGC multiplier with shades open. Overrides gtSMSO.

Units	Legal Range	Default	Required	Variability
fraction	$0 \leq x \leq 1$	gtSMSO or 1	No	Monthly - Hourly

wnSMSC=*float*

SHGC multiplier with shades closed. Overrides gtSMSC

Units	Legal Range	Default	Required	Variability
fraction	$0 \leq x \leq 1$	wnSMSO or gtSMSC	No	Monthly - Hourly

wnNGlz=*int*

Number of glazings in the window (bare glass only, not including any interior or exterior shades).

Units	Legal Range	Default	Required	Variability
	$0 < x \leq 4$	gtNGLZ	Required when *wnModel* = ASHWAT	Constant

wnExShd=*choice*

Exterior shading type (ASHWAT only).

Units	Legal Range	Default	Required	Variability
	NONE, INSCRN	gtExShd	no	Constant

wnInShd=*choice*

Interior shade type (ASHWAT only).

Units	Legal Range	Default	Required	Variability
	NONE, DRAPEMED	gtInShd	no	Constant

wnDirtLoss=*float*

Glazing dirt loss factor.

Units	Legal Range	Default	Required	Variability
fraction	$0 \leq x \leq 1$	0	no	Constant

wnGrndRefl=*float*

Ground reflectivity for this window.

Units	Legal Range	Default	Required	Variability
fraction	$0 \leq x \leq 1$	sfGrndRefl	No	Monthly - Hourly

wnVfSkyDf=*float*

View factor from this window to sky for diffuse radiation. For the shading effects of an overhang, a wnVfSkyDf value smaller than the default would be used

Units	Legal Range	Default	Required	Variability
fraction	$0 \leq x \leq 1$	0.5 - 0.5 * cos(tilt) = .5 for vertical surface	No	Monthly - Hourly

wnVfGrndDf=*float*

View factor from this window to ground for diffuse radiation. For the shading effects of a fin(s), both wnVfSkyDf and wnVfGrndDf would be used.

Units	Legal Range	Default	Required	Variability
fraction	$0 \leq x \leq 1$	0.5 + 0.5 * cos(tilt) = .5 for vertical surface	No	Monthly - Hourly

endWINDOW

Optionally indicates the end of the window definition. Alternatively, the end of the window definition can be indicated by END or the declaration of another object. END or endWindow, if used, should follow any subobjects of the window (SHADEs and/or SGDISTs).

Units	Legal Range	Default	Required	Variability
		N/A	No	constant

4.13 SHADE

SHADE constructs a subobject associated with the current WINDOW that represents fixed shading devices (overhangs and/or fins). A window may have at most one SHADE and only windows in vertical surfaces may have SHADEs. A SHADE can describe an overhang, a left fin, and/or a right fin; absence of any of these is specified by omitting or giving 0 for its depth. SHADE geometry can vary on a monthly basis, allowing modeling of awnings or other seasonal shading strategies.

shName

Name of shade; follows the word "SHADE" if given.

Units	Legal Range	Default	Required	Variability
	63 characters	*none*	No	constant

ohDepth=*float*

Depth of overhang (from plane of window to outside edge of overhang). A zero value indicates no overhang.

Units	Legal Range	Default	Required	Variability
ft	$x \geq 0$	0	No	monthly-hourly

ohDistUp=*float*

Distance from top of window to bottom of overhang.

Units	Legal Range	Default	Required	Variability
ft	$x \geq 0$	0	No	monthly-hourly

ohExL=*float*

Distance from left edge of window (as viewed from the outside) to the left end of the overhang.

Units	Legal Range	Default	Required	Variability
ft	$x \geq 0$	0	No	monthly-hourly

ohExR=*float*

Distance from right edge of window (as viewed from the outside) to the right end of the overhang.

Units	Legal Range	Default	Required	Variability
ft	$x \geq 0$	0	No	monthly-hourly

ohFlap=*float*

Height of flap hanging down from outer edge of overhang.

Units	Legal Range	Default	Required	Variability
ft	$x \geq 0$	0	No	monthly-hourly

lfDepth=*float*

Depth of left fin from plane of window. A zero value indicates no fin.

Units	Legal Range	Default	Required	Variability
ft	$x \geq 0$	0	No	monthly-hourly

lfTopUp=*float*

Vertical distance from top of window to top of left fin.

Units	Legal Range	Default	Required	Variability
ft	$x \geq 0$	0	No	monthly-hourly

lfDistL=*float*

Distance from left edge of window to left fin.

Units	Legal Range	Default	Required	Variability
ft	$x \geq 0$	0	No	monthly-hourly

lfBotUp=*float*

Vertical distance from bottom of window to bottom of left fin.

Units	Legal Range	Default	Required	Variability
ft	$x \geq 0$	0	No	monthly-hourly

rfDepth=*float*

Depth of right fin from plane of window. A 0 value indicates no fin.

Units	Legal Range	Default	Required	Variability
ft	$x \geq 0$	0	No	monthly-hourly

rfTopUp=*float*

Vertical distance from top of window to top of right fin.

Units	Legal Range	Default	Required	Variability
ft	$x \geq 0$	0	No	monthly-hourly

rfDistR=*float*

Distance from right edge of window to right fin.

Units	Legal Range	Default	Required	Variability
ft	$x \geq 0$	0	No	monthly-hourly

rfBotUp=*float*

Vertical distance from bottom of window to bottom of right fin.

Units	Legal Range	Default	Required	Variability
ft	$x \geq 0$	0	No	monthly-hourly

endShade

Optional to indicate the end of the SHADE definition. Alternatively, the end of the shade definition can be indicated by END or the declaration of another object.

Units	Legal Range	Default	Required	Variability
		N/A	No	constant

4.14 SGDIST

SGDIST creates a subobject of the current window that distributes a specified fraction of that window's solar gain to a specified delayed model (massive) surface. Any remaining solar gain (all of the window's solar gain if no SGDISTs are given) is added to the air of the zone containing the window. A window may have up to three SGDISTs; an error occurs if more than 100% of the window's gain is distributed.

Via members sgFSO and sgFSC, the fraction of the insolation distributed to the surface can be made dependent on whether the zone's shades are open or closed (see ZONE member znSC).

sgName

Name of solar gain distribution (follows "SGDIST" if given).

Units	Legal Range	Default	Required	Variability
	63 characters	*none*	No	constant

sgSurf=*sfName*

Name of surface to which gain is targeted.

If there is more than surface with the specified name: if one of the surfaces is in the current zone, it is used; otherwise, an error message is issued.

The specified surface must be modeled with the Delayed model. If gain is targeted to a Quick model surface, a warning message is issued and the gain is redirected to the air of the associated zone.

Units	Legal Range	Default	Required	Variability
	name of a *SURFACE*	*none*	Yes	constant

sgSide=*choice*

Designates the side of the surface to which the gain is to be targeted:

INTERIOR	Apply gain to interior of surface
EXTERIOR	Apply gain to exterior of surface

Units	Legal Range	Default	Required	Variability
	INTERIOR, EXTERIOR	Side of surface in zone containing window; or INTERIOR if both sides are in zone containing window.	Yes	constant

sgFSO=*float*

Fraction of solar gain directed to specified surface when the owning window's interior shading is in the open position (when the window's zone's shade closure (znSC) is 0).

Units	Legal Range	Default	Required	Variability
	$0 \leq x \leq 1$,and sum of window's sgFSO's ≤ 1	*none*	Yes	monthly-hourly

sgFSC=*float*

Fraction of solar gain directed to specified surface when the owning window's interior shading is in the closed position. If the zone's shades are partly closed (znSC between 0 and 1), a proportional fraction between sgFSO and sgFSC is used.

Units	Legal Range	Default	Required	Variability
	$0 \leq x \leq 1$, and sum of window's sgFSC's ≤ 1	*sgFSO*	No	monthly-hourly

endSGDist

Optionally indicates the end of the solar gain distribution definition. Alternatively, the end of the solar gain distribution definition can be indicated by END or by just beginning another object.

Units	Legal Range	Default	Required	Variability
		N/A	No	constant

4.15 DOOR

DOOR constructs a subobject belonging to the current SURFACE. The azimuth, tilt, ground reflectivity and exterior conditions associated with the door are the same as those of the owning surface, although the exterior surface conductance and the exterior absorptivity can be altered.

drName

Name of door.

Units	Legal Range	Default	Required	Variability
	63 characters	*none*	No	constant

drArea=*float*

Overall area of door.

Units	Legal Range	Default	Required	Variability
ft^2	$x > 0$	*none*	Yes	constant

drModel=*choice*

Provides user control over how CSE models conduction for this door:

QUICK	Surface is modeled using a simple conductance. Heat capacity effects are ignored. Either drCon or drU (next) can be specified.
DELAYED, DELAYED_HOUR, DELAYED_SUBOUR	Surface is modeled using a multi-layer finite difference technique which represents heat capacity effects. If the time constant of the door is too short to accurately simulate, a warning message is issued and the Quick model is used. drCon (next) must be specified – the program cannot use the finite difference model if drU rather than drCon is specified.
AUTO	Program selects Quick or appropriate Delayed automatically according to the time constant of the surface (if drU is specified, Quick is selected).

FD or FORWARD_DIFFERENCE	Selects the forward difference model (used with short time steps and the CZM/UZM zone models)

Units	Legal Range	Default	Required	Variability
	choices above	AUTO	No	constant

Either drU or drCon must be specified, but not both.

drU=*float*

Door U-value, NOT including surface (air film) conductances. Allows direct entry of U-value, without defining a CONSTRUCTION, when no heat capacity effects are to be modeled.

Units	Legal Range	Default	Required	Variability
Btuh/ft^2-^{o}F	$x > 0$	Determined from *drCon*	if *drCon* not given	constant

drCon=*conName*

Name of construction for door.

Units	Legal Range	Default	Required	Variability
	name of a *CONSTRUCTION*	*None*	unless *drU* given	constant

drLThkF=*float*

Sublayer thickness adjustment factor for FORWARD_DIFFERENCE conduction model used with drCon surfaces. Material layers in the construction are divided into sublayers as needed for numerical stability. drLThkF allows adjustment of the thickness criterion used for subdivision. A value of 0 prevents subdivision; the default value (0.5) uses layers with conservative thickness equal to half of an estimated safe value. Fewer (thicker) sublayers improves runtime at the expense of accurate representation of rapid changes.

Units	Legal Range	Default	Required	Variability
	x $\geq$ 0	.5	No	constant

drExAbs=*float*

Door exterior solar absorptivity. Applicable only if sfExCnd of owning surface is AMBIENT or SPECIFIEDT.

Units	Legal Range	Default	Required	Variability
Btuh/ft^2-^{o}F	$x > 0$	same as owning surface	No	monthly-hourly

drInAbs=*float*

Door interior solar absorptivity.

Units	Legal Range	Default	Required	Variability
(none)	$0 \leq x \leq 1$	0.5	No	monthly-hourly

Units	**Legal Range**	**Default**	**Required**	**Variability**

drExEpsLW=*float*

Door exterior long wave (thermal) emittance.

Units	**Legal Range**	**Default**	**Required**	**Variability**
(none)	$0 \leq x \leq 1$	0.9	No	constant

drInEpsLW=*float*

Door interior long wave (thermal) emittance.

Units	**Legal Range**	**Default**	**Required**	**Variability**
(none)	$0 \leq x \leq 1$	0.9	No	constant

drInH=*float*

Door interior surface (air film) conductance. Ignored if drModel = Forward_Difference

Units	**Legal Range**	**Default**	**Required**	**Variability**
Btuh/ft^2-^{o}F	$x > 0$	same as owning surface	No	constant

drExH=*float*

Door exterior surface (air film) conductance. Ignored if drModel = Forward_Difference

Units	**Legal Range**	**Default**	**Required**	**Variability**
Btuh/ft^2-^{o}F	$x > 0$	same as owning surface	No	constant

When drModel = Forward_Difference, several models are available for calculating inside and outside surface convective coefficients. Inside surface faces can be exposed only to zone conditions. Outside faces may be exposed either to ambient conditions or zone conditions, based on drExCnd. Only UNIFIED and INPUT are typically used. The other models were used during CSE development for comparison. For details, see CSE Engineering Documentation.

Model	Exposed to ambient	Exposed to zone
UNIFIED	default CSE model	default CSE model
INPUT	hc = drExHcMult	hc = drxxHcMult
AKBARI	Akbari model	n/a
WALTON	Walton model	n/a
WINKELMANN	Winkelmann model	n/a
MILLS	n/a	Mills model
ASHRAE	n/a	ASHRAE handbook values

drExHcModel=*choice*

Selects the model used for exterior surface convection when drModel = Forward_Difference.

Units	Legal Range	Default	Required	Variability
	choices above	UNIFIED	No	constant

drExHcLChar=*float*

Characteristic length of surface, used in derivation of forced exterior convection coefficients in some models when outside face is exposed to ambient (i.e. to wind).

Units	Legal Range	Default	Required	Variability
ft	x > 0	10	No	constant

drExHcMult=*float*

Exterior convection coefficient adjustment factor. When drExHcModel=INPUT, hc=drExHcMult. For other drExHcModel choices, the model-derived hc is multiplied by drExHcMult.

Units	Legal Range	Default	Required	Variability
		1	No	subhourly

drExRf=*float*

Exterior roughness factor. Typical roughness values:

Roughness Index	drExRf	Example
1 (very rough)	2.17	Stucco
2 (rough)	1.67	Brick
3 (medium rough)	1.52	Concrete
4 (Medium smooth)	1.13	Clear pine
5 (Smooth)	1.11	Smooth plaster
6 (Very Smooth)	1	Glass

Units	Legal Range	Default	Required	Variability
		drExHcModel = WINKELMANN: 1.66 else 2.17	No	constant

drInHcModel=*choice*

Selects the model used for the inside (zone) surface convection when drModel = Forward_Difference.

Units	Legal Range	Default	Required	Variability
	choices above (see drExHcModel)	UNIFIED	No	constant

drInHcMult=*float*

Interior convection coefficient adjustment factor. When drInHcModel=INPUT, hc=drInHcMult. For other drInHcModel choices, the model-derived hc is multiplied by drInHcMult.

Units	Legal Range	Default	Required	Variability
		1	No	subhourly

endDoor

Indicates the end of the door definition. Alternatively, the end of the door definition can be indicated by the declaration of another object or by END.

Units	Legal Range	Default	Required	Variability
		N/A	No	constant

4.16 PERIMETER

PERIMETER defines a subobject belonging to the current zone that represents a length of exposed edge of a (slab on grade) floor.

prName

Optional name of perimeter.

Units	Legal Range	Default	Required	Variability
	63 characters	*none*	No	constant

prLen=*float*

Length of exposed perimeter.

Units	Legal Range	Default	Required	Variability
ft	$x > 0$	*none*	Yes	constant

prF2=*float*

Perimeter conduction per unit length.

Units	Legal Range	Default	Required	Variability
Btuh/ft-°F	$x > 0$	*none*	Yes	constant

endPerimeter

Optionally indicates the end of the perimeter definition.

Units	Legal Range	Default	Required	Variability
		N/A	No	constant

4.17 IZXFER

IZXFER constructs an object that represents an interzone or zone/ambient heat transfer due to conduction and/or air transfer. The air transfer modeled by IZXFER transfers heat only; humidity transfer is not modeled as of July 2011. Note that SURFACE is the preferred way represent conduction between ZONEs.

The AIRNET types are used in a multi-cell pressure balancing model that finds zone pressures that produce net 0 mass flow into each zone. The model operates in concert with the znType=CZM or znType=UZM to represent ventilation strategies. During each time step, the pressure balance is found for two modes that can be thought of as "VentOff" (or infiltration-only) and "VentOn" (or infiltration+ventilation). The zone model then determines the ventilation fraction required to hold the desired zone temperature (if possible). AIRNET modeling methods are documented in the CSE Engineering Documentation.

Note that fan-driven types assume pressure-independent flow. That is, the specified flow is included in the zone pressure balance but the modeled fan flow does not change with zone pressure. The assumption is that in realistic configurations, zone pressure will generally be close to ambient pressure. Unbalanced fan ventilation in a zone without relief area will result in runtime termination due to excessively high or low pressure.

izName

Optional name of interzone transfer; give after the word "IZXFER" if desired.

Units	Legal Range	Default	Required	Variability
	63 characters	*none*	No	constant

izNVType=*choice*

Choice determining interzone ventilation

NONE	No interzone ventilation
ONEWAY	Uncontrolled flow from izZn1 to izZn2 when izZn1 air temperature exceeds izZn2 air temperature (using ASHRAE high/low vent model).
TWOWAY	Uncontrolled flow in either direction (using ASHRAE high/low vent model).
AIRNETIZ	Single opening to another zone (using pressure balance AirNet model). Flow is driven by buoyancy.
AIRNETEXT	Single opening to ambient (using pressure balance AirNet model). Flow is driven by buoyancy and wind pressure.
AIRNETHORIZ	Horizontal (large) opening between two zones, used to represent e.g. stairwells. Flow is driven by buoyancy; simultaneous up and down flow is modeled.
AIRNETEXTFAN	Fan from exterior to zone (flow either direction).
AIRNETIZFAN	Fan between two zones (flow either direction).
AIRNETEXTFLOW	Specified flow from exterior to zone (either direction). Behaves identically to AIRNETEXTFAN except no electricity is consumed and no fan heat is added to the air stream.
AIRNETIZFLOW	Specified flow between two zones (either direction). Behaves identically to AIRNETIZFAN except no electricity is consumed and no fan heat is added to the air stream.
AIRNETHERV	Heat or energy recovery ventilator. Supply and exhaust air are exchanged with the exterior with heat and/or moisture exchange between the air streams. Flow may or may not be balanced.

Units	Legal Range	Default	Required	Variability
	choices above	NONE	No	constant

izZn1=*znName*

Name of primary zone. Flow rates > 0 are into the primary zone.

Units	Legal Range	Default	Required	Variability
	name of a ZONE		Yes	constant

izZn2=*znName*

Name of secondary zone.

Units	Legal Range	Default	Required	Variability
	name of a ZONE		required unless izNVType = AIRNETEXT, AIRNETEXTFAN, AIRNETEXTFLOW, or AIRNETHERV	constant

Give izHConst for a conductive transfer between zones. Give izNVType other than NONE and the following variables for a convective (air) transfer between the zones or between a zone and outdoors. Both may be given if desired. Not known to work properly as of July 2011

izHConst=*float*

Conductance between zones.

Units	Legal Range	Default	Required	Variability
Btu/°F	$x \geq 0$	0	No	hourly

izALo=*float*

Area of low or only vent (typically VentOff)

Units	Legal Range	Default	Required	Variability
ft^2	$x \geq 0$	0	No	hourly

izAHi=*float*

Additional vent area (high vent or VentOn). If used in AIRNET, izAHi > izALo typically but this is not required.

Units	Legal Range	Default	Required	Variability
ft^2	$x \geq 0$	izALo	No	hourly

izL1=*float*

Length or width of AIRNETHORIZ opening.

Units	Legal Range	Default	Required	Variability
ft	$x > 0$		if izNVType = AIRNETHORIZ	constant

izL2=*float*

Width or length of AIRNETHORIZ opening.

Units	Legal Range	Default	Required	Variability
ft	$x > 0$		if izNVType = AIRNETHORIZ	constant

izStairAngle=*float*

Stairway angle for AIRNETHORIZ opening. Use 90 for an open hole. Note that 0 prevents flow.

Units	Legal Range	Default	Required	Variability
degrees	$x > 0$	34	No	constant

izHD=*float*

Vent center-to-center height difference (for TWOWAY) or vent height above nominal 0 level (for AirNet types)

Units	Legal Range	Default	Required	Variability
ft		0	No	constant

izNVEff=*float*

Vent discharge coefficient.

Units	Legal Range	Default	Required	Variability
	$0 \leq x \leq 1$	0.8	No	constant

izfanVfDs=*float*

Fan design or rated flow at rated pressure. For AIRNETHERV, this is the net air flow into the zone, gross flow at the fan is derived using izEATR (see below).

Units	Legal Range	Default	Required	Variability
cfm	$x \geq 0$	0 (no fan)	If fan present	constant

izCpr=*float*

Wind pressure coefficient (for AIRNETEXT).

Units	Legal Range	Default	Required	Variability
	$x \geq 0$	0.	No	constant

izExp=*float*

Opening exponent (for AIRNETEXT).

Units	Legal Range	Default	Required	Variability
none	$x > 0$	0.5	No	constant

izVfMin=*float*

Minimum volume flow rate (VentOff mode).

Units	Legal Range	Default	Required	Variability
cfm	$x \geq 0$	izfanVfDs	No	subhourly

izVfMax=*float*

Maximum volume flow rate (VentOn mode)

Units	Legal Range	Default	Required	Variability
cfm	$x \geq 0$	izVfMin	No	subhourly

izASEF=*float*

Apparent sensible effectiveness for AIRNETHERV ventilator. ASEF is a commonly-reported HERV rating and is calculated as (supplyT - sourceT) / (returnT - sourceT). This formulation includes fan heat (in supplyT), hence the term "apparent".

Units	Legal Range	Default	Required	Variability
		0	No	subhourly

izEATR=*float*

Exhaust air transfer ratio for AIRNETHERV ventilator. NetFlow = (1 - EATR)*(grossFlow).

Units	Legal Range	Default	Required	Variability
cfm	$0 \leq \text{x} \leq 1$	0	No	subhourly

izLEF=*float*

Latent heat recovery effectiveness for AIRNETHERV ventilator. The default value (0) results in sensible-only heat recovery.

Units	Legal Range	Default	Required	Variability
	$0 \leq \text{x} \leq 1$	0	No	subhourly

izVfExhRat=*float*

Exhaust volume flow ratio for AIRNETHERV ventilator = (exhaust flow) / (supply flow). Any value other than 1 indicates unbalanced flow that effects the zone pressure.

Units	Legal Range	Default	Required	Variability
		1 (balanced)	No	subhourly

izfanPress=*float*

Design or rated fan pressure.

Units	Legal Range	Default	Required	Variability
inches H_2O	$x > 0$	.3	No	constant

Only one of izfanElecPwr, izfanEff, and izfanShaftBhp may be given: together with izfanVfDs and izfanPress, any one is sufficient for CSE to determine the others and to compute the fan heat contribution to the air stream.

izfanElecPwr=*float*

Fan input power per unit air flow (at design flow and pressure).

Units	Legal Range	Default	Required	Variability
W/cfm	$x > 0$	derived from izfanEff and izfanShaftBhp	If izfanEff and izfanShaftBhp not present	constant

izfanEff=*float*

Fan efficiency at design flow and pressure, as a fraction.

Units	Legal Range	Default	Required	Variability
	$0 \leq x \leq 1$	derived from *izfanShaftBhp* if given, else 0.08	No	constant

izfanShaftBhp=*float*

Fan shaft brake horsepower at design flow and pressure.

Units	Legal Range	Default	Required	Variability
bhp	$x > 0$	derived from *izfanEff*.	No	constant

izfanCurvePy=k_0**,** k_1**,** k_2**,** k_3**,** x_0

k_0 through k_3 are the coefficients of a cubic polynomial for the curve relating fan relative energy consumption to relative air flow above the minimum flow x_0. Up to five *floats* may be given, separated by commas. 0 is used for any omitted trailing values. The values are used as follows:

$$z = k_0 + k_1 \cdot (x - x_0)| + k_2 \cdot (x - x_0)|^2 + k_3 \cdot (x - x_0)|^3$$

where:

- x is the relative fan air flow (as fraction of izfan*VfDs*; $0 \leq x \leq 1$);
- x_0 is the minimum relative air flow (default 0);

- $(x - x_0)|$ is the "positive difference", i.e. $(x - x_0)$ if $x > x_0$; else 0;
- z is the relative energy consumption.

If z is not 1.0 for $x = 1.0$, a warning message is displayed and the coefficients are normalized by dividing by the polynomial's value for $x = 1.0$.

Units	Legal Range	Default	Required	Variability
		0, 1, 0, 0, 0 (linear)	No	constant

izFanMtr=*mtrName*

Name of meter, if any, to record energy used by supply fan. End use category used is specified by izFanEndUse (next).

Units	Legal Range	Default	Required	Variability
	name of a METER	*not recorded*	No	constant

izFanEndUse=*choice*

End use to which fan energy is recorded (in METER specified by izFanMtr). See METER for available end use choices.

Units	Legal Range	Default	Required	Variability
	end use choice	Fan	No	constant

endIZXFER

Optionally indicates the end of the interzone transfer definition.

Units	Legal Range	Default	Required	Variability
		N/A	No	constant

4.18 RSYS

RSYS constructs an object representing an air-based residential HVAC system.

rsName

Optional name of HVAC system; give after the word "RSYS" if desired.

Units	Legal Range	Default	Required	Variability
	63 characters	*none*	No	constant

rsType=*choice*

Type of system.

rsType	Description
ACFURNACE	Compressor-based cooling and fuel-fired heating. Primary heating input energy is accumulated to end use HTG of meter rsFuelMtr.
ACRESISTANCE	Compressor-based cooling and electric ("strip") heating. Primary heating input energy is accumulated to end use HTG of meter rsElecMtr.
ASHP	Air-source heat pump (compressor-based heating and cooling). Primary (compressor) heating input energy is accumulated to end use HTG of meter rsElecMtr. Auxiliary heating input energy is accumulated to end use HPHTG of meter rsElecMtr.
ASHPHYDRONIC	Air-to-water heat pump with hydronic distribution. Compressor performance is approximated using the air-to-air model with adjusted efficiencies.
AC	Compressor-based cooling; no heating.
FURNACE	Fuel-fired heating. Primary heating input energy is accumulated to end use HTG of meter rsFuelMtr.
RESISTANCE	Electric heating. Primary heating input energy is accumulated to end use HTG of meter rsElecMtr

Units	Legal Range	Default	Required	Variability
	one of above choices	ACFURNACE	No	constant

rsDesc=*string*

Text description of system, included as documentation in debugging reports such as those triggered by rsPerfMap=YES

Units	Legal Range	Default	Required	Variability
	string	*blank*	No	constant

rsModeCtrl=*choice*

Specifies systems heating/cooling availability during simulation.

OFF	System is off (neither heating nor cooling is available)
HEAT	System can heat (assuming rsType can heat)
COOL	System can cool (assuming rsType can cool)
AUTO	System can either heat or cool (assuming rsType compatibility). First request by any zone served by this RSYS determines mode for the current time step.

Units	Legal Range	Default	Required	Variability
	OFF, HEAT, COOL, AUTO	AUTO	No	hourly

rsPerfMap=*choice*

Generate performance map(s) for this RSYS. Comma-separated text is written to file PM_[rsName].csv.

This is a debugging capability that is not necessarily maintained.

Units	Legal Range	Default	Required	Variability
	NO, YES	NO	No	constant

rsFanTy=*choice*

Specifies fan (blower) position relative to cooling coil.

Units	Legal Range	Default	Required	Variability
	BLOWTHRU, DRAWTHRU	BLOWTHRU	No	constant

rsFanMotTy=*choice*

Specifies type of motor driving the fan (blower). This is used in the derivation of the coil-only cooling capacity for the RSYS.

PSC	Permanent split capacitor
BPM	Brushless permanent magnet (aka ECM)

Units	Legal Range	Default	Required	Variability
	PSC, BPM	PSC	No	constant

rsElecMtr=*mtrName*

Name of METER object, if any, by which system's electrical energy use is recorded (under appropriate end uses).

Units	Legal Range	Default	Required	Variability
	name of a METER	*not recorded*	No	constant

rsFuelMtr =*mtrName*

Name of METER object, if any, by which system's fuel energy use is recorded (under appropriate end uses).

Units	Legal Range	Default	Required	Variability
	name of a METER	*not recorded*	No	constant

rsAFUE=*float*

Heating Annual Fuel Utilization Efficiency (AFUE).

Units	Legal Range	Default	Required	Variability
	$0 < x \leq 1$	0.9 if furnace, 1.0 if resistance	No	constant

rsCapH=*float*

Heating capacity, used when rsType is ACFURNACE, ACRESISTANCE, FURNACE, or RESISTANCE.

Units	Legal Range	Default	Required	Variability
Btu/hr	*AUTOSIZE* or x $\geq$ 0	0	No	constant

rsTdDesH=*float*

Nominal heating temperature rise (across system, not zone) used during autosizing (when capacity is not yet known).

Units	Legal Range	Default	Required	Variability
°F	$x > 0$	30 if ASHP else 50	No	constant

rsFxCapH=*float*

Heating autosizing capacity factor. If AUTOSIZEd, rsCapH or rsCap47 are set to rsFxCapH × (peak design-day load). Peak design-day load is the heating capacity that holds zone temperature at the thermostat set point during the *last substep* of all hours of all design days.

Units	Legal Range	Default	Required	Variability
	$x > 0$	1.4	No	constant

rsFanPwrH=*float*

Heating fan power. Heating air flow is estimated based on a 50 °F temperature rise.

Units	Legal Range	Default	Required	Variability
W/cfm	$x \geq 0$	.365	No	constant

rsHSPF=*float*

For rsType=ASHP, Heating Seasonal Performance Factor (HSPF).

Units	Legal Range	Default	Required	Variability
Btu/Wh	$x > 0$		Yes if rsType=ASHP	constant

rsCap47=*float*

For rsType=ASHP, rated heating capacity at outdoor dry-bulb temperature = 47 °F. If both rsCap47 and rsCapC are autosized, both are set to the larger consistent value.

Units	Legal Range	Default	Required	Variability
Btu/Wh	*AUTOSIZE* or $x > 0$	Calculated from rsCapC	no	constant

rsCap35=*float*

For rsType=ASHP, rated heating capacity at outdoor dry-bulb temperature = 35 °F.

Units	Legal Range	Default	Required	Variability
Btu/Wh	$x > 0$	Calculated from rsCap47 and rsCap17	no	constant

rsCap17=*float*

For rsType=ASHP, rated heating capacity at outdoor dry-bulb temperature = 17 °F.

Units	Legal Range	Default	Required	Variability
Btu/Wh	$x > 0$	Calculated from rsCap47	no	constant

rsCOP47=*float*

For rsType=ASHP, rated heating coefficient of performance at outdoor dry-bulb temperature = 47 °F.

Units	Legal Range	Default	Required	Variability
	$x > 0$	Estimated from rsHSPF, rsCap47, and rsCap17	no	constant

rsCOP35=*float*

For rsType=ASHP, rated heating coefficient of performance at outdoor dry-bulb temperature = 35 °F.

Units	Legal Range	Default	Required	Variability
	$x > 0$	Calculated from rsCap35, rsCap47, rsCap17, rsCOP47, and rsCOP17	no	constant

rsCOP17=*float*

For rsType=ASHP, rated heating coefficient of performance at outdoor dry-bulb temperature = 17 °F.

Units	Legal Range	Default	Required	Variability
	$x > 0$	Calculated from rsHSPF, rsCap47, and rsCap17	no	constant

rsCapAuxH=*float*

For rsType=ASHP, auxiliary electric ("strip") heating capacity. If autosized, rsCapAuxH is set to the peak heating load in excess of heat pump capacity evaluated at the heating design temperature (Top.heatDsTDbO).

Units	Legal Range	Default	Required	Variability
Btu/hr	*AUTOSIZE* or $x \geq 0$	0	no	constant

rsFxCapAuxH=*float*

Auxiliary heating autosizing capacity factor. If AUTOSIZEd, rsCapAuxH is set to rsFxCapAuxH × (peak design-day load). Peak design-day load is the heating capacity that holds zone temperature at the thermostat set point during the *last substep* of all hours of all design days.

Units	Legal Range	Default	Required	Variability
	$x > 0$	1	No	constant

rsCOPAuxH=*float*

For rsType=ASHP, auxiliary electric ("strip") heating coefficient of performance. Energy use for auxiliary heat is accumulated to end use HPHTG of meter rsElecMtr (that is, auxiliary heat is assumed to be electric).

Units	Legal Range	Default	Required	Variability
	$x \geq 0$	1.0	no	constant

rsSEER=*float*

Cooling rated Seasonal Energy Efficiency Ratio (SEER).

Units	Legal Range	Default	Required	Variability
Btu/Wh	$x > 0$		Yes	constant

rsEER=*float*

Cooling Energy Efficiency Ratio (EER) at standard AHRI rating conditions (outdoor drybulb of 95 °F and entering air at 80 °F drybulb and 67 °F wetbulb).

Units	Legal Range	Default	Required	Variability
Btu/Wh	$x > 0$	Estimated from SEER	no	constant

rsCapC=*float*

Cooling capacity at standard AHRI rating conditions. If rsType=ASHP and both rsCapC and rsCap47 are autosized, both are set to the larger consistent value.

Units	Legal Range	Default	Required	Variability
Btu/hr	*AUTOSIZE* or $x \leq 0$ (x > 0 coverted to < 0)		Yes if rsType includes cooling	constant

rsTdDesC=*float*

Nominal cooling temperature fall (across system, not zone) used during autosizing (when capacity is not yet known).

Units	Legal Range	Default	Required	Variability
°F	$x < 0$	-25	No	constant

rsFxCapC=*float*

Cooling autosizing capacity factor. rsCapC is set to rsFxCapC × (peak design-day load). Peak design-day load is the cooling capacity that holds zone temperature at the thermostat set point during the *last substep* of all hours of all design days.

Units	Legal Range	Default	Required	Variability
	$x > 0$	1.4	No	constant

rsFChg=*float*

Refrigerant charge adjustment factor.

Units	Legal Range	Default	Required	Variability
	$x > 0$	1	no	constant

rsFSize=*float*

Compressor sizing factor.

Units	Legal Range	Default	Required	Variability
	$x > 0$	1	no	constant

rsVFPerTon=*float*

Standard air volumetric flow rate per nominal ton of cooling capacity.

Units	Legal Range	Default	Required	Variability
cfm/ton	150 ≤ x ≤ 500	350	no	constant

rsFanPwrC=*float*

Cooling fan power.

Units	Legal Range	Default	Required	Variability
W/cfm	$x \geq 0$	.365	No	constant

rsASHPLockOutT=*float*

Source air dry-bulb temperature below which the air source heat pump compressor does not operate.

Units	Legal Range	Default	Required	Variability
°F		(no lockout)	No	hourly

rsCdH=*float*

Heating cyclic degradation coefficient, valid only for compressor-based heating (heat pumps).

Units	Legal Range	Default	**Require d	Variabili t y
	0 ≤ x ≤ 0.5	ASHPHYDRONIC: 0.25 ASHP: derived from rsHSPF	No	hourly

rsCdC=*float*

Cooling cyclic degradation coefficient, valid for configurations having compressor-based cooling.

Units	Legal Range	Default	Required	Variability
	$0 \leq x \leq 0.5$	0	No	hourly

rsDSEH=*float*

Heating distribution system efficiency. If given, (1-rsDSEH) of RSYS heating output is discarded. Cannot be combined with more detailed DUCTSEG model.

Units	Legal Range	Default	Required	Variability
	$0 < x < 1$	(use DUCTSEG model)	No	hourly

rsDSEC=*float*

Cooling distribution system efficiency. If given, (1-rsDSEC) of RSYS cooling output is discarded. Cannot be combined with more detailed DUCTSEG model.

Units	Legal Range	Default	Required	Variability
	$0 < x < 1$	(use DUCTSEG model)	No	hourly

rsOAVType=*choice*

Type of central fan integrated (CFI) outside air ventilation (OAV) included in this RSYS. OAV systems use the central system fan to circulate outdoor air (e.g. for night ventilation).

OAV cannot operate simultaneously with whole building ventilation (operable windows, whole house fans, etc.). Availability of ventilation modes is controlled on an hourly basis via Top ventAvail.

NONE	No CFI ventilation capabilities
FIXED	Fixed-flow CFI (aka SmartVent). The specified rsOAVVfDs is used whenever the RSYS operates in OAV mode.
VARIABLE	Variable-flow CFI (aka NightBreeze). Flow rate is determined at midnight based on prior day's average dry-bulb temperature according to a control algorithm defined by the NightBreeze vendor.

Units	Legal Range	Default	Required	Variability
	NONE, FIXED, VARIABLE	NONE	No	constant

rsOAVVfDs=*float*

Design air volume flow rate when RSYS is operating in OAV mode.

Units	Legal Range	Default	Required	Variability
cfm	≥ 0		if rsOAVType $\neq$ NONE	constant

rsOAVVfMinF=*float*

Minimum air volume flow rate fraction when RSYS is operating in OAV mode. When rsOAVType=VARIABLE, air flow rate is constrained to rsOAVVfMinF * rsOAVVfDs or greater.

Units	Legal Range	Default	Required	Variability
	$0 \leq x \leq 1$	0.2	No	constant

rsOAVFanPwr=*float*

RSYS OAV-mode fan power.

**Unit s	**Legal Range**	**Default**	**Required**	**Variabili t y**
W/cfm	$0 < x \leq 5$	per rsOAVTYPE FIXED: rsFanPwrC VARIABLE: NightBreeze vendor curve based on rsOAVvfDs	No	constant

rsOAVTDbInlet=*float*

OAV inlet (source) air temperature. Supply air temperature at the zone is generally higher due to fan heat. Duct losses, if any, also alter the supply air temperature.

Units	**Legal Range**	**Default**	**Required**	**Variabili ty
°F		Dry-bulb temperature from weather file	No	hourly

rsOAVTdiff=*float*

OAV temperature differential. When operating in OAV mode, the zone set point temperature is max(znTD, inletT+rsOAVTdiff). Small values can result in inadvertent zone heating, due to fan heat.

Units	Legal Range	Default	Required	Variability
°F	> 0	5 °F	No	hourly

rsOAVReliefZn=*znName*

Name of zone to which relief air is directed during RSYS OAV operation, typically an attic zone. Relief air flow is included in the target zone's pressure and thermal balance.

Units	Legal Range	Default	Required	Variability
	name of ZONE		if rsOAVType $\neq$ NONE	constant

rsParElec=*float*

Parasitic electrical power. rsParElec is unconditionally accumulated to rsElecMtr (if specified) and has no other effect.

Units	Legal Range	Default	Required	Variability
W		0	No	hourly

rsParFuel=*float*

Parasitic fuel use. rsParFuel is unconditionally accumulated to rsFuelMtr (if specified) and has no other effect.

Units	Legal Range	Default	Required	Variability
Btuh		0	No	hourly

rsRhIn=*float*

Entering air relative humidity (for model testing).

Units	Legal Range	Default	Required	Variability
W/cfm	$0 \leq x \leq 1$	Derived from entering air state	No	constant

rsTdbOut=*float*

Air dry-bulb temperature at the outdoor portion of this system.

Units	Legal Range	Default	Required	Variability
°F		From weather file	No	hourly

endRSYS

Optionally indicates the end of the RSYS definition.

Units	Legal Range	Default	Required	Variability
		N/A	No	constant

4.19 DUCTSEG

DUCTSEG defines a duct segment. Each RSYS has at most one return duct segment and at most one supply duct segment. That is, DUCTSEG input may be completely omitted to eliminate duct losses.

dsName

Optional name of duct segment; give after the word "DUCTSEG" if desired.

Units	Legal Range	Default	Required	Variability
	63 characters	*none*	No	constant

dsTy=*choice*

Duct segment type.

Units	Legal Range	Default	Required	Variability
	SUPPLY, RETURN		Yes	constant

The surface area of a DUCTSEG depends on its shape. 0 surface area is legal (leakage only). DUCTSEG shape is modeled either as flat or round –

- dsExArea specified: Flat. Interior and exterior areas are assumed to be equal (duct surfaces are flat and corner effects are neglected).
- dsExArea *not* specified: Round. Any two of dsInArea, dsDiameter, and dsLength must be given. Insulation thickness is derived from dsInsulR and dsInsulMat and this thickness is used to calculate the exterior surface area. Overall inside-to-outside conductance is also calculated including suitable adjustment for curvature.

dsExArea=*float*

Duct segment surface area at outside face of insulation for flat duct shape, see above.

Units	Legal Range	Default	Required	Variability
ft^2	$x \geq 0$		No	constant

dsInArea=*float*

Duct segment inside surface area (at duct wall, duct wall thickness assumed negligible) for round shaped duct.

Units	Legal Range	Default	Required	Variability
ft^2	$x \geq 0$	Derived from dsDiameter and dsLength	(see above re duct shape)	constant

dsDiameter=*float*

Duct segment round duct diameter (duct wall thickness assumed negligible)

Units	Legal Range	Default	Required	Variability
ft	$x \geq 0$	Derived from dsInArea and dsLength	(see above re duct	constant shape)

dsLength=*float*

Duct segment length.

Units	Legal Range	Default	Required	Variability
ft	$x \geq 0$	Derived from dsInArea and dsDiameter	(see above re duct shape)	constant

dsExCnd=*choice*

Conditions surrounding duct segment.

Units	Legal Range	Default	Required	Variability
	ADIABATIC, AMBIENT, SPECIFIEDT, ADJZN	ADJZN	No	constant

dsAdjZn=*znName*

Name of zone surrounding duct segment; used only when dsExCon is ADJZN. Can be the same as a zone served by the RSYS owning the duct segment.

Units	Legal Range	Default	Required	Variability
	name of a *ZONE*	*none*	Required when *dsExCon* = ADJZN	constant

dsEpsLW=*float*

Exposed (i.e. insulation) outside surface exterior long wave (thermal) emittance.

Units	Legal Range	Default	Required	Variability
(none)	$0 \leq x \leq 1$	0.9	No	constant

dsExT=*float*

Air dry-bulb temperature surrounding duct segment.

Units	Legal Range	Default	Required	Variability
°F	*unrestricted*	*none*	Required if *sfExCnd* = SPECIFIEDT	hourly

Duct insulation is modeled as a pure conductance (no mass).

dsInsulR=*float*

Insulation thermal resistance *not including*surface conductances. dsInsulR and dsInsulMat are used to calculate insulation thickness (see below).

Units	Legal Range	Default	Required	Variability
ft^2-F-hr / Btu	$x \geq 0$	0	No	constant

dsInsulMat=*matName*

Name of insulation MATERIAL. The conductivity of this material at 70 °F is combined with dsInsulR to derive the duct insulation thickness. If omitted, a typical fiberglass material is assumed having conductivity of 0.025 Btu/hr-ft^2-F at 70 °F and a conductivity coefficient of .00418 1/F (see MATERIAL). In addition, insulation conductivity is adjusted during the simulation in response its average temperature.

Units	Legal Range	Default	Required	Variability
	name of a *MATERIAL*	fiberglass	No	constant

dsLeakF=*float*

Duct leakage. Return duct leakage is modeled as if it all occurs at the segment inlet. Supply duct leakage is modeled as if it all occurs at the outlet.

Units	Legal Range	Default	Required	Variability
	$0 < x \leq 1$		No	constant

dsExH=*float*

Outside (exposed) surface convection coefficient.

Units	Legal Range	Default	Required	Variability
Btuh/ft^2-^{o}F	$x > 0$	.54	No	subhourly

endDuctSeg

Optionally indicates the end of the DUCTSEG definition.

Units	Legal Range	Default	Required	Variability
		N/A	No	

4.20 DHWDAYUSE

Defines an object that represents domestic hot water use for a single day. A DHWDAYUSE contains a collection of DHWUSE objects that specify the time, volume, and duration of individual draws. DHWDAYUSEs are referenced by DHWSYS wsDayUse. Unreferenced DHWDAYUSEs are allowed.

DHWDAYUSEs and their child DHWUSEs are used to construct minute-by-minute hot water use schedules in addition to aggregated hourly schedules. The minute-by-minute schedules are used for modeling resistance and heat pump storage water heaters, see DHWHEATER whType=SmallStorage whHeatSrc=ResistanceX or whHeatSrc=ASHPX.

The following illustrates some features of DHWDAYUSE / DHWUSE –

```
DHWDAYUSE "Sample"
   // 6 AM: 7 min shower, 2 gpm @ 105 F
   DHWUSE whStart=6.0 wuDuration=7 wuFlow=2 wuTemp=105 wuEndUse=Shower wuEventID=1

   // 7 AM: 1 min faucet draw, 100% hot
   DHWUSE whStart=7.0 wuDuration=1 wuFlow=1 wuHotF=1 whEndUse=Faucet wuEventID=2

   // 12:30 PM: dishwasher start, several draws over 70 mins; note common wuEventID
   DHWUSE whStart=12.5 wuDuration=2 wuFlow=2 wuHotF=1 whEndUse=DWashr wuEventID=3
   DHWUSE whStart=12.8 wuDuration=1.5 wuFlow=2 wuHotF=1 whEndUse=DWashr wuEventID=3
   DHWUSE whStart=13.6 wuDuration=3 wuFlow=2 wuHotF=1 whEndUse=DWashr wuEventID=3

   // 7 PM every 2nd day: clothes washer runs
   //   even days: 0 gpm (no draw)
   //   odd days: 3 gpm, 22% hot
  DHWUSE whStart=19 wuDuration=30 wuFlow = ($dayOfYear%2)*3 whEndUse=CWashr whHotF=.22 wuEventID=4

   // 11:54 PM: 20 min bath, 1.5 gpm, 80% hot water
   // Duration spans midnight: draw is wrapped to beginning of *current* day
```

```
   //   In this case a 12 M - 12:14 AM draw is modeled -- before (!) the bath start.
   DHWUSE whStart 23.9 wuDuration=20 wuFlow=1.5  wuHotF=.8 whEndUse=Bath wuEventID=99
endDHWDAYUSE

DHWSYS "DHWSYS1"
  ...
  wsDayUse = "Sample"
  ...
```

During the simulation, DHWUSEs are evaluated each hour. Many DHWUSE values have hourly variability and this allows complicated schemes to be constructed very flexibly. For example:

```
DHWDAYUSE "HourlyFaucet"
   // Every hour on the half hour: 5 minute, 2 gpm draw
   //   Same as 24 DHWUSEs, one for each hour
   DHWUSE wuStart=$hour+.5 wuDuration=5 wuFlow=2 wuEndUse=Faucet
endDAYUSE
```

Some DHWUSE configurations involve mixing to specified wuTemp. Hot and cold water arriving at the point of use is assumed to be at DHWSYS wsUseTemp and wsMainsTemp respectively. It is possible to set up situations where wuTemp cannot be achieved (wuTemp > wsUseTemp, for example). Runtime error messages are produced when impossible conditions are detected.

When more than one DHWSYS references the same DHWDAYUSE, DHWUSEs are allocated to DHWSYSs in wuEventID rotation. This procedure divides the water heating load approximately equally while retaining the peak demand of individual events. When detailed information is available about which loads are served by specific systems, separate DHWDAYUSEs should be given.

dhwDayUseName

Object name, given after "DHWDAYUSE". Required for referencing from DHWSYS.

Units	Legal Range	Default	Required	Variability
	63 characters	*none*	Yes	constant

wduMult=*float*

Scale factor applied to all draws in this DHWDAYUSE.

Units	Legal Range	Default	Required	Variability
	≥ 0	1	No	constant

endDHWDAYUSE

Indicates the end of the DHWDAYUSE definition. endDHWDAYUSE should follow all child DHWUSEs. Alternatively, the end of the meter definition can be indicated by the declaration of another object or by END.

Units	Legal Range	Default	Required	Variability
		N/A	No	constant

4.21 DHWUSE

Defines a single hot water draw as part of a DHWDAYUSE. See discussion and examples under DHW-DAYUSE. As noted there, most DHWUSE values have hourly variability, allowing flexible representation

wuName

Optional name; give after the word "DHWUSE" if desired.

Units	Legal Range	Default	Required	Variability
	63 characters	*none*	No	constant

wuStart=*float*

The starting time of the hot water draw.

Units	Legal Range	Default	Required	Variability
hr	$0 \leq x \leq 24$	–	Yes	constant

wuDuration=*float*

Draw duration. wuDuration = 0 is equivalent to omitting the DHWUSE. Durations that extend beyond midnight are included in the current day.

Units	Legal Range	Default	Required	Variability
min	$0 \leq x \leq 1440$	0	N	hourly

wuFlow=*float*

Draw flow rate at the point of use (in other words, the mixed-water flow rate). wuFlow = 0 is equivalent to omitting the DHWUSE. There is no enforced upper limit on wuFlow, however, unrealistically large values will cause runtime errors.

Units	Legal Range	Default	Required	Variability
gpm	$0 \leq x$	0	N	hourly

wuHotF=*float*

Fraction of draw that is hot water. Cannot be specified with wuTemp or wuHeatRecEF.

Units	Legal Range	Default	Required	Variability
–	$0 \leq x \leq 1$	1	N	hourly

wuTemp=*float*

Mixed-water use temperature at the fixture. Cannot be specified when wuHotF is given.

Units	Legal Range	Default	Required	Variability
°F	$0 \leq x$	0	when wuHeatRecEF is given	hourly

wuHeatRecEF=*float*

Heat recovery effectiveness. If non-0, wuHeatRecEF allows modeling of heat recovery devices such as drain water heat exchangers. If given, wuTemp must also be specified.

Units	Legal Range	Default	Required	Variability
–	$0 \leq x \leq 0.9$	0	when wu	hourly

wuHWEndUse=*choice*

Hot-water end use: one of Shower, Bath, CWashr, DWashr, or Faucet. whHWEndUse has two functions –

- Allocation of hot water use among multiple DHWSYSs (if more than one DHWSYS references a given DHWDAYUSE).
- DHWMETER end-use accounting (via DHWSYS).

Units	Legal Range	Default	Required	Variability
–	One of above choices	(use allocated to Unknown)	N	constant

wuEventID=*integer*

User-defined identifier that associates multiple DHWUSEs with a single event or activity. For example, a dishwasher uses water at several discrete times during a 90 minute cycle and all DHWUSEs would be assigned the same wuEventID. All DHWUSEs having the same wuEventID should have the same wuHWEndUse.

Units	Legal Range	Default	Required	Variability
–	$0 \leq x$	0	N	constant

endDHWUSE

Optionally indicates the end of the DHWUSE definition.

Units	Legal Range	Default	Required	Variability
		N/A	No	

4.22 DHWSYS

DHWSYS constructs an object representing a domestic hot water system consisting of one or more hot water heaters, storage tanks, loops, and pumps (DHWHEATER, DHWTANK, DHWLOOP, and DHWPUMP, see below) and a distribution system characterized by loss parameters. This model is based on Appendix B of the 2016 Residential ACM Reference Manual. This version is preliminary, revisions are expected.

The parent-child structure of DHWSYS components is determined by input order. For example, DHWHEATERs belong to DHWSYS that precedes them in the input file. The following hierarchy shows the relationship among components. Note that any of the commands can be repeated any number of times.

- DHWSYS
  - DHWHEATER
  - DHWTANK
  - DHWPUMP
  - DHWLOOP
    - DHWLOOPPUMP

* DHWLOOPSEG
  - DHWLOOPBRANCH

No actual controls are modeled. For example, if several DHWHEATERs are included in a DHWSYS, an equal fraction of the required hot water is assumed to be produced by each heater, even if they are different types or sizes. Thus a DHWSYS is in some ways just a collection of components, rather than a physically realistic system.

dhwsysName

Optional name of system; give after the word "DHWSYS" if desired.

Units	Legal Range	Default	Required	Variability
	63 characters	*none*	No	constant

wsCentralDHWSYS=*dhwsysName*

Name of the central DHWSYS that serves this DHWSYS, allowing representation of multiple units having distinct distribution configurations and/or water use patterns but served by a central DHWSYS. The child DHWSYS(s) may not include DHWHEATERs – they are "loads only" systems. wsCentralDHWSYS and wsLoadShareDHWSYS cannot both be given.

Units	Legal Range	Default	Required	Variability
	name of a DHWSYS	DHWSYS is standalone	No	constant

wsLoadShareDHWSYS=*dhwsysName*

Name of a DHWSYS that serves the same loads as this DHWSYS, allowing representation of multiple water heating systems within a unit. If given, wsUse and wsDayUse are not allowed, hot water requirements are derived from the referenced DHWSYS. wsCentralDHWSYS and wsLoadShareDHWSYS cannot both be given.

Units	Legal Range	Default	Required	Variability
	name of a DHWSYS	No shared loads	No	constant

wsMult=*integer*

Number of identical systems of this type (including all child objects). Any value > 1 is equivalent to repeated entry of the same DHWSYSs.

Units	Legal Range	Default	Required	Variability
	> 0	1	No	constant

wsTInlet=*float*

Specifies cold (mains) water temperature supplied to DHWHEATERs in this DHWSYS.

Units	Legal Range	Default	Required	Variability
°F	> 32 °F	Mains temp from weather file	No	hourly

wsUse=*float*

Hourly hot water use (at the point of use). See further info under wsDayUse.

Units	Legal Range	Default	Required	Variability
gal	≥ 0	0	No	hourly

wsDayUse=*dhwdayuseName*

Name of DHWDAYUSE object that specifies a detailed schedule of hot water use (at point of use).

The total water use modeled by CSE is the sum of amounts given by wsUse and the DWHDAYUSE schedule. DHWDAYUSE draws are resolved to minute-by-minute bins compatible with the HPWH model and wsUse/60 is added to each minute bin. Conversely, the hour total of the DHWDAYUSE amounts is included in the draw applied to non-HPWH DHWHEATERs.

wsDayUse variability is daily, so it is possible to select different schedules as a function of day type (or any other condition), as follows –

```
DHWSYS "DHW1"
  ...
  wsDayUse = choose( $isWeHol, "DUSEWeekday", "DUSEWeHol")
  ...
```

Note that while DHWDAYUSE selection is updated daily, the DHWUSE values within the DHWDAYUSE can be altered hourly, providing additional scheduling flexibility.

Units	Legal Range	Default	Required	Variability
gal	≥ 0	(no scheduled draws)	No	daily

wsTUse=*float*

Hot water delivery temperature (at the point of use). Note that draws defined via DHWDAYUSE / DHWUSE can specify mixing to a lower temperature.

Units	Legal Range	Default	Required	Variability
°F	> 32 °F	120	No	hourly

wsTSetPoint=*float*

Specifies hot water setpoint temperature

Units	Legal Range	Default	Required	Variability
°F	> 32 °F	wsTUse	No	hourly

wsParElec=*float*

Specifies electrical parasitic power to represent recirculation pumps or other system-level electrical devices. Calculated energy use is accumulated to the METER specified by wsElecMtr (end use DHW). No other effect, such as heat gain to surroundings, is modeled.

Units	Legal Range	Default	Required	Variability
W	> 0	0	No	hourly

wsSDLM=*float*

Specifies the standard distribution loss multiplier. See App B Eqn 4. To duplicate CEC 2016 methods, this value should be set according to the value derived with App B Eqn 5.

Units	Legal Range	Default	Required	Variability
	> 0	1	No	constant

wsDSM=*float*

Distribution system multiplier. See RACM App B Eqn 4. To duplicate CEC 2016 methods, wsDSM should be set to the appropriate value from App B Table B-2. Note the NCF (non-compliance factor) included in App B Eqn 4 is *not* a CSE input and thus must be applied externally to wsDSM.

Units	Legal Range	Default	Required	Variability
	> 0	1	No	constant

wsWF=*float*

Waste factor. See RACM App B Eqn 1. wsWF is applied to hot water draws. The default value (1) reflects the inclusion of waste in draw amounts. App B specifies wsWF=0.9 when the system has a within-unit pumped loop that reduces waste due to immediate availability of hot water at fixtures.

Units	Legal Range	Default	Required	Variability
	> 0	1	No	hourly

wsSSF=*float*

Specifies the solar savings fraction.

Units	Legal Range	Default	Required	Variability
	≥ 0	0	No	hourly

wsElecMtr=*mtrName*

Name of METER object, if any, to which DHWSYS electrical energy use is recorded (under end use DHW). In addition, wsElecMtr provides the default whElectMtr selection for all DHWHEATERs and DHWPUMPs in this DHWSYS.

Units	Legal Range	Default	Required	Variability
	name of a METER	*not recorded*	No	constant

wsFuelMtr =*mtrName*

Name of METER object, if any, to which DHWSYS fuel energy use is recorded (under end use DHW). DHWSYS fuel use is usually (always?) 0, so the primary use of this input is to specify the default whFuelMtr choice for DHWHEATERs in this DHWSYS.

Units	Legal Range	Default	Required	Variability
	name of a METER	*not recorded*	No	constant

wsWHhwMtr=*dhwmtrName*

Name of DHWMETER object, if any, to which hot water quantities (at water heater) are recorded by hot water end use.

Units	Legal Range	Default	Required	Variability
	name of a METER	*not recorded*	No	constant

wsFXhwMtr =*dhwmtrName*

Name of DHWMETER object, if any, to which mixed hot water use (at fixture) quantities are recorded by hot water end use. DHWDAYUSE and wsUse input can be verified using DHWMETER results.

Units	Legal Range	Default	Required	Variability
	name of a METER	*not recorded*	No	constant

wsCalcMode=*choice*

PRERUN	Calculate hot water heating load; at end of run, derive whLDEF for all child DHWHEATERs for which that value is required and defaulted. This procedure emulates methods used in the T24DHW.DLL implementation of CEC DHW procedures.
SIMULATE	Perform full modeling calculations

Units	Legal Range	Default	Required	Variability
	Codes listed above	SIMULATE	No	

endDHWSys

Optionally indicates the end of the DHWSYS definition.

Units	Legal Range	Default	Required	Variability
		N/A	No	

4.23 DHWHEATER

DHWHEATER constructs an object representing a domestic hot water heater (or several if identical).

whName

Optional name of water heater; give after the word "DHWHEATER" if desired.

Units	Legal Range	Default	Required	Variability
	63 characters	*none*	No	constant

whMult=*integer*

Number of identical water heaters of this type. Any value > 1 is equivalent to repeated entry of the same

DHWHEATER.

Units	Legal Range	Default	Required	Variability
	> 0	1	No	constant

whType=*choice*

Type of water heater. This categorization is based on CEC and federal rating standards that change from time to time.

SMALLSTORAGE	A storage water heater having an energy factor (EF) rating. Generally, a gas-fired storage water heater with input of 75,000 Btuh or less, an oil-fired storage water heater with input of 105,000 Btuh or less, an electric storage water heater with input of 12 kW or less, or a heat pump water heater rated at 24 amps or less.
LARGESTORAGE	Any storage water heater that is not SMALLSTORAGE.
SMALLINSTANTANEOUS	A water heater that has an input rating of at least 4,000 Btuh per gallon of stored water. Small instantaneous water heaters include: gas instantaneous water heaters with an input of 200,000 Btu per hour or less, oil instantaneous water heaters with an input of 210,000 Btu per hour or less, and electric instantaneous water heaters with an input of 12 kW or less.
LARGEINSTANTANEOUS	An instantaneous water heater that does not conform to the definition of SMALLINSTANTANEOUS, an indirect fuel-fired water heater, or a hot water supply boiler.

Units	Legal Range	Default	Required	Variability
	Codes listed above	SMALLSTORAGE	No	constant

whHeatSrc=*choice*

Heat source for water heater. CSE implements uses efficiency-based models for all whTypes (as documented in RACM, App. B). In addition, the detailed Ecotope HPWH model is available for electric (air source heat pump and resistance) SMALLSTORAGE water heaters.

RESISTANCE	Electric resistance heating element Deprecated for whType=SMALLSTORAGE (use RESISTANCEX)
RESISTANCEX	Electric resistance heating element, detailed HPWH model
ASHP	Air source heat pump, EF model Deprecated for whType=SMALLSTORAGE (use ASHPX)
ASHPX	Air source heat pump, detailed HPWH model
FUEL	Fuel-fired burner

Units	Legal Range	Default	Required	Variability
	Codes listed above	FUEL	No	constant

whVol=*float*

Storage tank volume. Must be omitted or 0 for instantaneous whTypes. Used in the detailed HPWH model when whHeatSrc=RESISTANCEX or whHeatSrc=ASHPX with whASHPType=GENERIC (other whASHPTypes implicitly determine tank volume). For all other configurations, whVol is documentation-only.

Units	Legal Range	Default	Required	Variability
gal	$\geq$ 0.1 (caution: small values may cause runtime errors)	50 (when not required)	When used by detailed HPWH model, see above	constant

whEF=*float*

Rated energy factor that specifies DHWHEATER efficiency under test conditions. Used by CSE to derive annual water heating efficiency and/or other characteristics as described below. Calculation methods are documented in RACM, Appendix B.

Configuration	whEF default	Use
whType=SMALLSTORAGE whHeatSrc=RESISTANCE or FUEL	0.82	Derivation of whLDEF
whType=SMALLSTORAGE whHeatSrc=ASHP	0.82	Derivation of whLDEF note inappropriate default (deprecated, use ASHPX)
whType=SMALLSTORAGE whHeatSrc=ASHPX whASHPType=GENERIC	(req'd)	Tank losses Overall efficiency
whType=SMALLSTORAGE whHeatSrc=RESISTANCEX	(req'd)	Tank losses Note: maximum whEF=0.98.
whType=SMALLINSTANTANEOUS whHeatSrc=RESISTANCE or FUEL	0.82	Annual efficiency = whEF*0.92
Any other	(unused)	

Units	Legal Range	**Defaul t	**Require d	Variabilit y
	> 0 *Caution: maximum not checked. Unrealistic values will cause runtime errors and/or invalid results*	*See above*	*See above*	constant

whLDEF=*float*

Load-dependent energy factor for DHWHEATERs with whType=SMALLSTORAGE and whHeat-Src=FUEL or whHeatSrc=RESISTANCE. If not given, whLDEF is derived using a preliminary simulation activated via DHWSYS wsCalcMode=PRERUN. See RACM Appendix B.

Units	Legal Range	Default	Required	Variability
	> 0	Calculated via DHWSYS PreRun mechanism	When whType = SMALLSTORAGE and PreRun not used	constant

whZone=*znName*

Name of zone where water heater is located, used only in detailed HPWH models (whHeatSrc=ASHPX or whHeatSrc=RESISTANCEX), otherwise no effect. Zone conditions are used for tank heat loss calculations. Heat exchanged with the DHWHEATER are included in the zone heat balance. whZone also provides the default for whASHPSrcZn (see below). whZone and whTEx cannot both be specified.

Units	Legal Range	Default	Required	**Variability
	name of a ZONE	Not in a zone (heat losses discarded)	No	constant

whTEx=*float*

Water heater surround temperature, used only in detailed HPWH models (whHeatSrc=ASHPX or whHeatSrc=RESISTANCEX), otherwise no effect. whZone and whTEx cannot both be specified.

Units	Legal Range	Default	Required	Variability
°F	≥ 0	whZone air temperature if specified, else 70 °F	No	hourly

whASHPType=*choice*

Air source heat pump type, valid only if whHeatSrc=ASHPX. These choices are supported by the detailed HPWH model. Except for Generic, all heater characteristics are set by HPWH based on whASHPType.

Choice	Specified type
Generic	General generic (parameterized by wh_EF and wh_vol)
AOSmithPHPT60	60 gallon Voltex
AOSmithPHPT80	80 gallon Voltex
AOSmithHPTU50	50 gallon AOSmith HPTU
AOSmithHPTU66	66 gallon AOSmith HPTU
AOSmithHPTU80	80 gallon AOSmith HPTU
Sanden40	Sanden 40 gallon CO2 external heat pump
Sanden80	Sanden 80 gallon CO2 external heat pump
GE2012	2012 era GeoSpring
GE2014	2014 50 gal GE run in the efficiency mode
GE2014StdMode	2014 50 gal GE run in standard mode
GE2014StdMode80	2014 80 gal GE run in standard mode
RheemHB50	newish Rheem (2014 model?)
Stiebel220E	Stiebel Eltron (2014 model?)
GenericTier1	Generic Tier 1
GenericTier2	Generic Tier 2
GenericTier3	Generic Tier 3

Choice	Specified type
UEF2Generic	Experimental UEF=2
BasicIntegrated	Typical integrated HPWH
ResTank	Resistance heater (no compressor). Superceded by whHeatSrc=RESITANCEX
ResTankNoUA	Resistance heater (no compressor) with no tank losses. Superseded by whHeatSrc=RESISTANCEX.
AOSmithHPTU80DR	80 gallon AOSmith HPTU with fixed backup setpoint (experimental for demand response testing)
AOSmithSHPT50	50 gal AOSmith SHPT
AOSmithSHPT66	66 gal AOSmith SHPT
AOSmithSHPT80	80 gal AOSmith SHPT

Units	**Legal Range**	**Default**	**Required**	**Variability**
	Codes listed above	–	When whHeatSrc=ASHPX	constant

whASHPSrcZn=*znName*

Name of zone that serves as heat pump heat source used when whHeatSrc=ASHPX. Used for tank heat loss calculations and default for whASHPSrcZn. Heat exchanges are included in zone heat balance. whASHPSrcZn and whASHPSrcT cannot both be specified.

Units	**Legal Range**	**Default**	**Required**	**Variability**
	name of a ZONE	Same as whZone if whASHPSrcT not specified. If no zone is specified by input or default, heat extracted by ASHP has no effect.	No	constant

whASHPSrcT=*float*

Heat pump source air temperature used when whHeatSrc=ASHPX. Heat removed from this source is added to the heated water but has no other effect. whASHPSrcZn and whASHPSrcT cannot both be specified.

Units	**Legal Range**	**Default**	**Required**	**Variability**
°F	≥ 0	whASHPZn air temperature if specified, else 70 °F	No	hourly

whASHPResUse=*float*

Specifies activation temperature difference for resistance heating, used only when whHeatSrc=ASHPX and whASHPType=GENERIC. Refer to HPWH engineering documentation for model details.

Units	**Legal Range**	**Default**	**Required**	**Variability**
°C	≥ 0	7.22	N	constant

whResHtPwr=*float*

Specifies resistance upper element power, used only with whHeatSrc=RESISTANCEX.

Units	Legal Range	Default	Required	Variability
W	≥ 0	4500	N	constant

whResHtPwr2=*float*

Specifies resistance lower element power, used only with whHeatSrc=RESISTANCEX.

Units	Legal Range	Default	Required	Variability
W	≥ 0	whResHtPwr	N	constant

whHPAF=*float*

Heat pump adjustment factor, applied to whLDEF when modeling whType=SMALLSTORAGE and whHeatSrc=ASHP. This value should be derived according to RACM App B Table B-6. Deprecated: the detailed HPWH model (whHeatSrc=ASHPX) is recommended for air source heat pumps.

Units	Legal Range	Default	Required	Variability
	> 0	1	When whType=SMALLSTORAGE and whHeatSrc=ASHP co	nstant

whEff=*float*

Water heating efficiency, used in modeling whType=LARGESTORAGE and whType=LARGEINSTANTANEOUS.

Units	Legal Range	Default	Required	Variability
	$0 <$ whEff ≤ 1	.82	No	constant

whSBL=*float*

Standby loss, used in modeling whType=LARGESTORAGE.

Units	Legal Range	Default	Required	Variability
Btuh	≥ 0	0	No	constant

whPilotPwr=*float*

Pilot light consumption, included in fuel energy use of DHWHEATERs with whHeatSrc=FUEL.

Units	Legal Range	Default	Required	Variability
Btuh	≥ 0	0	No	hourly

whParElec=*float*

Parasitic electricity power, included in electrical energy use of all DHWHEATERs.

Units	Legal Range	Default	Required	Variability
W	≥ 0	0	No	hourly

whElecMtr=*mtrName*

Name of METER object, if any, by which DHWHEATER electrical energy use is recorded (under end use DHW).

Units	Legal Range	Default	Required	Variability
	name of a METER	*Parent DHWSYS wsElecMtr*	No	constant

whFuelMtr =*mtrName*

Name of METER object, if any, by which DHWHEATER fuel energy use is recorded (under end use DHW).

Units	Legal Range	Default	Required	Variability
	name of a METER	*Parent DHWSYS wsFuelMtr*	No	constant

endDHWHEATER

Optionally indicates the end of the DHWHEATER definition.

Units	Legal Range	Default	Required	Variability
		N/A	No	

4.24 DHWTANK

DHWTANK constructs an object representing one or more unfired water storage tanks in a DHWSYS. DHWTANK heat losses contribute to the water heating load.

wtName

Optional name of tank; give after the word "DHWTANK" if desired.

Units	Legal Range	Default	Required	Variability
	63 characters	*none*	No	constant

wtMult=*integer*

Number of identical tanks of this type. Any value > 1 is equivalent to repeated entry of the same DHWTANK.

Units	Legal Range	Default	Required	Variability
	> 0	1	No	constant

Tank heat loss is calculated hourly (note that default heat loss is 0) –

$$\text{qLoss} = \text{wtMult} \cdot (\text{wtUA} \cdot (\text{wtTTank} - \text{wtTEx}) + \text{wtXLoss})$$

wtUA=*float*

Tank heat loss coefficient.

Units	Legal Range	Default	Required	Variability
Btuh/°F	≥ 0	Derived from wtVol and wtInsulR	No	constant

wtVol=*float*

Specifies tank volume.

Units	Legal Range	Default	Required	Variability
gal	≥ 0	0	No	constant

wtInsulR=*float*

Specifies total tank insulation resistance. The input value should represent the total resistance from the water to the surroundings, including both built-in insulation and additional exterior wrap insulation.

Units	Legal Range	Default	Required	Variability
ft^2-°F/Btuh	$\geq .01$	0	No	constant

wtTEx=*float*

Tank surround temperature.

Units	Legal Range	Default	Required	Variability
°F	≥ 0	70	No	hourly

wtTTank=*float*

Tank average water temperature.

Units	Legal Range	Default	Required	Variability
°F	> 32 °F	Parent DHWSYSTEM wsTUse	No	hourly

wtXLoss=*float*

Additional tank heat loss. To duplicate CEC 2016 procedures, this value should be used to specify the fitting loss of 61.4 Btuh.

Units	Legal Range	Default	Required	Variability
Btuh	(any)	0	No	hourly

endDHWTank

Optionally indicates the end of the DHWTANK definition.

Units	Legal Range	Default	Required	Variability
		N/A	No	

4.25 DHWPUMP

DHWPUMP constructs an object representing a domestic hot water circulation pump (or more than one if identical).

wpName

Optional name of pump; give after the word "DHWPUMP" if desired.

Units	Legal Range	Default	Required	Variability
	63 characters	*none*	No	constant

wpMult=*integer*

Number of identical pumps of this type. Any value > 1 is equivalent to repeated entry of the same DHW-PUMP.

Units	Legal Range	Default	Required	Variability
	> 0	1	No	constant

wpPwr=*float*

Pump power.

Units	Legal Range	Default	Required	Variability
W	> 0	0	No	hourly

wpElecMtr=*mtrName*

Name of METER object, if any, to which DHWPUMP electrical energy use is recorded (under end use DHW).

Units	Legal Range	Default	Required	Variability
	name of a METER	*Parent DHWSYS wsElecMtr*	No	constant

endDHWPump

Optionally indicates the end of the DHWPUMP definition.

Units	Legal Range	Default	Required	Variability
		N/A	No	

4.26 DHWLOOP

DHWLOOP constructs one or more objects representing a domestic hot water circulation loop. The actual pipe runs in the DHWLOOP are specified by any number of DHWLOOPSEGs (see below). Circulation pumps are specified by DHWLOOPPUMPs (also below).

wlName

Optional name of loop; give after the word "DHWLOOP" if desired.

Units	Legal Range	Default	Required	Variability
	63 characters	*none*	No	constant

wlMult=*integer*

Number of identical loops of this type. Any value >1 is equivalent to repeated entry of the same DHWLOOP (and all child objects).

Units	Legal Range	Default	Required	Variability
	> 0	1	No	constant

wlFlow=*float*

Loop flow rate (when operating).

Units	Legal Range	Default	Required	Variability
gpm	≥ 0	6	No	hourly

wlTIn1=*float*

Inlet temperature of first DHWLOOPSEG.

Units	Legal Range	Default	Required	Variability
°F	> 0	130	No	hourly

wlRunF=*float*

Fraction of hour that loop circulation operates.

Units	Legal Range	Default	Required	Variability
–	≥ 0	1	No	hourly

wlFUA=*float*

DHWLOOPSEG pipe heat loss adjustment factor.

Units	Legal Range	Default	Required	Variability
–	> 0	1	No	constant

endDHWLoop

Optionally indicates the end of the DHWLOOP definition.

Units	Legal Range	Default	Required	Variability
		N/A	No	

4.27 DHWLOOPPUMP

DHWLOOPPUMP constructs an object representing a pump serving part a DHWLOOP. The model is identical to DHWPUMP *except* that that the electricity use calculation reflects wlRunF of the parent DHWLOOP.

wlpName

Optional name of pump; give after the word "DHWLOOPPUMP" if desired.

Units	Legal Range	Default	Required	Variability
	63 characters	*none*	No	constant

wlpMult=*integer*

Number of identical pumps of this type. Any value > 1 is equivalent to repeated entry of the same DHWPUMP.

Units	Legal Range	Default	Required	Variability
	> 0	1	No	constant

wlpPwr=*float*

Pump power.

Units	Legal Range	Default	Required	Variability
W	> 0	0	No	hourly

wlpElecMtr=*mtrName*

Name of METER object, if any, to which DHWLOOPPUMP electrical energy use is recorded (under end use DHW).

Units	Legal Range	Default	Required	Variability
	name of a METER	*Parent DHWSYS wsElecMtr*	No	constant

endDHWLOOPPUMP

Optionally indicates the end of the DHWPUMP definition.

Units	Legal Range	Default	Required	Variability
		N/A	No	

4.28 DHWLOOPSEG

DHWLOOPSEG constructs one or more objects representing a segment of the preceeding DHWLOOP. A DHWLOOP can have any number of DHWLOOPSEGs to represent the segments of the loop with possibly differing sizes, insulation, or surrounding conditions.

wgName

Optional name of segment; give after the word "DHWLOOPSEG" if desired.

Units	Legal Range	Default	Required	Variability
	63 characters	*none*	No	constant

wgTy=*choice*

Specifies the type of segment

SUPPLY	Indicates a supply segment (flow is sum of circulation and draw flow, child DHWLOOPBRANCHs permitted).
RETURN	Indicates a return segment (flow is only due to circulation, child DHWLOOPBRANCHs not allowed)

Units	Legal Range	Default	Required	Variability
–		–	Yes	constant

wgLength=*float*

Length of segment.

Units	Legal Range	Default	Required	Variability
ft	≥ 0	0	No	constant

wgSize=*float*

Nominal size of pipe. CSE assumes the pipe outside diameter = size + 0.125 in.

Units	Legal Range	Default	Required	Variability
in	> 0	1	Yes	constant

wgInsulK=*float*

Pipe insulation conductivity

Units	Legal Range	Default	Required	Variability
Btuh-ft/ft^2-^{o}F	> 0	0.02167	No	constant

wgInsulThk=*float*

Pipe insulation thickness

Units	Legal Range	Default	Required	Variability
in	≥ 0	1	No	constant

wgExH=*float*

Combined radiant/convective exterior surface conductance between insulation (or pipe if no insulation) and surround.

Units	Legal Range	Default	Required	Variability
Btuh/ft^2-°F	> 0	1.5	No	hourly

wgExT=*float*

Surrounding equivalent temperature.

Units	Legal Range	Default	Required	Variability
°F	> 0	70	No	hourly

wgFNoDraw=*float*

Fraction of hour when no draw occurs.

Units	Legal Range	Default	Required	Variability
°F	> 0	70	No	hourly

endDHWLoopSeg

Optionally indicates the end of the DHWLOOPSEG definition.

Units	Legal Range	Default	Required	Variability
		N/A	No	

4.29 DHWLOOPBRANCH

DHWLOOPBRANCH constructs one or more objects representing a branch pipe from the preceeding DHWLOOPSEG. A DHWLOOPSEG can have any number of DHWLOOPBRANCHs to represent pipe runs with differing sizes, insulation, or surrounding conditions.

wbNameOptional name of segment; give after the word "DHWLOOPBRANCH" if desired.

Units	Legal Range	Default	Required	Variability
	63 characters	*none*	No	constant

wbMult=*float*

Specifies the number of identical DHWLOOPBRANCHs. Note may be non-integer.

Units	Legal Range	Default	Required	Variability
–	> 0	1	No	constant

wbLength=*float*

Length of branch.

Units	Legal Range	Default	Required	Variability
ft	≥ 0	0	No	constant

wbSize=*float*

Nominal size of pipe. CSE assumes the pipe outside diameter = size + 0.125 in.

Units	Legal Range	Default	Required	Variability
in	> 0	–	Yes	constant

wbInsulK=*float*

Pipe insulation conductivity

Units	Legal Range	Default	Required	Variability
Btuh-ft/ft^2-°F	> 0	0.02167	No	constant

wbInsulThk=*float*

Pipe insulation thickness

Units	Legal Range	Default	Required	Variability
in	≥ 0	1	No	constant

wbExH=*float*

Combined radiant/convective exterior surface conductance between insulation (or pipe if no insulation) and surround.

Units	Legal Range	Default	Required	Variability
Btuh/ft^2-°F	> 0	1.5	No	hourly

wbExT=*float*

Surrounding equivalent temperature.

Units	Legal Range	Default	Required	Variability
°F	> 0	70	No	hourly

wbFlow=*float*

Branch flow rate assumed during draw.

Units	Legal Range	Default	Required	Variability
gpm	≥ 0	2	No	hourly

wbFWaste=*float*

Number of times during the hour when the branch volume is discarded.

Units	Legal Range	Default	Required	Variability
	≥ 0	0	No	hourly

endDHWLOOPBRANCH

Optionally indicates the end of the DHWLOOPBRANCH definition.

Units	Legal Range	Default	Required	Variability
		N/A	No	

4.30 PVARRAY

PVARRAY describes a photovoltaic panel system. The algorithms are based on the PVWatts calculator.

pvName

Name of photovoltaic array. Give after the word PVARRAY.

Units	Legal Range	Default	Required	Variability
	63 characters	*none*	No	constant

pvElecMtr=*choice*

Name of meter by which this PVARRAY's AC power out is recorded. Generated power is expressed as a negative value.

Units	Legal Range	Default	Required	Variability
	name of a METER	*none*	No	constant

pvEndUse=*choice*

Meter end use to which the PVARRAY's generated energy should be accumulated.

Clg	Cooling
Htg	Heating (includes heat pump compressor)
HPHTG	Heat pump backup heat
DHW	Domestic (service) hot water
DHWBU	Domestic (service) hot water heating backup (HPWH resistance)
FANC	Fans, AC and cooling ventilation
FANH	Fans, heating
FANV	Fans, IAQ venting
FAN	Fans, other purposes
AUX	HVAC auxiliaries such as pumps
PROC	Process
LIT	Lighting
RCP	Receptacles
EXT	Exterior lighting
REFR	Refrigeration
DISH	Dishwashing

DRY	Clothes drying
WASH	Clothes washing
COOK	Cooking
USER1	User-defined category 1
USER2	User-defined category 2
PV	Photovoltaic power generation

Units	Legal Range	Default	Required	Variability
	Codes listed above	PV	No	constant

pvDCSysSize=*float*

The rated photovoltaic system DC capacity/size as indicated by the nameplate.

Units	Legal Range	Default	Required	Variability
kW	x 0	*none*	Yes	constant

pvModuleType=*choice*

Type of module to model. The module type determines the refraction index and temperature coefficient used in the simulation. Alternatively, the "Custom" module type may be used in conjunction with user-defined input for *pvCoverRefrInd* and *pvTempCoeff*.

Module Type	pvCoverRefrInd	pvTempCoeff
Standard	1.0	-0.0026
Premium	1.3	-0.0019
ThinFilm	1.0	-0.0011
Custom	User-defined	User-defined

Units	Legal Range	Default	Required	Variability
	Standard Premium ThinFilm Custom	Standard	No	constant

pvCoverRefrInd=*float*

The refraction index for the coating applied to the module cover. A value of 1.0 represents refraction through air. Coatings have higher refraction indexes that capture more solar at lower angles of incidence.

Units	Legal Range	Default	Required	Variability
	x 1.0	1.0	No	constant

pvTempCoeff=*float*

The temperature coefficient how the efficiency of the module varies with the cell temperature. Values are typically negative.

Units	Legal Range	Default	Required	Variability
1/°F	*no restrictions*	-0.0026	No	constant

pvArrayType=*choice*

The type of array describes mounting and tracking options. Roof mounted arrays have a higher installed nominal operating cell temperature (INOCT) of 120 °F compared to the default of 113 °F. Array self-shading is not currently calculated for adjacent rows of modules within an array.

Units	Legal Range	Default	Required	Variability
	FixedOpenRack, FixedRoofMount, OneAxisTracking, TwoAxisTracking	FixedOpenRack	No	constant

pvTilt=*float*

The tilt of the photovoltaic array from horizontal. Values outside the range 0 to 360 are first normalized to that range. For one-axis tracking, defines the tilt of the rotation axis. Not used for two-axis tracking arrays. Should be omitted if pvVertices is given.

**Unit s	Legal Range	Default	**Require d	Variabili t y
degrees	unrestricted	from pvVertices (if given) else 0	No	hourly

pvAzm=*float*

Photovoltaic array azimuth (0 = north, 90 = east, etc.). If a value outside the range $0^o \leq x < 360^o$ is given, it is normalized to that range. For one-axis tracking, defines the azimuth of the rotation axis. Not used for two-axis tracking arrays. Should be omitted if pvVertices is given.

**Unit s	Legal Range	Default	Required	Variabilit y
degrees	unrestricted	from pvVertices (if given) else 0	No	hourly

pvVertices=*list of up to 36 floats*

Vertices of an optional polygon representing the position and shape of the photovoltaic array. The polygon is used to calculate the shaded fraction using an advanced shading model. Only PVARRAYs and SHADEXs are considered in the advanced shading model – PVARRAYs can be shaded by SHADEXs or other PVARRAYs. If pvVertices is omitted, the PVARRAY is assumed to be unshaded at all times. Advanced shading must be enabled via TOP exShadeModel. Note that the polygon is used only for evaluating shading; array capacity is specified by pvDCSysSize (above).

The values that follow pvVertices are a series of X, Y, and Z values for the vertices of the polygon using a coordinate system defined from a viewpoint facing north. X and Y values convey east-west and north-south location respectively relative to an arbitrary origin (positive X value are to the east; positive Y values are to the north). Z values convey height relative to the building 0 level and positive values are upward.

The vertices are specified in counter-clockwise order when facing the receiving surface of the PVARRAY. The number of values provided must be a multiple of 3. The defined polygon must be planar and have no crossing edges. When pvMounting=Building, the effective position of the polygon is modified in response to building rotation specified by TOP bldgAzm.

For example, to specify a rectangular photovoltaic array that is 10 x 20 ft, tilted 45 degrees, and facing south –

```
pvVertices = 0, 0, 15,   20, 0, 15,  20, 7.07, 22.07,  0, 7.07, 22.07
```

Units	Legal Range	Default	Required	Variability
ft	unrestricted	no polygon	9, 12, 15, 18, 21, 24, 27, 30, 33, or 36 values	constant

pvMounting=*choice*

Specified mounting location of this PVARRAY. pvMounting=Site indicates the array position is not altered by building rotation via TOP bldgAzm, while PVARRAYs with pvMounting=Building are assumed to rotate with the building.

Units	Legal Range	Default	Required	Variability
	Building or Site	Building	No	constant

pvGrndRefl=*float*

Ground reflectance used for calculating reflected solar incidence on the array.

Units	Legal Range	Default	Required	Variability
	$0 < x \leq 1.0$	0.2	No	hourly

pvDCtoACRatio=*float*

DC-to-AC ratio used to intentionally undersize the AC inverter. This is used to increase energy production in the beginning and end of the day despite the possibility of clipping peak sun hours.

Units	Legal Range	Default	Required	Variability
	$x > 0.0$	1.1	No	constant

pvInverterEff=*float*

AC inverter efficiency at rated DC power.

Units	Legal Range	Default	Required	Variability
	$0 < x \leq 1.0$	0.96	No	constant

pvSysLosses=*float*

Fraction of total DC energy lost. The total loss from a system is aggregated from several possible causes as illustrated below:

Loss Type	Default Assumption
Soiling	0.02

Loss Type	**Default Assumption**
Shading	0.03
Snow	0
Mismatch	0.02
Wiring	0.02
Connections	0.005
Light-induced degradation	0.015
Nameplate rating	0.01
Age	0
Availability	0.03
Total	**0.14**

Units	**Legal Range**	**Default**	**Required**	**Variability**
	$0 < x \leq 1.0$	0.14	No	hourly

endPVARRAY

Optionally indicates the end of the PVARRAY definition. Alternatively, the end of the definition can be indicated by END or by beginning another object.

Units	**Legal Range**	**Default**	**Required**	**Variability**
		N/A	No	constant

4.31 SHADEX

SHADEX describes an object that shades other building surfaces using an advanced shading model. Advanced shading calculations are provided only for PVARRAYs. Advanced shading must be enabled via Top exShadeModel.

sxName

Name of photovoltaic array. Give after the word SHADEX.

Units	**Legal Range**	**Default**	**Required**	**Variability**
	63 characters	*none*	No	constant

sxMounting=*choice*

Specifies the mounting location of the shade. sxMounting=Site indicates the SHADEX position is fixed and is not modified if the building is rotated. The position of SHADEXs with sxMounting=Building are modified to include the effect of building rotation specified via Top bldgAz

Units	**Legal Range**	**Default**	**Required**	**Variability**
	Building or Site	Site	No	constant

sxVertices=*list of up to 36 floats*

Vertices of a polygon representing the shape of the shading object.

The values that follow sxVertices are a series of X, Y, and Z values for the vertices of the polygon. The coordinate system is defined from a viewpoint facing north. X and Y values convey east-west and north-south location respectively relative to an arbitrary origin (positive X value are to the east; positive Y values are to the north). Z values convey height relative to the building 0 level and positive values are upward.

The vertices are specified in counter-clockwise order when facing the shading object from the south. The number of values provided must be a multiple of 3. The defined polygon must be planar and have no crossing edges. When sxType=Building, the effective position of the polygon reflects building rotation specified by TOP bldgAzm.

For example, to specify a rectangular shade "tree" that is 10 x 40 ft, facing south, and 100 ft to the south of the nominal building origin –

```
sxVertices = 5, -100, 0,   15, -100, 0,  15, -100, 40,  5, -100, 40
```

Units	Legal Range	Default	Required	Variability
ft	unrestricted	*none*	9, 12, 15, 18, 21, 24, 27, 30, 33 or 36 values	constant

endSHADEX

Optionally indicates the end of the SHADEX definition. Alternatively, the end of the definition can be indicated by END or by beginning another object.

Units	Legal Range	Default	Required	Variability
		N/A	No	constant

4.32 BATTERY

BATTERY describes input data for a model of an energy-storage system which is not tied to any specific energy storage technology. The battery model integrates the energy added and removed (accounting for efficiency losses). Note: although we use the term battery, the underlying model is flexible enough to model any energy storage system.

The modeler can set limits and constraints on capacities and flows and the associated efficiencies for this model.

btName

Name of the battery system. Given after the word BATTERY.

Units	Legal Range	Default	Required	Variability
	63 characters	*none*	No	constant

btMeter=*choice*

Name of a meter by which the BATTERY's power input/output (i.e., charge/discharge) is recorded. Charges to the BATTERY system would be seen as a positive powerflow while discharges from the BATTERY system would be seen as a negative value.

Units	Legal Range	Default	Required	Variability
	meter name	*none*	No	constant

btEndUse=*choice*

Meter end use to which the BATTERY's charged/discharged energy should be accumulated. Note that the battery end use is seen from the standpoint of a "load" on the electric grid. That is, when the battery is being charged, the end use will show up as positive. When the battery is being discharged (i.e., when it is offsetting other loads), it is seen as negative.

Clg	Cooling
Htg	Heating (includes heat pump compressor)
HPHTG	Heat pump backup heat
DHW	Domestic (service) hot water
DHWBU	Domestic (service) hot water heating backup (HPWH resistance)
FANC	Fans, AC and cooling ventilation
FANH	Fans, heating
FANV	Fans, IAQ venting
FAN	Fans, other purposes
AUX	HVAC auxiliaries such as pumps
PROC	Process
LIT	Lighting
RCP	Receptacles
EXT	Exterior lighting
REFR	Refrigeration
DISH	Dishwashing
DRY	Clothes drying
WASH	Clothes washing
COOK	Cooking
USER1	User-defined category 1
USER2	User-defined category 2
BT	Battery charge power
PV	Photovoltaic power generation

Units	Legal Range	Default	Required	Variability
	Codes listed above	BT	No	constant

btChgEff=*float*

The charging efficiency of storing electricity into the BATTERY system. A value of 1.0 means that no energy is lost and 100% of charge energy enters and is stored in the battery.

Units	Legal Range	Default	Required	Variability
	$0 \leq x \leq 1$	0.975	No	subhourly

btDschgEff=*float*

The discharge efficiency for when the BATTERY system is discharging power. A value of 1.0 means that no energy is lost and 100% of discharge energy leaves the system.

Units	Legal Range	Default	Required	Variability
	$0 \leq x \leq 1$	0.975	No	subhourly

btMaxCap=*float*

This is the maximum amount of energy that can be stored in the BATTERY system in kilowatt-hours. Once the BATTERY has reached its maximum capacity, no additional energy will be stored.

Units	Legal Range	Default	Required	Variability
KWhr	$x \geq 0$	16	No	constant

btInitSOE=*float*

The initial state of energy of the BATTERY system as a fraction of the total capacity. If `btInitSOE` is specified, the battery state-of-energy at the beginning of the actual simulation will be set to the amount specified, regardless of whether there was a warm-up period or not. If `btInitSOE` is NOT specififed, it will default to 1.0 (i.e., 100%) at the beginning of the warmup period (if any).

Units	Legal Range	Default	Required	Variability
	$0 \leq x \leq 0$	1.0	No	constant

btInitCycles=*int*

The number of cycles on the battery at the beginning of the run.

Units	Legal Range	Default	Required	Variability
number of cycles	$x \geq 0$	0	No	runly

btMaxChgPwr=*float*

The maximum rate at which the BATTERY can be charged in kilowatts (i.e., energy flowing *into* the BATTERY).

Units	Legal Range	Default	Required	Variability
kW	$x \geq 0$	4	No	subhourly

btMaxDschgPwr=*float*

The maximum rate at which the BATTERY can be discharged in kilowatts (i.e., energy flowing *out of* the BATTERY).

Units	Legal Range	Default	Required	Variability
kW	$x \geq 0$	4	No	subhourly

btChgReq=*float*

The power request to charge (or discharge if negative) the battery in kilowatts. The value of this parameter gets limited by the physical limitations of the battery and can be set by an expression to allow complex energy management/dispatch strategies.

Units	Legal Range	Default	Required	Variability
kW		0	No	subhourly

btUseUsrChg=*bool*

A boolean choice (YES/NO) that defaults to NO. If YES, then the user specified `btChgReq` will be used to set the battery's charge request; if false, the default strategy (i.e., to attempt to satisfy all loads and absorb all available excess power), will be used. Both the `btChgReq` and the default strategy requested power are literally *requests*: that is, more power will not be delivered than is available; more power will not be absorbed than capacity exits to store; and the battery's power limits will be respected.

Units	Legal Range	Default	Required	Variability
	YES, NO	NO	No	runly

endBATTERY

Optionally indicates the end of the BATTERY definition. Alternatively, the end of the definition can be indicated by END or by beginning another object.

Units	Legal Range	Default	Required	Variability
		N/A	No	constant

4.33 REPORTFILE

REPORTFILE allows optional specification of different or additional files to receive CSE reports.

By default, CSE generates several "reports" on each run showing the simulated HVAC energy use, the input statements specifying the run, any error or warning messages, etc. Different or additional reports can be specified using the REPORT object, described in Section 5.25, next.

All CSE reports are written to text files as plain ASCII text. The files may be printed (on most printers other than postscript printers) by copying them to your printer with the COPY command. Since many built-in reports are over 80 characters wide; you may want to set your printer for "compressed" characters or a small font first. You may wish to examine the report file with a text editor or LIST program before printing it. (?? Improve printing discussion)

By default, the reports are output to a file with the same name as the input file and extension .REP, in the same directory as the input file. By default, this file is formatted into pages, and overwrites any existing file of the same name without warning. CSE automatically generates a REPORTFILE object called "Primary" for this report file, as though the following input had been given:

```
REPORTFILE "Primary"
    rfFileName = <inputFile>.REP;
    // other members defaulted: rfFileStat=OVERWRITE; rfPageFmt=YES.
```

Using REPORTFILE, you can specify additional report files. REPORTs specified within a REPORTFILE object definition are output by default to that file; REPORTs specified elsewhere may be directed to a specific report file with the REPORT member *rpReportFile*. Any number of REPORTFILEs and REPORTs may be used in a run or session. Any number of REPORTs can be directed to each REPORTFILE.

Using ALTER (Section 4.5.1.2) with REPORTFILE, you can change the characteristics of the Primary report output file. For example:

```
ALTER REPORTFILE Primary
    rfPageFmt = NO;     // do not format into pages
    rfFileStat = NEW;   // error if file exists
```

rfName

Name of REPORTFILE object, given immediately after the word REPORTFILE. Note that this name, not the fileName of the report file, is used to refer to the REPORTFILE in REPORTs.

Units	Legal Range	Default	Required	Variability
	63 characters		No	constant

rfFileName=*path*

path name of file to be written. If no path is specified, the file is written in the current directory. The default extension is .REP.

Units	Legal Range	Default	Required	Variability
	file name, path and extension optional		Yes	constant

rfFileStat=*choice*

Choice indicating what CSE should do if the file specified by *rfFileName*already exists:

OVERWRITE	Overwrite pre-existing file.
NEW	Issue error message if file exists at beginning of session. If there are several runs in session using same file, output from runs after the first will append.
APPEND	Append new output to present contents of existing file.

If the specified file does not exist, it is created and *rfFileStat* has no effect.

Units	Legal Range	Default	Required	Variability
	OVERWRITE, NEW, APPEND	OVERWRITE	No	constant

rfPageFmt=*Choice*

Choice controlling page formatting. Page formatting consists of dividing the output into pages (with form feed characters), starting a new page before each report too long to fit on the current page, and putting headers and footers on each page. Page formatting makes attractive printed output but is a distraction when examining the output on the screen and may inappropriate if you are going to further process the output with another program.

Yes	Do page formatting in this report file.
No	Suppress page formatting. Output is continuous, uninterrupted by page headers and footers or large blank spaces.

Units	Legal Range	Default	Required	Variability
	Yes, No	Yes	No	constant

Unless page formatting is suppressed, the page formats for all report files are controlled by the TOP members *repHdrL, repHdrR, repLPP, repTopM, repBotM,*and *repCPL*, described in Section 5.1.

Each page header shows the *repHdrL* and *repHdrR* text, if given.

Each page footer shows the input file name, run serial number within session (see *runSerial* in Section 5.1), user-input *runTitle* (see Section 5.1), date and time of run, and page number in file.

Vertical page layout is controlled by *repLPP, repTopM,* and *repBotM* (Section 5.1). The width of each header and footer is controlled by *repCPL*. Since many built-in reports are now over 80 columns wide, you may want to use repCPL=120 or repCPL=132 to make the headers and footers match the text better.

In addition to report file *page* headers and footers, individual REPORTs have *REPORT* headers and footers related to the report content. These are described under REPORT, Section 5.25.

endReportFile

Optionally indicates the end of the report file definition. Alternatively, the end of the report file definition can be indicated by END or by beginning another object.

Units	**Legal Range**	**Default**	**Required**	**Variability**
		N/A	No	constant

4.34 REPORT

REPORT generates a report object to specify output of specific textual information about the results of the run, the input data, the error messages, etc. The various report types available are enumerated in the description of *rpType* in this section, and may be described at greater length in Section 6.

REPORTs are output by CSE to files, via the REPORTFILE object (previous section). After CSE has completed, you may print the report file(s), examine them with a text editor or by TYPEing, process them with another program, etc., as desired.

REPORTs that you do not direct to a different file are written to the automatically-supplied "Primary" report file, whose file name is (by default) the input file name with the extension changed to .REP.

Each report consists of a report header, one or more data rows, and a report footer. The header gives the report type (as specified with *rpType*, described below), the frequency (as specified with *rpFreq)*, the month or date where appropriate, and includes headings for the report's columns where appropriate.

Usually a report has one data row for each interval being reported. For example, a daily report has a row for each day, with the day of the month shown in the first column.

The report footer usually contains a line showing totals for the rows in the report.

The header-data-footer sequence is repeated as necessary. For example, a daily report extending over more than one month has a header-data-footer sequence for each month. The header shows the month name; the data rows show the day of the month; the footer contains totals for the month.

In addition to the headers and footers of individual reports, the report file has (by default) *page headers* and *footers*, described in the preceding section.

Default Reports: CSE generates the following reports by default for each run, in the order shown. They are output by default to the "Primary" report file. They may be ALTERed or DELETEd as desired, using the object names shown.

rpName	rpType	Additional members
Err	ERR	
eb	ZEB	rpFreq=MONTH; rpZone=SUM;
Log	LOG	
Inp	INP	

Any reports specified by the user and not assigned to another file appear in the Primary report file between the default reports "eb" and "Log", in the order in which the REPORT objects are given in the input file.

Because of the many types of reports supported, the members required for each REPORT depend on the report type and frequency in a complex manner. When in doubt, testing is helpful: try your proposed REPORT specification; if it is incomplete or overspecified, CSE will issue specific error messages telling you what additional members are required or what inappropriate members have been given and why.

rpName

Name of report. Give after the word REPORT.

Units	Legal Range	Default	Required	Variability
	63 characters	*none*	No	constant

rpReportfile=*rfname*

Name of report file to which current report will be written. If omitted, if REPORT is within a REPORTFILE object, report will be written to that report file, or else to REPORTFILE "Primary", which (as described in previous section) is automatically supplied and by default uses the file name of the input file with the extension .REP.

Units	Legal Range	Default	Required	Variability
	name of a *REPORTFILE*	current *REPORTFILE*, if any, else "Primary"	No	constant

rpType=*choice*

Choice indicating report type. Report types may be described at greater length, with examples, in Section 6.

ERR	Error and warning messages. If there are any such messages, they are also displayed on the screen *AND* written to a file with the same name as the input file and extension .ERR. Furthermore, * *many error messages are repeated in the INP report.
LOG	Run "log". As of July 1992, contains only CSE version number; should be enhanced or deleted.??
INP	Input echo: shows the portion of the input file used to specify this run. Does not repeat descriptions of objects left from prior runs in the same session when CLEAR is not used. Error and warning messages relating to specific lines of the input are repeated after or near the line to which they relate, prefixed with "?". Lines not used due to a preprocessor #if command (Section 4.4.4) with a false expression are prefixed with a "0" in the leftmost column; all preprocessor command lines are prefixed with a "#" in that column.
SUM	Run summary. As of July 1992, *NOT IMPLEMENTED*: generates no output and no error message. Should be defined and implemented, or else deleted??.
ZDD	Zone data dump. Detailed dump of internal simulation values, useful for verifying that your input is as desired. Should be made less cryptic (July 1992)??. Requires *rpZone*.
ZST	Zone statistics. Requires *rpZone*.
ZEB	Zone energy balance. Requires *rpZone*.
MTR	Meter report. Requires *rpMeter*.
DHWMTR	DHW meter report. Requires *rpDHWMeter*
UDT	User-defined table. Data items are specified with REPORTCOL commands (next section). Allows creating almost any desired report by using CSE expressions to specify numeric or string values to tabulate; "Probes" may be used in the expressions to access CSE internal data.

Units	Legal Range	Default	Required	Variability
	see above		Yes	constant

The next three members specify how frequently values are reported and the start and end dates for the REPORT. They are not allowed with *rpTypes* ERR, LOG, INP, SUM, and ZDD, which involve no time-varying data.

rpFreq=*choice*

Report Frequency: specifies interval for generating rows of report data:

YEAR	at run completion
MONTH	at end of each month (and at run completion if mid-month)
DAY	at end of each day
HOUR	at end of each hour
HOURANDSUB	at end of each subhour AND at end of hour
SUBHOUR	at end of each subhour

rpFreq values of HOURANDSUB and SUBHOUR are not supported in some combinations with data selection of ALL or SUM.

We recommend using HOURly and more frequent reports sparingly, to report on only a few typical or extreme days, or to explore a problem once it is known what day(s) it occurs on. Specifying such reports for a full-year run will generate a huge amount of output and cause extremely slow CSE execution.

Units	Legal Range	Default	Required	Variability
	choices above		per rpType	constant

rpDayBeg=*date*

Initial day of period to be reported. Reports for which *rpFreq* = YEAR do not allow specification of *rpDayBeg* and *rpDayEnd*; for MONTH reports, these members default to include all months in the run; for DAY and shorter-interval reports, *rpDayBeg* is required and *rpDayEnd* defaults to *rpDayBeg*.

Units	Legal Range	Default	Required	Variability
	date	first day of simulation if *rpFreq* = MONTH	Required for *rpTypes* ZEB, ZST, MTR, AH, and UDT if *rpFreq* is DAY, HOUR, HOURANDSUB, or SUBHOUR	constant

rpDayEnd=*date*

Final day of period to be reported, except for YEAR reports.

Units	Legal Range	Default	Required	Variability
	date	last day of simulation if *rpFreq*= MONTH, else *rpDayBeg*	No	constant

Units	Legal Range	Default	Required	Variability

rpZone=*znName*

Name of ZONE for which a ZEB, ZST, or ZDD report is being requested. For *rpType* ZEB or ZST, you may use *rpZone*=SUM to obtain a report showing only the sum of the data for all zones, or *rpZone*=ALL to obtain a report showing, for each time interval, a row of data for each zone plus a sum-of-zones row.

Units	Legal Range	Default	Required	Variability
	name of a *ZONE*, ALL, SUM		Required for *rpTypes* ZDD, ZEB, and ZST.	constant

rpMeter=*mtrName*

Specifies meter(s) to be reported, for *rpType*=MTR.

Units	Legal Range	Default	Required	Variability
	name of a *METER*, ALL, SUM		Required for *rpType*=MTR	constant

rpDHWMeter=*dhwMtrName*

Specifies DHW meter(s) to be reported, for *rpType*=DHWMTR.

Units	Legal Range	Default	Required	Variability
	name of a *DHWMETER*, ALL, SUM		Required for *rpType*=DHWMTR	constant

rpBtuSf=*float*

Scale factor to be used when reporting energy values. Internally, all energy values are represented in Btu. This member allows scaling to more convenient units for output. *rpBtuSf* is not shown in the output, so if you change it, be sure the readers of the report know the energy units being used. *rpBtuSf* is not applied in UDT reports, but column values can be scaled as needed with expressions.

Units	Legal Range	Default	Required	Variability
	any multiple of ten	1,000,000: energy reported in MBtu.	No	constant

rpCond=*expression*

Conditional reporting flag. If given, report rows are printed only when value of expression is non-0. Permits selective reporting according to any condition that can be expressed as a CSE expression. Such conditional reporting can be used to shorten output and make it easy to find data of interest when you are only interested in the information under exceptional conditions, such as excessive zone temperature. Allowed with *rpTypes* ZEB, ZST, MTR, AH, and UDT.

Units	Legal Range	Default	Required	Variability
	any numeric expression	1 (reporting enabled)	No	subhour /end of interval

rpCPL=*int*

Characters per line for a User-Defined report. If widths specified in REPORTCOLs add up to more than this, a message occurs; if they total substantially less, additional whitespace is inserted between columns to make the report more readable. If rpCpl = -1, the report width determined based on required space with a single between each column. Not allowed if *rpType* is not UDT.

Units	Legal Range	Default	Required	Variability
	x = -1 or x > 78	top level	*repCPL* No	constant

rpTitle=*string*

Title for use in report header of User-Defined report. Disallowed if *rpType* is not UDT.

Units	Legal Range	Default	Required	Variability
		"User-defined Report"	No	constant

rpHeader=*choice*

Use NO to suppress the report header which gives the report type, zone, meter, or air handler being reported, time interval, column headings, etc. One reason to do this might be if you are putting only a single report in a report file and intend to later embed the report in a document or process it with some other program (but for the latter, see also EXPORT, below).

Use with caution, as the header contains much of the identification of the data. For example, in an hourly report, only the hour of the day is shown in each data row; the day and month are shown in the header, which is repeated for each 24 data rows.

See REPORTFILE member *rfPageFmt*, above, to control report *FILE* page headers and footers, as opposed to *REPORT* headers and footers.

Units	Legal Range	Default	Required	Variability
	YES, NO	YES	No	constant

rpFooter=*choice*

Use NO to suppress the report footers. The report footer is usually a row which sums hourly data for the day, daily data for the month, or monthly data for the year. For a report with *rpZone, rpMeter,*or *rpAh* = ALL, the footer row shows sums for all zones, meters, or air handlers. Sometimes the footer is merely a blank line.

Units	Legal Range	Default	Required	Variability
	YES, NO	YES	No	constant

endReport

Optionally indicates the end of the report definition. Alternatively, the end of the report definition can be indicated by END or by beginning another object.

Units	Legal Range	Default	Required	Variability
		N/A	No	constant

4.35 REPORTCOL

Each REPORTCOL defines a single column of a User Defined Table (UDT) report. REPORTCOLs are not used with report types other than UDT.

Use as many REPORTCOLs as there are values to be shown in each row of the user-defined report. The values will appear in columns, ordered from left to right in the order defined. Be sure to include any necessary values to identify the row, such as the day of month, hour of day, etc. CSE supplies *NO* columns automatically.

colName

Name of REPORTCOL.

Units	Legal Range	Default	Required	Variability
	63 characters	*none*	No	constant

colReport=*rpName*

Name of report to which current report column belongs. If REPORTCOL is given within a REPORT object, then *colReport* defaults to that report.

Units	Legal Range	Default	Required	Variability
	name of a *REPORT*	*current report, if any*	Unless in a *REPORT*	constant

colVal=*expression*

Value to show in this column of report.

Units	Legal Range	Default	Required	Variability
	any numeric or string expression		Yes	subhour /end interval

colHead=*string*

Text used for column head.

Units	Legal Range	Default	Required	Variability
		colName or blank	No	constant

colGap=*int*

Space between (to left of) column, in character positions. Allows you to space columns unequally, to emphasize relations among columns or to improve readability. If the total of the *colGaps* and *colWids* in the report's REPORTCOLs is substantially less than the REPORT's *rpCPL* (characters per line, see REPORT), CSE will insert additional spaces between columns. To suppress these spaces, use a smaller *rpCPL* or use *rpCPL* = -1.

Units	Legal Range	Default	Required	Variability
	$x \geq 0$	1	No	constant

colWid=*int*

Column width.

Units	Legal Range	Default	Required	Variability
	$x \geq 0$	10	No	constant

colDec=*int*

Number of digits after decimal point.

Units	Legal Range	Default	Required	Variability
	$x \geq 0$	*flexible format*	No	constant

colJust=*choice*

Specifies positioning of data within column:

Left	Left justified
Right	Right justified

endReportCol

Optionally indicates the end of the report column definition. Alternatively, the end of the report column definition can be indicated by END or by beginning another REPORTCOL or other object.

Units	Legal Range	Default	Required	Variability
		N/A	No	constant

4.36 EXPORTFILE

EXPORTFILE allows optional specification of different or additional files to receive CSE EXPORTs.

EXPORTs contain the same information as reports, but formatted for reading by other programs rather than by people. By default, CSE generates no exports. Exports are specified via the EXPORT object, described in Section 5.28 (next). As for REPORTs, CSE automatically supplies a primary export file; it has the same name and path as the input file, and extension .csv.

Input for EXPORTFILEs and EXPORTs is similar to that for REPORTFILEs and REPORTs, except that there is no page formatting. Refer to their preceding descriptions (Sections 5.24 and 5.25) for more additional discussion.

xfName

Name of EXPORTFILE object.

Units	Legal Range	Default	Required	Variability
	63 characters		No	constant

xfFileName=*string*

path name of file to be written. If no path is specified, the file is written in the current directory. If no extension is specified, .csv is used.

Units	Legal Range	Default	Required	Variability
	file name, path and extension optional		Yes	constant

xfFileStat=*choice*

What CSE should do if file *xfFileName* already exists:

OVERWRITE	Overwrite pre-existing file.
NEW	Issue error message if file exists.
APPEND	Append new output to present contents of existing file.

If the specified file does not exist, it is created and *xfFileStat* has no effect.

Units	Legal Range	Default	Required	Variability
	OVERWRITE, NEW, APPEND	OVERWRITE	No	constant

endExportFile

Optionally indicates the end of the export file definition. Alternatively, the end of the Export file definition can be indicated by END or by beginning another object.

Units	Legal Range	Default	Required	Variability
		N/A	No	constant

4.37 EXPORT

Exports contain the same information as CSE reports, but in a "comma-quote" format intended for reading into a spreadsheet or other program for further processing, plotting, special print formatting, etc.

No exports are generated by default; each desired export must be specified with an EXPORT object.

Each row of an export contains several values, separated by commas, with quotes around string values. The row is terminated with a carriage return/line feed character pair. The first fields of the row identify the data. Multiple fields are used as necessary to identify the data. For example, the rows of an hourly ZEB export begin with the month, day of month, and hour of day. In contrast, reports, being subject to a width limitation, use only a single column of each row to identify the data; additional identification is put in the header. For example, an hourly ZEB Report shows the hour in a column and the day and month in the header; the header is repeated at the start of each day. The header of an export is never repeated.

Depending on your application, if you specify multiple exports, you may need to place each in a separate file. Generate these files with EXPORTFILE, preceding section. You may also need to suppress the export header and/or footer, with *exHeader* and/or *exFooter*, described in this section.

Input for EXPORTs is similar to input for REPORTs; refer to the REPORT description in Section 5.25 for further discussion of the members shown here.

exName

Name of export. Give after the word EXPORT.

Units	Legal Range	Default	Required	Variability
	63 characters	*none*	No	constant

exExportfile=*fname*

Name of export file to which current export will be written. If omitted, if EXPORT is within an EXPORTFILE object, report will be written to that export file, or else to the automatically-supplied EXPORTFILE "Primary", which by default uses the name of the input file with the extension .csv.

Units	Legal Range	Default	Required	Variability
	name of an *EXPORTFILE*	current *EXPORTFILE*, if any, else "Primary"	No	constant

exType=*choice*

Choice indicating export type. See descriptions in Section 5.22, REPORT. While not actually disallowed, use of *exType* = ERR, LOG, INP, or ZDD is unexpected.

Units	Legal Range	Default	Required	Variability
	ZEB, ZST, MTR, DHWMTR, AH, UDT, or SUM		Yes	constant

exFreq=*choice*

Export Frequency: specifies interval for generating rows of export data:

Units	Legal Range	Default	Required	Variability
	YEAR, MONTH, DAY, HOUR, HOURANDSUB, SUBHOUR		Yes	constant

exDayBeg=*date*

Initial day of export. Exports for which *exFreq* = YEAR do not allow specification of *exDayBeg* and *exDayEnd*; for MONTH exports, these members are optional and default to include the entire run; for DAY and shorter-interval exports, *exDayBeg* is required and *exDayEnd* defaults to *exDayBeg*.

Units	Legal Range	Default	Required	Variability
	date	first day of simulation if *exFreq* = MONTH	Required for *exTypes* ZEB, ZST, MTR, AH, and UDT if *exFreq* is DAY, HOUR, HOURANDSUB, or SUBHOUR	constant

exDayEnd=*date*

Final day of export period, except for YEAR exports.

Units	Legal Range	Default	Required	Variability
	date	last day of simulation if *exFreq*= MONTH, else *exDayBeg*	No	constant

exZone=*znName*

Name of ZONE for which a ZEB, ZST, or ZDD export is being requested; ALL and SUM are also allowed except with *exType* = ZST.

Units	**Legal Range**	**Default**	**Required**	**Variability**
	name of a *ZONE*, ALL, SUM		Required for *exTypes* ZDD, ZEB, and ZST.	constant

exMeter=*mtrName*

Specifies meter(s) whose data is to be exported, for *exType*=MTR.

Units	**Legal Range**	**Default**	**Required**	**Variability**
name of a *M	ETER*, ALL, SUM	Required for	*exType*=MTR constant	

exDHWMeter=*dhwMtrName*

Specifies DHW meter(s) whose data is to be exported, for *exType*=DHWMTR.

Units	**Legal Range**	**Default**	**Required** **V	ariability**
	name of a *DHWMETER*, ALL, SUM		Required for *exType*=DHWMTR	constant

exAh=ah*Name*

Specifies air handler(s) to be exported, for *exType*=AH.

Units	**Legal Range**	**Default**	**Required**	**Variability**
	name of an *AIRHANDLER*, ALL, SUM		Required for *exType*=AH	constant

exBtuSf=*float*

Scale factor used for exported energy values.

Units	**Legal Range**	**Default**	**Required**	**Variability**
	any multiple of ten	1,000,000: energy exported in MBtu.	No	constant

exCond=*expression*

Conditional exporting flag. If given, export rows are generated only when value of expression is non-0. Allowed with *exTypes* ZEB, ZST, MTR, AH, and UDT.

Units	**Legal Range**	**Default**	**Required**	**Variability**
	any numeric expression	1 (exporting enabled)	No	subhour /end of interval

exTitle=*string*

Title for use in export header of User-Defined export. Disallowed if *exType* is not UDT.

Units	**Legal Range**	**Default**	**Required**	**Variability**
		"User-defined Export"	No	constant

exHeader=*choice*

Use NO to suppress the export header which gives the export type, zone, meter, or air handler being exported, time interval, column headings, etc. You might do this if the export is to be subsequently imported to a program that is confused by the header information. Alternatively, one may use COLUMNSONLY to print only the column headings. This can be useful when plotting CSV data in a spreadsheet tool or DView.

If not suppressed, the export header shows, in four lines:

runTitle and *runSerial* (see Section 5.1);the run date and timethe export type ("Energy Balance", "Statistics", etc., or *exTitle* if given)and frequency ("year", "day", etc.)a list of field names in the order they will be shown in the data rows("Mon", "Day", "Tair", etc.)

The *specific* month, day, etc. is NOT shown in the export header (as it is shown in the report header), because it is shown in each export row.

The field names may be used by a program reading the export to identify the data in the rows which follow; if the program does this, it will not require modification when fields are added to or rearranged in the export in a future version of CSE.

Units	Legal Range	Default	Required	Variability
	YES, NO, COLUMNSONLY	YES	No	constant

exFooter=*choice*

Use NO to suppress the blank line otherwise output as an export "footer". (Exports do not receive the total lines that most reports receive as footers.)

Units	Legal Range	Default	Required	Variability
	YES, NO	YES	No	constant

endExport

Optionally indicates the end of the export definition. Alternatively, the end of the export definition can be indicated by END or by beginning another object.

Units	Legal Range	Default	Required	Variability
		N/A	No	constant

4.38 EXPORTCOL

Each EXPORTCOL defines a single datum of a User Defined Table (UDT) export; EXPORTCOLs are not used with other export types.

Use as many EXPORTCOLs as there are values to be shown in each row of the user-defined export. The values will appear in the order defined in each data row output. Be sure to include values needed to identify the data, such as the month, day, and hour, as appropriate – these are NOT automatically supplied in user-defined exports.

EXPORTCOL members are similar to the corresponding REPORTCOL members. See Section 5.265.1.5 for further discussion.

colName

Name of EXPORTCOL.

Units	Legal Range	Default	Required	Variability
	63 characters	*none*	No	constant

colExport=*exName*

Name of export to which this column belongs. If the EXPORTCOL is given within an EXPORT object, then *colExport* defaults to that export.

Units	Legal Range	Default	Required	Variability
	name of an *EXPORT*	*current export, if any*	Unless in an *EXPORT*	constant

colVal=*expression*

Value to show in this position in each row of export.

Units	Legal Range	Default	Required	Variability
	any numeric or string expression		Yes	subhour /end interval

colHead=*string*

Text used for field name in export header.

Units	Legal Range	Default	Required	Variability
		colName or blank	No	constant

colWid=*int*

Maximum width. Leading and trailing spaces and non-significant zeroes are removed from export data to save file space. Specifying a *colWid* less than the default may reduce the maximum number of significant digits output.

Units	Legal Range	Default	Required	Variability
	$x \geq 0$	13	No	constant

colDec=*int*

Number of digits after decimal point.

Units	Legal Range	Default	Required	Variability
	$x \geq 0$	*flexible format*	No	constant

colJust=*choice*

Specifies positioning of data within column:

Left	Left justified
Right	Right justified

endExportCol

Optionally indicates the end of the EXPORTCOL. Alternatively, the end of the definition can be indicated by END or by beginning another object.

Units	Legal Range	Default	Required	Variability
		N/A	No	constant

4.39 IMPORTFILE

IMPORTFILE allows specification of a file from which external data can be accessed using the import() and importStr() functions. This allows external values to be referenced in expressions. Any number of IMPORTFILEs can be defined and any number of import()/importStr() references can be made to a give IMPORTFILE.

Import files are text files containing an optional header and comma-separated data fields. With the header present, the structure of an import file matches that of an EXPORT file. This makes it convenient to import unmodified files EXPORTed from prior runs. The file structure is as follows (noting that the header in lines 1-4 should not be present when imHeader=NO) –

Line	Contents	Notes
1	*runTitle*, *runNumber*	read but not checked
2	*timestamp*	in quotes, read but not checked
3	*title*, *freq*	should match imTitle and imFreq
4	*colName1*,*colName2*,...	comma separated column names in quotes
5 ..	*val1*,*val2*,...	comma separated values (string values in quotes)

Example import file imp1.csv

```
"Test run",001
"Fri 04-Nov-16  10:54:37 am"
"Daily Data","Day"
"Mon","Day","Tdb","Twb"
1,1,62.2263,53.2278
1,2,61.3115,52.8527
1,3,60.4496,52.4993
1,4,60.2499,52.4174
1,5,60.9919,52.7216
1,6,61.295,52.8459
1,7,62.3178,53.2654
1,8,62.8282,53.4747
(... continues for 365 data lines ...)
```

Example IMPORTFILE use (reading from imp1.csv)

```
// ... various input statements ...

IMPORTFILE Example imFileName="imp1.csv" imFreq=Day imTitle="Daily Data"
...
// Compute internal gain based on temperature read from import file.
// result is 3000 W per degree temperature is above 60.
// Note gnPower can have hourly variability, but here varies daily.
GAIN gnPower = 3000 * max( 0, import(Example,"Tdb") - 60) / 3.412
...
```

Notes

- As usual, file order is not important – IMPORTFILEs can be referenced before they are defined.
- Columns are referenced by 1-based index or column names (assuming file header is present). In the example above, "Tdb" could be replaced by 3.

imName

Name of IMPORTFILE object (for reference from Import()).

Units	Legal Range	Default	Required	Variability
	63 characters		No	constant

imFileName=*string*

Gives path name of file to be read. If directory is specified, CSE first looks for the file the current directory and searches include paths specified by the -I command line parameter (if any).

Units	Legal Range	Default	Required	Variability
	file name, path optional		Yes	constant

imTitle=*string*

Title expected to be found on line 3 of the import file. A warning is issued if a non-blank imTitle does not match the import file title.

Units	Legal Range	Default	Required	Variability
	Text string	(blank)	No	constant

imFreq=*choice*

Specifies the interval at which CSE reads from the import file. Data is read at the beginning of the indicated interval and buffered in memory for access in expressions via import() or importStr().

Units	Legal Range	Default	Required	Variability
	YEAR, MONTH, DAY, or HOUR		Yes	constant

imHeader=*choice*

Indicates whether the import file include a 4 line header, as described above. If NO, the import file should contain only comma-separated data rows and data items can be referenced only by 1-based column number.

Units	Legal Range	Default	Required	Variability
	YES NO	YES	No	constant

endImportFile

Optionally indicates the end of the import file definition. Alternatively, the end of the import file definition can be indicated by END or by beginning another object.

Units	Legal Range	Default	Required	Variability
		N/A	No	constant

5 Output Reports

CSE report data is accumulated during simulation and written to the report file at the end of the run. Some reports are generated by default and cannot be turned off. There are a set of predefined reports which may be requested in the input. The user may also define custom reports which include many CSE internal variables. Reports may accumulate data on an a variety of frequencies including subhourly, hourly, daily, monthly, and annual (run) intervals.

5.1 Units

The default units for CSE reports are:

Energy	mBtu, millions of Btu (to convert to kWh divide by 292)
Temperature	degrees Farenheit
Air Flow	cfm (cubic feet per minute)

5.2 Time

Hourly reports show hour 1 through 24 where hour 1 includes the time period from midnight to 1 AM. By default, CSE specifies that January first is a Thursday and the simulation occurs on a non-leap year. Daylight savings is in effect from the second Sunday of March on which CSE skips hour 3 until the first Sunday of November when CSE simulates 25 hours. These calendar defaults can be modified as required.

5.3 METER Reports

A Meter Report displays the energy use of a METER object, a user-defined "device" that records energy consumption of equipment as simulated by CSE. CSE allows the user to define as many meters as desired and to assign any energy using device to any meter.

Meters account for energy use in pre-defined categories, called *end uses, that are documented with METER.*

5.4 Energy Balance Report

The Energy Balance Report displays the temperature and sensible and latent heat flows into and out of the air of a single zone. Sign conventions assume that a positive flow increases the air temperature. Heat flow from a warm mass element such as a concrete wall into the zone air is defined as a positive flow, heat flow from air into mass is negative. Solar gain into the zone is defined as a positive heat flow. Solar gain that is incident on and absorbed directly into a mass element is shown as both a positve in the SOLAR column (gain to the zone) and a negative in the MASS column (lost from the zone to the mass).

In a real building zone energy and moisture flows must balance due to the laws of physics. CSE uses approximate solutions for the energy and moisture balances and displays the net balance which is a measure of internal calculation error.

The following items are displayed (using the abbreviations shown in the report headings):

Tair	Air temperature in the zone (since CSE uses combined films this is technically the effective temperature and includes radiant effects).
WBair	Wet Bulb temperature in the zone.
Cond	Heat flow through light weight surfaces from or to the outdoors.
InfS	Sensible infiltration heat flow from outdoors.
Slr	Solar gain through glazing (net) and solar gains absorbed by light surfaces and transmitted into the zone air.
IgnS	Sensible internal gains from lights, equipment, people, etc.
Mass	Net heat flow to (negative) and from (positive) the mass elements of the zone.
Izone	Net heat flows to other zones in the building.
MechS	Net heat flows from heating, cooling and ventilation.
BALS	The balance (error) calculated by summing the sensible gains and losses.
InfL	Latent infiltration heat flow.
IgnL	Latent internal gains.
AirL	Latent heat absorbed (negative) or released (positive) by changes in the room air moisture content.
MechL	Latent heat added or removed by cooling or ventilation.
BalL	The balance (error) calculated by summing the sensible gains and losses.

5.5 Air Handler Load Report

The Air Handler Load Report displays conditions and loads at the peak load hours for the air handler for a single zone. The following items are displayed:

PkVf	Peak flow (cfm) at supply fan
VfDs	Supply fan design flow (same as peak for E10 systems)
PkQH	Peak heat output from heating coil.
Hcapt	Rated capacity of heat coil

The rest are about the cooling coil. Most of the columns are values at the time of peak part load ratio (plr). Note that, for example, the peak sensible load is the sensible load at the time of peak part load ratio, even if there was a higher sensible load at another time when the part load ratio was smaller.

PkMo	Month of cooling coil peak plr, 1-12
Dy	Day of month 1-31 of peak
Hr	Hour of day 1-24 of cooling coil peak plr.
Tout	Outdoor drybulb temperature at time of cooling coil peak plr.
Wbou	Outdoor wetbulb similarly
Ten	Cooling coil entering air temperature at time of peak plr.
Wben	Entering wetbulb similarly
Tex	Exiting air temperature at plr peak
	WbexExiting air wetbulb similarly
-PkQs	Sensible load at time of peak plr, shown positive.
-PkQl	Latent load likewise
-PkQC	Total load – sum of PkQs and PkQl

CPlr	Peak part load ratio: highest fraction of coil's capacity used, reflecting both fraction of maximum output under current conditions used when on and fraction of the time the fan is on. The maximum output under actual conditions can vary considerably from the rated capacity for DX coils. The fraction of maximum output used can only be 1.0 if the sensible and total loads happen to occur in the same ratio as the sensible and total capacities. The time the fan is on can be less than 1.0 for residential systems in which the fan cycles on with the compressor. For example, if at the cooling peak the coil ran at .8 power with the fan on .9 of the time, a CPlr of .72 would be reported. The preceding 12 columns are values at the time this peak occured.
Ccapt	Cooling coil rated total capacity
Ccaps	Rated sensible capacity.

5.6 Air Handler Report

The Air Handler Load Report displays conditions and heat flows in the air handler for the time period specified. It is important to note that the air handler report only accumulates data if the air handler is on during an hour. The daily and monthly values are averages of the hours the air handler was on and DO NOT INCLUDE OFF HOUR VALUES. The following items are displayed:

Tout	Outdoor drybulb temperature during hours the air handler was on.
Wbou	Outdoor wetbulb temperature similarly.
Tret	Return air dry bulb temperature during hours the air handler was on before return duct losses or leaks.
Wbre	Return air wetbulb similarly
po	Fraction outside air including economizer damper leakage, but not return duct leakage.
Tmix	Mixed air dry bulb temperature – after return air combined with outside air; after return fan, but before supply fan and coil(s).
Wbmi	Mixed air wet bulb temperature, similarly.
Tsup	Supply air dry bulb temperature to zone terminals – after coil(s) and air handler supply duct leak and loss; (without in zone duct losses after terminals).
WBsu	Supply air wet bulb temperature similarly.
HrsOn	Hours during which the fan operated at least part of the time.
FOn	Fraction of the time the fan was on during the hours it operated (HrsOn). CHECK FOR VAV, IS IT FLOW OR TIME
VF	Volumetric flow, measured at mix point/supply fan/coils; includes air that leaks out of supply duct and is thus non-0 even when zone terminals are taking no flow
Qheat	Heat energy added to air stream by heat coil, if any, MEASURED AT COIL not as delivered to zones (see Qload).
Qsens, Qlat and Qcool	Sensible, latent, and total heat added to air stream (negative values) by cooling coil, MEASURED AT COIL, including heat cancelled by fan heat and duct losses, and heat added to air lost through supply duct leak.
Qout	Net heat taken from outdoor air. Sum of sensible and latent, measured RELATIVE TO CURRENT RETURN AIR CONDITIONS.
Qfan	Heat added to air stream by supply fan, plus return fan if any – but not relief fan..
Qloss	Heat added to air stream by supply and return duct leaks and conductive loss. Computed in each case as the sensible and latent heat in the air stream relative to return air conditions after the leak or loss, less the same value before the leak or loss.
Qload	Net energy delivered to the terminals – Sensible and latent energy, measured relative to return air conditions. INCLUDES DUCT LOSSES after terminals; thus will differ from sum of zone qMech's + qMecLat's.

Qbal	Sum of all the 'Q' columns, primarily a development aid. Zero indicates consistent and accurate computation; the normal printout is something like .0000, indicating that the value was too small to print in the space alloted, but not precisely zero, due to computational tolerances and internal round-off errors.

6 Probe Definitions

6.1 @ahRes[1..].

Name	Input?	Runtime?	Type	Variability	Description
name	–	X	string	constant	–
Y.n	–	X	unrecognized	end of run (of each phase, autoSize or simulate)	–
Y.tDbO	–	X	number	end of run (of each phase, autoSize or simulate)	–
Y.wO	–	X	number	end of run (of each phase, autoSize or simulate)	–
Y.tr	–	X	number	end of run (of each phase, autoSize or simulate)	–
Y.wr	–	X	number	end of run (of each phase, autoSize or simulate)	–
Y.tmix	–	X	number	end of run (of each phase, autoSize or simulate)	–
Y.wmix	–	X	number	end of run (of each phase, autoSize or simulate)	–
Y.ts	–	X	number	end of run (of each phase, autoSize or simulate)	–
Y.ws	–	X	number	end of run (of each phase, autoSize or simulate)	–
Y.po	–	X	number	end of run (of each phase, autoSize or simulate)	–
Y.frFanOn	–	X	number	end of run (of each phase, autoSize or simulate)	–
Y.vf	–	X	number	end of run (of each phase, autoSize or simulate)	–
Y.qh	–	X	number	end of run (of each phase, autoSize or simulate)	–

Name	Input?	Runtime?	Type	Variability	Description
Y.qc	–	X	number	end of run (of each phase, autoSize or simulate)	–
Y.qs	–	X	number	end of run (of each phase, autoSize or simulate)	–
Y.ql	–	X	number	end of run (of each phase, autoSize or simulate)	–
Y.qO	–	X	number	end of run (of each phase, autoSize or simulate)	–
Y.qFan	–	X	number	end of run (of each phase, autoSize or simulate)	–
Y.qLoss	–	X	number	end of run (of each phase, autoSize or simulate)	–
Y.qLoad	–	X	number	end of run (of each phase, autoSize or simulate)	–
Y.qBal	–	X	number	end of run (of each phase, autoSize or simulate)	–
Y.ph	–	X	number	end of run (of each phase, autoSize or simulate)	–
Y.pc	–	X	number	end of run (of each phase, autoSize or simulate)	–
Y.pAuxH	–	X	number	end of run (of each phase, autoSize or simulate)	–
Y.pAuxC	–	X	number	end of run (of each phase, autoSize or simulate)	–
Y.pFan	–	X	number	end of run (of each phase, autoSize or simulate)	–
Y.hrsOn	–	X	number	end of run (of each phase, autoSize or simulate)	–
Y.nSubhr	–	X	number	end of run (of each phase, autoSize or simulate)	–
Y.nIter1	–	X	number	end of run (of each phase, autoSize or simulate)	–
Y.nIter2	–	X	number	end of run (of each phase, autoSize or simulate)	–

Name	Input?	Runtime?	Type	Variability	Description
Y.nIter4	–	X	number	end of run (of each phase, autoSize or simulate)	–
Y.nIterFan	–	X	number	end of run (of each phase, autoSize or simulate)	–
M.n	–	X	unrecognized	end of each month	–
M.tDbO	–	X	number	end of each month	–
M.wO	–	X	number	end of each month	–
M.tr	–	X	number	end of each month	–
M.wr	–	X	number	end of each month	–
M.tmix	–	X	number	end of each month	–
M.wmix	–	X	number	end of each month	–
M.ts	–	X	number	end of each month	–
M.ws	–	X	number	end of each month	–
M.po	–	X	number	end of each month	–
M.frFanOn	–	X	number	end of each month	–
M.vf	–	X	number	end of each month	–
M.qh	–	X	number	end of each month	–
M.qc	–	X	number	end of each month	–
M.qs	–	X	number	end of each month	–
M.ql	–	X	number	end of each month	–
M.qO	–	X	number	end of each month	–
M.qFan	–	X	number	end of each month	–
M.qLoss	–	X	number	end of each month	–
M.qLoad	–	X	number	end of each month	–
M.qBal	–	X	number	end of each month	–
M.ph	–	X	number	end of each month	–
M.pc	–	X	number	end of each month	–
M.pAuxH	–	X	number	end of each month	–
M.pAuxC	–	X	number	end of each month	–
M.pFan	–	X	number	end of each month	–
M.hrsOn	–	X	number	end of each month	–
M.nSubhr	–	X	number	end of each month	–
M.nIter1	–	X	number	end of each month	–
M.nIter2	–	X	number	end of each month	–
M.nIter4	–	X	number	end of each month	–
M.nIterFan	–	X	number	end of each month	–
D.n	–	X	unrecognized	end of each day	–
D.tDbO	–	X	number	end of each day	–
D.wO	–	X	number	end of each day	–
D.tr	–	X	number	end of each day	–
D.wr	–	X	number	end of each day	–
D.tmix	–	X	number	end of each day	–
D.wmix	–	X	number	end of each day	–
D.ts	–	X	number	end of each day	–
D.ws	–	X	number	end of each day	–
D.po	–	X	number	end of each day	–
D.frFanOn	–	X	number	end of each day	–
D.vf	–	X	number	end of each day	–
D.qh	–	X	number	end of each day	–
D.qc	–	X	number	end of each day	–

Name	Input?	Runtime?	Type	Variability	Description
D.qs	–	X	number	end of each day	–
D.ql	–	X	number	end of each day	–
D.qO	–	X	number	end of each day	–
D.qFan	–	X	number	end of each day	–
D.qLoss	–	X	number	end of each day	–
D.qLoad	–	X	number	end of each day	–
D.qBal	–	X	number	end of each day	–
D.ph	–	X	number	end of each day	–
D.pc	–	X	number	end of each day	–
D.pAuxH	–	X	number	end of each day	–
D.pAuxC	–	X	number	end of each day	–
D.pFan	–	X	number	end of each day	–
D.hrsOn	–	X	number	end of each day	–
D.nSubhr	–	X	number	end of each day	–
D.nIter1	–	X	number	end of each day	–
D.nIter2	–	X	number	end of each day	–
D.nIter4	–	X	number	end of each day	–
D.nIterFan	–	X	number	end of each day	–
H.n	–	X	unrecognized	end of each hour	–
H.tDbO	–	X	number	end of each hour	–
H.wO	–	X	number	end of each hour	–
H.tr	–	X	number	end of each hour	–
H.wr	–	X	number	end of each hour	–
H.tmix	–	X	number	end of each hour	–
H.wmix	–	X	number	end of each hour	–
H.ts	–	X	number	end of each hour	–
H.ws	–	X	number	end of each hour	–
H.po	–	X	number	end of each hour	–
H.frFanOn	–	X	number	end of each hour	–
H.vf	–	X	number	end of each hour	–
H.qh	–	X	number	end of each hour	–
H.qc	–	X	number	end of each hour	–
H.qs	–	X	number	end of each hour	–
H.ql	–	X	number	end of each hour	–
H.qO	–	X	number	end of each hour	–
H.qFan	–	X	number	end of each hour	–
H.qLoss	–	X	number	end of each hour	–
H.qLoad	–	X	number	end of each hour	–
H.qBal	–	X	number	end of each hour	–
H.ph	–	X	number	end of each hour	–
H.pc	–	X	number	end of each hour	–
H.pAuxH	–	X	number	end of each hour	–
H.pAuxC	–	X	number	end of each hour	–
H.pFan	–	X	number	end of each hour	–
H.hrsOn	–	X	number	end of each hour	–
H.nSubhr	–	X	number	end of each hour	–
H.nIter1	–	X	number	end of each hour	–
H.nIter2	–	X	number	end of each hour	–
H.nIter4	–	X	number	end of each hour	–
H.nIterFan	–	X	number	end of each hour	–
S.n	–	X	unrecognized	end of each subhour	–
S.tDbO	–	X	number	end of each subhour	–

Name	Input?	Runtime?	Type	Variability	Description
S.wO	–	X	number	end of each subhour	–
S.tr	–	X	number	end of each subhour	–
S.wr	–	X	number	end of each subhour	–
S.tmix	–	X	number	end of each subhour	–
S.wmix	–	X	number	end of each subhour	–
S.ts	–	X	number	end of each subhour	–
S.ws	–	X	number	end of each subhour	–
S.po	–	X	number	end of each subhour	–
S.frFanOn	–	X	number	end of each subhour	–
S.vf	–	X	number	end of each subhour	–
S.qh	–	X	number	end of each subhour	–
S.qc	–	X	number	end of each subhour	–
S.qs	–	X	number	end of each subhour	–
S.ql	–	X	number	end of each subhour	–
S.qO	–	X	number	end of each subhour	–
S.qFan	–	X	number	end of each subhour	–
S.qLoss	–	X	number	end of each subhour	–
S.qLoad	–	X	number	end of each subhour	–
S.qBal	–	X	number	end of each subhour	–
S.ph	–	X	number	end of each subhour	–
S.pc	–	X	number	end of each subhour	–
S.pAuxH	–	X	number	end of each subhour	–
S.pAuxC	–	X	number	end of each subhour	–
S.pFan	–	X	number	end of each subhour	–
S.hrsOn	–	X	number	end of each subhour	–
S.nSubhr	–	X	number	end of each subhour	–
S.nIter1	–	X	number	end of each subhour	–
S.nIter2	–	X	number	end of each subhour	–
S.nIter4	–	X	number	end of each subhour	–
S.nIterFan	–	X	number	end of each subhour	–

6.2 @airHandler[1..].

Name	Input?	Runtime?	Type	Variability	Description
name	X	X	string	constant	–
ahTsDsH	X	X	number	hourly	–
ahTsDsC	X	X	number	hourly	–
ahccSHR	X	X	number	autosize and simulate phase start time	–
coilOversize	X	X	number	autosize and simulate phase start time	–
fanOversize	X	X	number	autosize and simulate phase start time	–
asRfan	X	X	integer number	run start time (of each phase, autoSize or simulate)	–

Name	Input?	Runtime?	Type	Variability	Description
asFlow	X	X	integer number	run start time (of each phase, autoSize or simulate)	–
hcAs.az_active	X	X	integer number	run start time (of each phase, autoSize or simulate)	–
hcAs.az_a	X	X	number	end of each subhour	–
hcAs.az_b	X	X	number	end of each subhour	–
hcAs.ldPk	X	X	number	end of each subhour	–
hcAs.ldPkAs	X	X	number	end of each day	–
hcAs.ldPkAs1	X	X	number	end of each day	–
hcAs.plrPk	X	X	number	end of each subhour	–
hcAs.plrPkAs	X	X	number	end of each day	–
hcAs.xPk	X	X	number	end of each subhour	–
hcAs.xPkAs	X	X	number	end of each day	–
ccAs.az_active	X	X	integer number	run start time (of each phase, autoSize or simulate)	–
ccAs.az_a	X	X	number	end of each subhour	–
ccAs.az_b	X	X	number	end of each subhour	–
ccAs.ldPk	X	X	number	end of each subhour	–
ccAs.ldPkAs	X	X	number	end of each day	–
ccAs.ldPkAs1	X	X	number	end of each day	–
ccAs.plrPk	X	X	number	end of each subhour	–
ccAs.plrPkAs	X	X	number	end of each day	–
ccAs.xPk	X	X	number	end of each subhour	–
ccAs.xPkAs	X	X	number	end of each day	–
fanAs.az_active	X	X	integer number	run start time (of each phase, autoSize or simulate)	–
fanAs.az_a	X	X	number	end of each subhour	–
fanAs.az_b	X	X	number	end of each subhour	–
fanAs.ldPk	X	X	number	end of each subhour	–
fanAs.ldPkAs	X	X	number	end of each day	–
fanAs.ldPkAs1	X	X	number	end of each day	–

Name	Input?	Runtime?	Type	Variability	Description
fanAs.plrPk	X	X	number	end of each subhour	–
fanAs.plrPkAs	X	X	number	end of each day	–
fanAs.xPk	X	X	number	end of each subhour	–
fanAs.xPkAs	X	X	number	end of each day	–
bVfDs	X	X	number	end of each subhour	–
qcPkS	X	X	number	end of each subhour	–
qcPkL	X	X	number	end of each subhour	–
qcPkH	X	X	integer number	end of each subhour	–
qcPkD	X	X	integer number	end of each subhour	–
qcPkM	X	X	integer number	end of each subhour	–
qcPkTDbO	X	X	number	end of each subhour	–
qcPkWO	X	X	number	end of each subhour	–
qcPkTen	X	X	number	end of each subhour	–
qcPkWen	X	X	number	end of each subhour	–
qcPkTex	X	X	number	end of each subhour	–
qcPkWex	X	X	number	end of each subhour	–
qcPkSAs	X	X	number	end of each subhour	–
qcPkLAs	X	X	number	end of each subhour	–
qcPkHAs	X	X	integer number	end of each subhour	–
qcPkDAs	X	X	integer number	end of each subhour	–
qcPkMAs	X	X	integer number	end of each subhour	–
qcPkTDbOAs	X	X	number	end of each subhour	–
qcPkWOAs	X	X	number	end of each subhour	–
qcPkTenAs	X	X	number	end of each subhour	–
qcPkWenAs	X	X	number	end of each subhour	–
qcPkTexAs	X	X	number	end of each subhour	–
qcPkWexAs	X	X	number	end of each subhour	–

Name	Input?	Runtime?	Type	Variability	Description
ahTsSp	X	X	unrecognized	hourly	–
ahFanCycles	X	X	unrecognized	hourly	–
ahTsMn	X	X	number	hourly	–
ahTsMx	X	X	number	hourly	–
ahTsRaMn	X	X	number	hourly	–
ahTsRaMx	X	X	number	hourly	–
ahCtu	X	X	integer number	run start time (of each phase, autoSize or simulate)	–
ahWzCzns[0]	X	X	integer number	autosize and simulate phase start time	–
ahWzCzns[1]	X	X	integer number	autosize and simulate phase start time	–
ahWzCzns[2]	X	X	integer number	autosize and simulate phase start time	–
ahWzCzns[3]	X	X	integer number	autosize and simulate phase start time	–
ahWzCzns[4]	X	X	integer number	autosize and simulate phase start time	–
ahWzCzns[5]	X	X	integer number	autosize and simulate phase start time	–
ahWzCzns[6]	X	X	integer number	autosize and simulate phase start time	–
ahWzCzns[7]	X	X	integer number	autosize and simulate phase start time	–
ahWzCzns[8]	X	X	integer number	autosize and simulate phase start time	–
ahWzCzns[9]	X	X	integer number	autosize and simulate phase start time	–
ahWzCzns[10]	X	X	integer number	autosize and simulate phase start time	–
ahWzCzns[11]	X	X	integer number	autosize and simulate phase start time	–
ahWzCzns[12]	X	X	integer number	autosize and simulate phase start time	–
ahWzCzns[13]	X	X	integer number	autosize and simulate phase start time	–

Name	Input?	Runtime?	Type	Variability	Description
ahWzCzns[14]	X	X	integer number	autosize and simulate phase start time	–
ahWzCzns[15]	X	X	integer number	autosize and simulate phase start time	–
ahCzCzns[0]	X	X	integer number	autosize and simulate phase start time	–
ahCzCzns[1]	X	X	integer number	autosize and simulate phase start time	–
ahCzCzns[2]	X	X	integer number	autosize and simulate phase start time	–
ahCzCzns[3]	X	X	integer number	autosize and simulate phase start time	–
ahCzCzns[4]	X	X	integer number	autosize and simulate phase start time	–
ahCzCzns[5]	X	X	integer number	autosize and simulate phase start time	–
ahCzCzns[6]	X	X	integer number	autosize and simulate phase start time	–
ahCzCzns[7]	X	X	integer number	autosize and simulate phase start time	–
ahCzCzns[8]	X	X	integer number	autosize and simulate phase start time	–
ahCzCzns[9]	X	X	integer number	autosize and simulate phase start time	–
ahCzCzns[10]	X	X	integer number	autosize and simulate phase start time	–
ahCzCzns[11]	X	X	integer number	autosize and simulate phase start time	–
ahCzCzns[12]	X	X	integer number	autosize and simulate phase start time	–
ahCzCzns[13]	X	X	integer number	autosize and simulate phase start time	–
ahCzCzns[14]	X	X	integer number	autosize and simulate phase start time	–

Name	Input?	Runtime?	Type	Variability	Description
ahCzCzns[15]	X	X	integer number	autosize and simulate phase start time	–
oaMnCm	X	X	unrecognized	autosize and simulate phase start time	–
oaMnFrac	X	X	number	hourly	–
oaVfDsMn	X	X	number	run start time (of each phase, autoSize or simulate)	–
oaEcoTy	X	X	unrecognized	autosize and simulate phase start time	–
oaLimT	X	X	unrecognized	hourly	–
oaLimE	X	X	unrecognized	hourly	–
oaOaLeak	X	X	number	autosize and simulate phase start time	–
oaRaLeak	X	X	number	autosize and simulate phase start time	–
ahSOLeak	X	X	number	autosize and simulate phase start time	–
ahROLeak	X	X	number	autosize and simulate phase start time	–
ahSOLoss	X	X	number	autosize and simulate phase start time	–
ahROLoss	X	X	number	autosize and simulate phase start time	–
ahSch	X	X	unrecognized	hourly	–
sfan.fanTy	X	X	unrecognized	autosize and simulate phase start time	–
sfan.vfDs	X	X	number	end of each subhour	–
sfan.vfDs__As	X	X	number	autosize and simulate phase start time	–
sfan.vfDs__AsNov	X	X	number	autosize and simulate phase start time	–
sfan.vfMxF	X	X	number	autosize and simulate phase start time	–

Name	Input?	Runtime?	Type	Variability	Description
sfan.press	X	X	number	run start time (of each phase, autoSize or simulate)	–
sfan.eff	X	X	number	run start time (of each phase, autoSize or simulate)	–
sfan.shaftPwr	X	X	number	run start time (of each phase, autoSize or simulate)	–
sfan.elecPwr	X	X	number	run start time (of each phase, autoSize or simulate)	–
sfan.motTy	X	X	unrecognized	run start time (of each phase, autoSize or simulate)	–
sfan.motEff	X	X	number	autosize and simulate phase start time	–
sfan.motPos	X	X	unrecognized	autosize and simulate phase start time	–
sfan.curvePy.k[0]	X	X	number	autosize and simulate phase start time	–
sfan.curvePy.k[1]	X	X	number	autosize and simulate phase start time	–
sfan.curvePy.k[2]	X	X	number	autosize and simulate phase start time	–
sfan.curvePy.k[3]	X	X	number	autosize and simulate phase start time	–
sfan.curvePy.k[4]	X	X	number	autosize and simulate phase start time	–
sfan.curvePy.k[5]	X	X	number	autosize and simulate phase start time	–
sfan.mtri	X	X	integer number	autosize and simulate phase start time	–
sfan.endUse	X	X	integer number	autosize and simulate phase start time	–

Name	Input?	Runtime?	Type	Variability	Description
sfan.ausz	X	X	integer number	run start time (of each phase, autoSize or simulate)	–
sfan.outPower	X	X	number	subhourly	–
sfan.airPower	X	X	number	subhourly	–
sfan.cMx	X	X	number	end of each subhour	–
sfan.c	X	X	number	end of each subhour	–
sfan.t	X	X	number	end of each subhour	–
sfan.frOn	X	X	number	end of each subhour	–
sfan.p	X	X	number	end of each subhour	–
sfan.q	X	X	number	end of each subhour	–
sfan.dT	X	X	number	end of each subhour	–
sfan.qAround	X	X	number	end of each subhour	–
rfan.fanTy	X	X	unrecognized	autosize and simulate phase start time	–
rfan.vfDs	X	X	number	end of each subhour	–
rfan.vfDs_As	X	X	number	autosize and simulate phase start time	–
rfan.vfDs_AsNov	X	X	number	autosize and simulate phase start time	–
rfan.vfMxF	X	X	number	autosize and simulate phase start time	–
rfan.press	X	X	number	run start time (of each phase, autoSize or simulate)	–
rfan.eff	X	X	number	run start time (of each phase, autoSize or simulate)	–
rfan.shaftPwr	X	X	number	run start time (of each phase, autoSize or simulate)	–
rfan.elecPwr	X	X	number	run start time (of each phase, autoSize or simulate)	–

Name	Input?	Runtime?	Type	Variability	Description
rfan.motTy	X	X	unrecognized	run start time (of each phase, autoSize or simulate)	–
rfan.motEff	X	X	number	autosize and simulate phase start time	–
rfan.motPos	X	X	unrecognized	autosize and simulate phase start time	–
rfan.curvePy.k[0]	X	X	number	autosize and simulate phase start time	–
rfan.curvePy.k[1]	X	X	number	autosize and simulate phase start time	–
rfan.curvePy.k[2]	X	X	number	autosize and simulate phase start time	–
rfan.curvePy.k[3]	X	X	number	autosize and simulate phase start time	–
rfan.curvePy.k[4]	X	X	number	autosize and simulate phase start time	–
rfan.curvePy.k[5]	X	X	number	autosize and simulate phase start time	–
rfan.mtri	X	X	integer number	autosize and simulate phase start time	–
rfan.endUse	X	X	integer number	autosize and simulate phase start time	–
rfan.ausz	X	X	integer number	run start time (of each phase, autoSize or simulate)	–
rfan.outPower	X	X	number	subhourly	–
rfan.airPower	X	X	number	subhourly	–
rfan.cMx	X	X	number	end of each subhour	–
rfan.c	X	X	number	end of each subhour	–
rfan.t	X	X	number	end of each subhour	–
rfan.frOn	X	X	number	end of each subhour	–
rfan.p	X	X	number	end of each subhour	–
rfan.q	X	X	number	end of each subhour	–

Name	Input?	Runtime?	Type	Variability	Description
rfan.dT	X	X	number	end of each subhour	–
rfan.qAround	X	X	number	end of each subhour	–
cch.cchCM	X	X	unrecognized	run start time (of each phase, autoSize or simulate)	–
cch.pMx	X	X	number	autosize and simulate phase start time	–
cch.pMn	X	X	number	autosize and simulate phase start time	–
cch.tMx	X	X	number	autosize and simulate phase start time	–
cch.tMn	X	X	number	autosize and simulate phase start time	–
cch.dt	X	X	number	autosize and simulate phase start time	–
cch.tOn	X	X	number	autosize and simulate phase start time	–
cch.tOff	X	X	number	run start time (of each phase, autoSize or simulate)	–
cch.mtri	X	X	integer number	autosize and simulate phase start time	–
cch.p47Off	X	X	number	run start time (of each phase, autoSize or simulate)	–
cch.p17	X	X	number	run start time (of each phase, autoSize or simulate)	–
cch.p47	X	X	number	run start time (of each phase, autoSize or simulate)	–
cch.frCprOn	X	X	number	end of each subhour	–
cch.tState	X	X	integer number	end of each subhour	–
cch.p	X	X	number	end of each subhour	–

Name	Input?	Runtime?	Type	Variability	Description
ahhc.coilTy	X	X	unrecognized	run start time (of each phase, autoSize or simulate)	–
ahhc.sched	X	X	unrecognized	hourly	–
ahhc.captRat	X	X	number	end of each subhour	–
ahhc.captRat_As	X	X	number	autosize and simulate phase start time	–
ahhc.captRat_AsNov	X	X	number	autosize and simulate phase start time	–
ahhc.bCaptRat	X	X	number	end of each subhour	–
ahhc.eirRat	X	X	number	hourly	–
ahhc.mtri	X	X	integer number	autosize and simulate phase start time	–
ahhc.auxOn	X	X	number	hourly	–
ahhc.auxOnMtri	X	X	integer number	autosize and simulate phase start time	–
ahhc.auxOff	X	X	number	hourly	–
ahhc.auxOffMtri	X	X	integer number	autosize and simulate phase start time	–
ahhc.auxOnAtall	X	X	number	hourly	–
ahhc.auxOnAtallMtri	X	X	integer number	autosize and simulate phase start time	–
ahhc.auxFullOff	X	X	number	hourly	–
ahhc.auxFullOffMtri	X	X	integer number	autosize and simulate phase start time	–
ahhc.q	X	X	number	end of each subhour	–
ahhc.qPr	X	X	number	end of each subhour	–
ahhc.p	X	X	number	end of each subhour	–
ahhc.plr	X	X	number	end of each subhour	–
ahhc.plrAv	X	X	number	end of each subhour	–
ahhc.eir	X	X	number	end of each subhour	–
ahhc.pAuxOn	X	X	number	end of each subhour	–
ahhc.pAuxOff	X	X	number	end of each subhour	–

Name	Input?	Runtime?	Type	Variability	Description
ahhc.pAuxOnAtall	X	X	number	end of each subhour	–
ahhc.pAuxFullOff	X	X	number	end of each subhour	–
ahhc.effRat	X	X	number	autosize and simulate phase start time	–
ahhc.pyEi.k[0]	X	X	number	autosize and simulate phase start time	–
ahhc.pyEi.k[1]	X	X	number	autosize and simulate phase start time	–
ahhc.pyEi.k[2]	X	X	number	autosize and simulate phase start time	–
ahhc.pyEi.k[3]	X	X	number	autosize and simulate phase start time	–
ahhc.pyEi.k[4]	X	X	number	autosize and simulate phase start time	–
ahhc.stackEffect	X	X	number	hourly	–
ahhc.hpi	X	X	integer number	autosize and simulate phase start time	–
ahhc.nxTu4hp	X	X	integer number	run start time (of each phase, autoSize or simulate)	–
ahhc.nxAh4hp	X	X	integer number	run start time (of each phase, autoSize or simulate)	–
ahhc.flueLoss	X	X	number	end of each subhour	–
ahhc.qWant	X	X	number	end of each subhour	–
ahhc.cap17	X	X	number	autosize and simulate phase start time	–
ahhc.cap47	X	X	number	autosize and simulate phase start time	–
ahhc.cap35	X	X	number	run start time (of each phase, autoSize or simulate)	–
ahhc.fd35Df	X	X	number	autosize and simulate phase start time	–

Name	Input?	Runtime?	Type	Variability	Description
ahhc.capIa	X	X	number	autosize and simulate phase start time	–
ahhc.supRh	X	X	number	autosize and simulate phase start time	–
ahhc.tFrMn	X	X	number	autosize and simulate phase start time	–
ahhc.tFrMx	X	X	number	autosize and simulate phase start time	–
ahhc.tFrPk	X	X	number	autosize and simulate phase start time	–
ahhc.dfrFMn	X	X	number	autosize and simulate phase start time	–
ahhc.dfrFMx	X	X	number	autosize and simulate phase start time	–
ahhc.dfrCap	X	X	number	run start time (of each phase, autoSize or simulate)	–
ahhc.dfrRh	X	X	number	autosize and simulate phase start time	–
ahhc.tOff	X	X	number	autosize and simulate phase start time	–
ahhc.tOn	X	X	number	autosize and simulate phase start time	–
ahhc.in17	X	X	number	autosize and simulate phase start time	–
ahhc.in47	X	X	number	autosize and simulate phase start time	–
ahhc.inIa	X	X	number	autosize and simulate phase start time	–
ahhc.cd	X	X	number	autosize and simulate phase start time	–
ahhc.in17c	X	X	number	run start time (of each phase, autoSize or simulate)	–

Name	Input?	Runtime?	Type	Variability	Description
ahhc.in47c	X	X	number	run start time (of each phase, autoSize or simulate)	–
ahhc.cdm	X	X	number	run start time (of each phase, autoSize or simulate)	–
ahhc.tIa	X	X	number	end of each subhour	–
ahhc.qSupLim	X	X	number	end of each subhour	–
ahhc.frFanOn	X	X	number	end of each subhour	–
ahhc.loTLockout	X	X	integer number	end of each subhour	–
ahhc.supOn	X	X	integer number	end of each subhour	–
ahhc.capCon	X	X	number	end of each subhour	–
ahhc.pDfrhCon	X	X	number	end of each subhour	–
ahhc.cap	X	X	number	end of each subhour	–
ahhc.frCprOn	X	X	number	end of each subhour	–
ahhc.pCpr	X	X	number	end of each subhour	–
ahhc.pRh	X	X	number	end of each subhour	–
ahccBypass	X	X	number	autosize and simulate phase start time	–
ahcc.coilTy	X	X	unrecognized	run start time (of each phase, autoSize or simulate)	–
ahcc.sched	X	X	unrecognized	hourly	–
ahcc.captRat	X	X	number	end of each subhour	–
ahcc.captRat__As	X	X	number	autosize and simulate phase start time	–
ahcc.captRat__AsNov	X	X	number	autosize and simulate phase start time	–
ahcc.bCaptRat	X	X	number	end of each subhour	–
ahcc.eirRat	X	X	number	hourly	–
ahcc.mtri	X	X	integer number	autosize and simulate phase start time	–

Name	Input?	Runtime?	Type	Variability	Description
ahcc.auxOn	X	X	number	hourly	–
ahcc.auxOnMtri	X	X	integer number	autosize and simulate phase start time	–
ahcc.auxOff	X	X	number	hourly	–
ahcc.auxOffMtri	X	X	integer number	autosize and simulate phase start time	–
ahcc.auxOnAtall	X	X	number	hourly	–
ahcc.auxOnAtallMtri	X	X	integer number	autosize and simulate phase start time	–
ahcc.auxFullOff	X	X	number	hourly	–
ahcc.auxFullOffMtri	X	X	integer number	autosize and simulate phase start time	–
ahcc.q	X	X	number	end of each subhour	–
ahcc.qPr	X	X	number	end of each subhour	–
ahcc.p	X	X	number	end of each subhour	–
ahcc.plr	X	X	number	end of each subhour	–
ahcc.plrAv	X	X	number	end of each subhour	–
ahcc.eir	X	X	number	end of each subhour	–
ahcc.pAuxOn	X	X	number	end of each subhour	–
ahcc.pAuxOff	X	X	number	end of each subhour	–
ahcc.pAuxOnAtall	X	X	number	end of each subhour	–
ahcc.pAuxFullOff	X	X	number	end of each subhour	–
ahcc.capsRat	X	X	number	end of each subhour	–
ahcc.capsRat__As	X	X	number	autosize and simulate phase start time	–
ahcc.capsRat__AsNov	X	X	number	autosize and simulate phase start time	–
ahcc.minTEvap	X	X	number	autosize and simulate phase start time	–
ahcc.k1	X	X	number	autosize and simulate phase start time	–

Name	Input?	Runtime?	Type	Variability	Description
ahcc.dsTDbCnd	X	X	number	autosize and simulate phase start time	–
ahcc.dsTDbEn	X	X	number	autosize and simulate phase start time	–
ahcc.dsTWbEn	X	X	number	autosize and simulate phase start time	–
ahcc.vfR	X	X	number	run start time (of each phase, autoSize or simulate)	–
ahcc.vfRperTon	X	X	number	run start time (of each phase, autoSize or simulate)	–
ahcc.minUnldPlr	X	X	number	autosize and simulate phase start time	–
ahcc.minFsldPlr	X	X	number	autosize and simulate phase start time	–
ahcc.pydxCaptT.k[0]	X	X	number	autosize and simulate phase start time	–
ahcc.pydxCaptT.k[1]	X	X	number	autosize and simulate phase start time	–
ahcc.pydxCaptT.k[2]	X	X	number	autosize and simulate phase start time	–
ahcc.pydxCaptT.k[3]	X	X	number	autosize and simulate phase start time	–
ahcc.pydxCaptT.k[4]	X	X	number	autosize and simulate phase start time	–
ahcc.pydxCaptT.k[5]	X	X	number	autosize and simulate phase start time	–
ahcc.pydxCaptT.k[6]	X	X	number	autosize and simulate phase start time	–
ahcc.pydxCaptF.k[0]	X	X	number	autosize and simulate phase start time	–
ahcc.pydxCaptF.k[1]	X	X	number	autosize and simulate phase start time	–

Name	Input?	Runtime?	Type	Variability	Description
ahcc.pydxCaptF.k[2]	X	X	number	autosize and simulate phase start time	–
ahcc.pydxCaptF.k[3]	X	X	number	autosize and simulate phase start time	–
ahcc.pydxCaptF.k[4]	X	X	number	autosize and simulate phase start time	–
ahcc.pydxCaptFLim	X	X	number	autosize and simulate phase start time	–
ahcc.pydxEirT.k[0]	X	X	number	autosize and simulate phase start time	–
ahcc.pydxEirT.k[1]	X	X	number	autosize and simulate phase start time	–
ahcc.pydxEirT.k[2]	X	X	number	autosize and simulate phase start time	–
ahcc.pydxEirT.k[3]	X	X	number	autosize and simulate phase start time	–
ahcc.pydxEirT.k[4]	X	X	number	autosize and simulate phase start time	–
ahcc.pydxEirT.k[5]	X	X	number	autosize and simulate phase start time	–
ahcc.pydxEirT.k[6]	X	X	number	autosize and simulate phase start time	–
ahcc.pydxEirUl.k[0]	X	X	number	autosize and simulate phase start time	–
ahcc.pydxEirUl.k[1]	X	X	number	autosize and simulate phase start time	–
ahcc.pydxEirUl.k[2]	X	X	number	autosize and simulate phase start time	–
ahcc.pydxEirUl.k[3]	X	X	number	autosize and simulate phase start time	–
ahcc.pydxEirUl.k[4]	X	X	number	autosize and simulate phase start time	–
ahcc.cpi	X	X	integer number	autosize and simulate phase start time	–

Name	Input?	Runtime?	Type	Variability	Description
ahcc.gpmDs	X	X	number	autosize and simulate phase start time	–
ahcc.ntuoDs	X	X	number	autosize and simulate phase start time	–
ahcc.ntuiDs	X	X	number	autosize and simulate phase start time	–
ahcc.wsatMinTEvap	X	X	number	run start time (of each phase, autoSize or simulate)	–
ahcc.hsatMinTEvap	X	X	number	run start time (of each phase, autoSize or simulate)	–
ahcc.efecOR	X	X	number	run start time (of each phase, autoSize or simulate)	–
ahcc.ntuR	X	X	number	run start time (of each phase, autoSize or simulate)	–
ahcc.eirMinUnldPlr	X	X	number	run start time (of each phase, autoSize or simulate)	–
ahcc.menR	X	X	number	run start time (of each phase, autoSize or simulate)	–
ahcc.nxAh4cp	X	X	integer number	run start time (of each phase, autoSize or simulate)	–
ahcc.mwDs	X	X	number	run start time (of each phase, autoSize or simulate)	–
ahcc.wantQflag	X	X	integer number	end of each subhour	–
ahcc.tewd	X	X	number	end of each subhour	–
ahcc.chwQ	X	X	number	end of each subhour	–
ahcc.tr	X	X	number	end of each subhour	–
ahcc.cpTsPr	X	X	number	end of each subhour	–

Name	Input?	Runtime?	Type	Variability	Description
ahcc.trPr	X	X	number	end of each subhour	–
ahcc.fullLoadWet	X	X	integer number	end of each subhour	–
ahcc.frCprOn	X	X	number	end of each subhour	–
ahcc.tWbEn	X	X	number	end of each subhour	–
ahcc.hen	X	X	number	end of each subhour	–
ahcc.tDbCnd	X	X	number	end of each subhour	–
ahcc.efecO	X	X	number	end of each subhour	–
ahcc.capt	X	X	number	end of each subhour	–
ahcc.caps	X	X	number	end of each subhour	–
ahcc.plrVf	X	X	number	end of each subhour	–
ahcc.plrSens	X	X	number	end of each subhour	–
ahcc.qs	X	X	number	end of each subhour	–
ahcc.ql	X	X	number	end of each subhour	–
ahcc.xLGain	X	X	number	end of each subhour	–
ahcc.xLGainYr	X	X	number	end of each subhour	–
ahcc.nSubhrsLX	X	X	number	end of each subhour	–
ahcc.minTLtd	X	X	integer number	end of each subhour	–
ahcc.cfm2Few	X	X	integer number	end of each subhour	–
tu1	X	X	integer number	run start time (of each phase, autoSize or simulate)	–
zhx1	X	X	integer number	run start time (of each phase, autoSize or simulate)	–
ahMode	X	X	unrecognized	end of each subhour	–
ts	X	X	number	end of each subhour	–
ws	X	X	number	end of each subhour	–
wsls	X	X	number	subhourly	–

Name	Input?	Runtime?	Type	Variability	Description
airxTs	X	X	number	end of each subhour	–
tsMnFo	X	X	number	end of each subhour	–
tsMnFoOk	X	X	integer number	end of each subhour	–
tsMxFo	X	X	number	end of each subhour	–
tsMxFoOk	X	X	integer number	end of each subhour	–
tr	X	X	number	end of each subhour	–
wr	X	X	number	end of each subhour	–
cr	X	X	number	end of each subhour	–
cMxfcc	X	X	number	end of each subhour	–
frFanOn	X	X	number	end of each subhour	–
leakCOn	X	X	number	end of each subhour	–
tr1	X	X	number	end of each subhour	–
wr1	X	X	number	end of each subhour	–
cr1	X	X	number	end of each subhour	–
tr2	X	X	number	end of each subhour	–
rfanQ	X	X	number	end of each subhour	–
tmix	X	X	number	end of each subhour	–
wen	X	X	number	end of each subhour	–
cmix	X	X	number	end of each subhour	–
dtMixEn	X	X	number	end of each subhour	–
ten	X	X	number	end of each subhour	–
cen	X	X	number	end of each subhour	–
men	X	X	number	end of each subhour	–
tex	X	X	number	end of each subhour	–
wex	X	X	number	end of each subhour	–
tex1	X	X	number	end of each subhour	–

Name	Input?	Runtime?	Type	Variability	Description
dtExSen	X	X	number	end of each subhour	–
tSen	X	X	number	end of each subhour	–
dtSenS	X	X	number	end of each subhour	–
aTs	X	X	number	end of each subhour	–
aWs	X	X	number	end of each subhour	–
trNx	X	X	number	end of each subhour	–
wrNx	X	X	number	end of each subhour	–
crNx	X	X	number	end of each subhour	–
cMxnx	X	X	number	end of each subhour	–
frFanOnNx	X	X	number	end of each subhour	–
leakCOnNx	X	X	number	end of each subhour	–
tr1Nx	X	X	number	end of each subhour	–
wr1Nx	X	X	number	end of each subhour	–
cr1Nx	X	X	number	end of each subhour	–
tr2Nx	X	X	number	end of each subhour	–
uUseAr	X	X	unrecognized	end of each subhour	–
fcc	X	X	integer number	end of each hour	–
isZNorZN2	X	X	integer number	end of each hour	–
tsSp1	X	X	number	end of each subhour	–
tsFullFlow	X	X	number	end of each subhour	–
ecoEnabled	X	X	integer number	end of each subhour	–
coilLockout	X	X	integer number	end of each subhour	–
po	X	X	number	end of each subhour	–
coilUsed	X	X	unrecognized	end of each subhour	–
fanF	X	X	number	end of each subhour	–
fanFMax	X	X	number	end of each subhour	–

Name	Input?	Runtime?	Type	Variability	Description
fanLimited	X	X	integer number	end of each subhour	–
coilLimited	X	X	integer number	end of each subhour	–
tPossH	X	X	number	end of each subhour	–
tPossC	X	X	number	end of each subhour	–
ahClf	X	X	integer number	end of each subhour	–
ahPtf	X	X	integer number	end of each subhour	–
ahPtf2	X	X	integer number	end of each subhour	–

6.3 @boiler[1..]. (owner: heatPlant)

Name	Input?	Runtime?	Type	Variability	Description
name	X	X	string	constant	–
blrCap	X	X	number	autosize and simulate phase start time	capacity (Btuh). required input.
blrEffR	X	X	number	autosize and simulate phase start time	efficiency at steady-state full load, default .80.
blrEirR	X	X	number	autosize and simulate phase start time	Energy Input Ratio (1/eff): alternate input; used internally.
blrPyEi.k[0]	X	X	number	autosize and simulate phase start time	–
blrPyEi.k[1]	X	X	number	autosize and simulate phase start time	–
blrPyEi.k[2]	X	X	number	autosize and simulate phase start time	–
blrPyEi.k[3]	X	X	number	autosize and simulate phase start time	–
blrPyEi.k[4]	X	X	number	autosize and simulate phase start time	–
mtri	X	X	integer number	input time	subscript of MTR to which to charge boiler input power, default none

Name	Input?	Runtime?	Type	Variability	Description
blrp.gpm	X	X	number	run start time (of each phase, autoSize or simulate)	–
blrp.hdLoss	X	X	number	autosize and simulate phase start time	–
blrp.motEff	X	X	number	autosize and simulate phase start time	–
blrp.hydEff	X	X	number	autosize and simulate phase start time	–
blrp.ovrunF	X	X	number	run start time (of each phase, autoSize or simulate)	–
blrp.mtri	X	X	integer number	autosize and simulate phase start time	–
blrp.mw	X	X	number	run start time (of each phase, autoSize or simulate)	–
blrp.q	X	X	number	run start time (of each phase, autoSize or simulate)	–
blrp.p	X	X	number	run start time (of each phase, autoSize or simulate)	–
auxOn	X	X	number	hourly	addl input energy used in proportion to plr when on, default 0, hourly vbl for future flexblty.
auxOnMtri	X	X	integer number	input time	MTR to which to charge "auxOn"
auxOff	X	X	number	hourly	addl input energy when off for part or all of subhr (proportional to 1-plr), for unforseen uses.
auxOffMtri	X	X	integer number	input time	MTR for "auxOff"
auxOnAtall	X	X	number	hourly	addl input energy used in toto when blr on for any part of subhour, for unforseen uses.

Name	Input?	Runtime?	Type	Variability	Description
auxOnAtallMtri	X	X	integer number	input time	MTR for "auzOnAtall"
auxFullOff	X	X	number	hourly	additional input energy when off FOR ENTIRE SUBHOUR (as opposed to in proportion to 1-plr).
auxFullOffMtri	X	X	integer number	input time	MTR to which auxFullOff is charged, default c.mtri.
nxBlr4hp	X	X	integer number	run start time (of each phase, autoSize or simulate)	0 or subscript of next boiler for same heatplant. 1st is HEATPLANT.blr1.
used	X	X	integer number	run start time (of each phase, autoSize or simulate)	during input checking (cncult6.cpp), TRUE if a stage uses this boiler
blrMode	X	X	unrecognized	end of each subhour	mode this subhour: off or on. Can be on with 0 q if in HEATPLANT's 1st stage.
plr	X	X	number	end of each subhour	part load ratio
q	X	X	number	end of each subhour	current output power level (excluding pump heat), share of total of connected coils & hx's
p	X	X	number	end of each subhour	current input power
pAuxOn	X	X	number	end of each subhour	blr-on proporotinal aux power this subhour
pAuxOff	X	X	number	end of each subhour	blr-off proportional aux power this subhour
pAuxOnAtall	X	X	number	end of each subhour	blr on-at-all aux power this subhour
pAuxFullOff	X	X	number	end of each subhour	auxFullOff power this subhour

6.4 @chiller[1..]. (owner: coolPlant)

Name	Input?	Runtime?	Type	Variability	Description
name	X	X	string	constant	–
chCapDs	X	X	number	autosize and simulate phase start time	capacity at chDsTs,chDsTcnd, Btuh. Required. Negative internally. Niles capDsn.
chTsDs	X	X	number	autosize and simulate phase start time	temp leaving chiller at which chCapDs applies, default 44. Niles twSuDsn.
chTcndDs	X	X	number	autosize and simulate phase start time	temp entering condenser (twoDel value) for chCapDs, default 85. Niles twCndDsn.
chPyCapT.k[0]	X	X	number	run start time (of each phase, autoSize or simulate)	–
chPyCapT.k[1]	X	X	number	run start time (of each phase, autoSize or simulate)	–
chPyCapT.k[2]	X	X	number	run start time (of each phase, autoSize or simulate)	–
chPyCapT.k[3]	X	X	number	run start time (of each phase, autoSize or simulate)	–
chPyCapT.k[4]	X	X	number	run start time (of each phase, autoSize or simulate)	–
chPyCapT.k[5]	X	X	number	run start time (of each phase, autoSize or simulate)	–
chPyCapT.k[6]	X	X	number	run start time (of each phase, autoSize or simulate)	–

Name	Input?	Runtime?	Type	Variability	Description
chCop	X	X	number	autosize and simulate phase start time	–
chEirDs	X	X	number	run start time (of each phase, autoSize or simulate)	–
chPyEirT.k[0]	X	X	number	run start time (of each phase, autoSize or simulate)	–
chPyEirT.k[1]	X	X	number	run start time (of each phase, autoSize or simulate)	–
chPyEirT.k[2]	X	X	number	run start time (of each phase, autoSize or simulate)	–
chPyEirT.k[3]	X	X	number	run start time (of each phase, autoSize or simulate)	–
chPyEirT.k[4]	X	X	number	run start time (of each phase, autoSize or simulate)	–
chPyEirT.k[5]	X	X	number	run start time (of each phase, autoSize or simulate)	–
chPyEirT.k[6]	X	X	number	run start time (of each phase, autoSize or simulate)	–
chPyEirUl.k[0]	X	X	number	run start time (of each phase, autoSize or simulate)	–

Name	Input?	Runtime?	Type	Variability	Description
chPyEirUl.k[1]	X	X	number	run start time (of each phase, autoSize or simulate)	–
chPyEirUl.k[2]	X	X	number	run start time (of each phase, autoSize or simulate)	–
chPyEirUl.k[3]	X	X	number	run start time (of each phase, autoSize or simulate)	–
chPyEirUl.k[4]	X	X	number	run start time (of each phase, autoSize or simulate)	–
chMinUnldPlr	X	X	number	autosize and simulate phase start time	min unloading loading part load ratio, default 0.1. Niles minUnLdPlr.
chMinFsldPlr	X	X	number	autosize and simulate phase start time	min false loading part load ratio, default 0.1. Niles minFsLdPlr. must be <= chMinUnldPlr.
chMotEff	X	X	number	autosize and simulate phase start time	–
mtri	X	X	integer number	input time	–
chpp.gpm	X	X	number	run start time (of each phase, autoSize or simulate)	–
chpp.hdLoss	X	X	number	autosize and simulate phase start time	–
chpp.motEff	X	X	number	autosize and simulate phase start time	–
chpp.hydEff	X	X	number	autosize and simulate phase start time	–

Name	Input?	Runtime?	Type	Variability	Description
chpp.ovrunF	X	X	number	run start time (of each phase, autoSize or simulate)	–
chpp.mtri	X	X	integer number	autosize and simulate phase start time	–
chpp.mw	X	X	number	run start time (of each phase, autoSize or simulate)	–
chpp.q	X	X	number	run start time (of each phase, autoSize or simulate)	–
chpp.p	X	X	number	run start time (of each phase, autoSize or simulate)	–
chcp.gpm	X	X	number	run start time (of each phase, autoSize or simulate)	–
chcp.hdLoss	X	X	number	autosize and simulate phase start time	–
chcp.motEff	X	X	number	autosize and simulate phase start time	–
chcp.hydEff	X	X	number	autosize and simulate phase start time	–
chcp.ovrunF	X	X	number	run start time (of each phase, autoSize or simulate)	–
chcp.mtri	X	X	integer number	autosize and simulate phase start time	–

Name	Input?	Runtime?	Type	Variability	Description
chcp.mw	X	X	number	run start time (of each phase, autoSize or simulate)	–
chcp.q	X	X	number	run start time (of each phase, autoSize or simulate)	–
chcp.p	X	X	number	run start time (of each phase, autoSize or simulate)	–
auxOn	X	X	number	hourly	addl input energy used in proportion to plr when on, default 0, hourly vbl for future flexblty.
auxOnMtri	X	X	integer number	input time	MTR to which to charge "auxOn"
auxOff	X	X	number	hourly	addl input energy when off for part or all of subhr (proportional to 1-plr), for unforseen uses.
auxOffMtri	X	X	integer number	input time	MTR for "auxOff"
auxOnAtall	X	X	number	hourly	addl input energy used in toto when chiller on for any part of subhour, for unforseen uses.
auxOnAtallMtri	X	X	integer number	input time	MTR for "auxOnAtall"
auxFullOff	X	X	number	hourly	additional input energy when off FOR ENTIRE SUBHOUR (as opposed to in proportion to 1-plr).
auxFullOffMtri	X	X	integer number	input time	MTR to which auxFullOff is charged, default c.mtri.
nxCh4cp	X	X	integer number	run start time (of each phase, autoSize or simulate)	0 or subscript of next CHILLER in same COOLPLANT. 1st is COOLPLANT.ch1.

Name	Input?	Runtime?	Type	Variability	Description
used	X	X	integer number	run start time (of each phase, autoSize or simulate)	non-0 if a COOLPLANT uses this chiller – else warning
eirMinUnldPlr	X	X	number	run start time (of each phase, autoSize or simulate)	chPyEirUl(minUnldPlr): precomputed energy input corr for false loading, prorate for cycling
chMode	X	X	unrecognized	end of each subhour	C_OFFONCH_OFF or _ON: whether this chiller is running, set by staging code.
cap	X	X	number	end of each subhour	capacity @ current cpTs and tCnd, set only if running
q	X	X	number	end of each subhour	this chiller's current primary output power to pri loop
p	X	X	number	end of each subhour	compressor power input. also see chpp.p, chcp.p. (Niles cndPmpPwrIn, prmPmpPwrIn, totPwrIn)
pAuxOn	X	X	number	end of each subhour	chiller-on proporotinal aux power this subhour
pAuxOff	X	X	number	end of each subhour	chiller-off proportional aux power this subhour
pAuxOnAtall	X	X	number	end of each subhour	chiller on-at-all aux power this subhour
pAuxFullOff	X	X	number	end of each subhour	auxFullOff power this subhour

6.5 @construction[1..].

Name	Input?	Runtime?	Type	Variability	Description
name	X	–	string	constant	–
conU	X	–	number	input time	–
nLr	X	–	integer number	run start time (of each phase, autoSize or simulate)	–
nFrmLr	X	–	integer number	run start time (of each phase, autoSize or simulate)	–
r	X	–	number	run start time (of each phase, autoSize or simulate)	–
hc	X	–	number	run start time (of each phase, autoSize or simulate)	–
rNom	X	–	number	run start time (of each phase, autoSize or simulate)	–

6.6 @coolPlant[1..].

Name	Input?	Runtime?	Type	Variability	Description
name	X	X	string	constant	–
cpSched	X	X	unrecognized	hourly	schedule, hourly choice of OFF, AVAIL (default), ON.
cpTsSp	X	X	number	hourly	supply temp cooling setpoint, hourly variable, default 44.
cpPipeLossF	X	X	number	autosize and simulate phase start time	–
cpTowi	X	X	integer number	input time	subscript of TOWERPLANT supporting this COOLPLANT. Input as name "cpTowerplant". RQD.
cpStage1[0]	X	X	integer number	autosize and simulate phase start time	–
cpStage1[1]	X	X	integer number	autosize and simulate phase start time	–
cpStage1[2]	X	X	integer number	autosize and simulate phase start time	–
cpStage1[3]	X	X	integer number	autosize and simulate phase start time	–
cpStage1[4]	X	X	integer number	autosize and simulate phase start time	–
cpStage1[5]	X	X	integer number	autosize and simulate phase start time	–
cpStage1[6]	X	X	integer number	autosize and simulate phase start time	–
cpStage1[7]	X	X	integer number	autosize and simulate phase start time	–
cpStage2[0]	X	X	integer number	autosize and simulate phase start time	–
cpStage2[1]	X	X	integer number	autosize and simulate phase start time	–
cpStage2[2]	X	X	integer number	autosize and simulate phase start time	–

Name	Input?	Runtime?	Type	Variability	Description
cpStage2[3]	X	X	integer number	autosize and simulate phase start time	–
cpStage2[4]	X	X	integer number	autosize and simulate phase start time	–
cpStage2[5]	X	X	integer number	autosize and simulate phase start time	–
cpStage2[6]	X	X	integer number	autosize and simulate phase start time	–
cpStage2[7]	X	X	integer number	autosize and simulate phase start time	–
cpStage3[0]	X	X	integer number	autosize and simulate phase start time	–
cpStage3[1]	X	X	integer number	autosize and simulate phase start time	–
cpStage3[2]	X	X	integer number	autosize and simulate phase start time	–
cpStage3[3]	X	X	integer number	autosize and simulate phase start time	–
cpStage3[4]	X	X	integer number	autosize and simulate phase start time	–
cpStage3[5]	X	X	integer number	autosize and simulate phase start time	–
cpStage3[6]	X	X	integer number	autosize and simulate phase start time	–
cpStage3[7]	X	X	integer number	autosize and simulate phase start time	–
cpStage4[0]	X	X	integer number	autosize and simulate phase start time	–
cpStage4[1]	X	X	integer number	autosize and simulate phase start time	–
cpStage4[2]	X	X	integer number	autosize and simulate phase start time	–
cpStage4[3]	X	X	integer number	autosize and simulate phase start time	–

Name	Input?	Runtime?	Type	Variability	Description
cpStage4[4]	X	X	integer number	autosize and simulate phase start time	–
cpStage4[5]	X	X	integer number	autosize and simulate phase start time	–
cpStage4[6]	X	X	integer number	autosize and simulate phase start time	–
cpStage4[7]	X	X	integer number	autosize and simulate phase start time	–
cpStage5[0]	X	X	integer number	autosize and simulate phase start time	–
cpStage5[1]	X	X	integer number	autosize and simulate phase start time	–
cpStage5[2]	X	X	integer number	autosize and simulate phase start time	–
cpStage5[3]	X	X	integer number	autosize and simulate phase start time	–
cpStage5[4]	X	X	integer number	autosize and simulate phase start time	–
cpStage5[5]	X	X	integer number	autosize and simulate phase start time	–
cpStage5[6]	X	X	integer number	autosize and simulate phase start time	–
cpStage5[7]	X	X	integer number	autosize and simulate phase start time	–
cpStage6[0]	X	X	integer number	autosize and simulate phase start time	–
cpStage6[1]	X	X	integer number	autosize and simulate phase start time	–
cpStage6[2]	X	X	integer number	autosize and simulate phase start time	–
cpStage6[3]	X	X	integer number	autosize and simulate phase start time	–
cpStage6[4]	X	X	integer number	autosize and simulate phase start time	–

Name	Input?	Runtime?	Type	Variability	Description
cpStage6[5]	X	X	integer number	autosize and simulate phase start time	–
cpStage6[6]	X	X	integer number	autosize and simulate phase start time	–
cpStage6[7]	X	X	integer number	autosize and simulate phase start time	–
cpStage7[0]	X	X	integer number	autosize and simulate phase start time	–
cpStage7[1]	X	X	integer number	autosize and simulate phase start time	–
cpStage7[2]	X	X	integer number	autosize and simulate phase start time	–
cpStage7[3]	X	X	integer number	autosize and simulate phase start time	–
cpStage7[4]	X	X	integer number	autosize and simulate phase start time	–
cpStage7[5]	X	X	integer number	autosize and simulate phase start time	–
cpStage7[6]	X	X	integer number	autosize and simulate phase start time	–
cpStage7[7]	X	X	integer number	autosize and simulate phase start time	–
ch1	X	X	integer number	run start time (of each phase, autoSize or simulate)	subscript of 1st CHILLER in this COOLPLANT. Next is CHILLER.nxCh4cp.
ah1	X	X	integer number	run start time (of each phase, autoSize or simulate)	subscript of 1st AH with CHW coil served by this COOLPLANT. Next is AH.ahcc.nxAh4cp.
nxCp4tp	X	X	integer number	run start time (of each phase, autoSize or simulate)	subscript of next COOLPLANT using same TOWERPLANT. 1st is TOWERPLANT.c1.
mwDsCoils	X	X	number	run start time (of each phase, autoSize or simulate)	sum of dsgn flows of connected CHW coils, accum by COOLCOIL::setup, for check vs pumps.

Name	Input?	Runtime?	Type	Variability	Description
stgPPQ[0]	X	X	number	run start time (of each phase, autoSize or simulate)	–
stgPPQ[1]	X	X	number	run start time (of each phase, autoSize or simulate)	–
stgPPQ[2]	X	X	number	run start time (of each phase, autoSize or simulate)	–
stgPPQ[3]	X	X	number	run start time (of each phase, autoSize or simulate)	–
stgPPQ[4]	X	X	number	run start time (of each phase, autoSize or simulate)	–
stgPPQ[5]	X	X	number	run start time (of each phase, autoSize or simulate)	–
stgPPQ[6]	X	X	number	run start time (of each phase, autoSize or simulate)	–
stgCPQ[0]	X	X	number	run start time (of each phase, autoSize or simulate)	–
stgCPQ[1]	X	X	number	run start time (of each phase, autoSize or simulate)	–
stgCPQ[2]	X	X	number	run start time (of each phase, autoSize or simulate)	–
stgCPQ[3]	X	X	number	run start time (of each phase, autoSize or simulate)	–
stgCPQ[4]	X	X	number	run start time (of each phase, autoSize or simulate)	–
stgCPQ[5]	X	X	number	run start time (of each phase, autoSize or simulate)	–

Name	Input?	Runtime?	Type	Variability	Description
stgCPQ[6]	X	X	number	run start time (of each phase, autoSize or simulate)	–
stgPMw[0]	X	X	number	run start time (of each phase, autoSize or simulate)	–
stgPMw[1]	X	X	number	run start time (of each phase, autoSize or simulate)	–
stgPMw[2]	X	X	number	run start time (of each phase, autoSize or simulate)	–
stgPMw[3]	X	X	number	run start time (of each phase, autoSize or simulate)	–
stgPMw[4]	X	X	number	run start time (of each phase, autoSize or simulate)	–
stgPMw[5]	X	X	number	run start time (of each phase, autoSize or simulate)	–
stgPMw[6]	X	X	number	run start time (of each phase, autoSize or simulate)	–
stgCMw[0]	X	X	number	run start time (of each phase, autoSize or simulate)	–
stgCMw[1]	X	X	number	run start time (of each phase, autoSize or simulate)	–
stgCMw[2]	X	X	number	run start time (of each phase, autoSize or simulate)	–
stgCMw[3]	X	X	number	run start time (of each phase, autoSize or simulate)	–
stgCMw[4]	X	X	number	run start time (of each phase, autoSize or simulate)	–

Name	Input?	Runtime?	Type	Variability	Description
stgCMw[5]	X	X	number	run start time (of each phase, autoSize or simulate)	–
stgCMw[6]	X	X	number	run start time (of each phase, autoSize or simulate)	–
stgN	X	X	integer number	run start time (of each phase, autoSize or simulate)	max+1 used stage subscript 1-7 (used stages need not be contiguous)
stgMxCap	X	X	integer number	run start time (of each phase, autoSize or simulate)	subscript 0-6 of stage with most design power
mxCapDs	X	X	number	run start time (of each phase, autoSize or simulate)	design power of most powerful stage (negative)
mxPMw	X	X	number	run start time (of each phase, autoSize or simulate)	largest primary pump flow, for computing minimum delta-t at runtime
mxPMwOv	X	X	number	run start time (of each phase, autoSize or simulate)	primary pump flow w/ overrun of stage with most design power, for check vs coils
mxCondQ	X	X	number	run start time (of each phase, autoSize or simulate)	max design rejected heat (positive), re defaulting TOWERPLANT capacity
mxCondGpm	X	X	number	run start time (of each phase, autoSize or simulate)	condenser pump flow of same stage (not verified largest), gpm: input value
qPipeLoss	X	X	number	run start time (of each phase, autoSize or simulate)	pipe "loss" power: cpPipeLossF * mxCapDs. Negative.
cpTs	X	X	number	end of each subhour	primary water supply temp to coils. cp- to not confuse with AH::ts when used re coil.
q	X	X	number	end of each subhour	current primary output power to coils, for results
qTow	X	X	number	end of each subhour	heat added to condenser water, incl pump heat, Btuh.

Name	Input?	Runtime?	Type	Variability	Description
tTow	X	X	number	end of each subhour	temp of water returned from chiller condensers, avail to towerplant (not used 10-92).
mwTow	X	X	number	end of each subhour	condenser water flow to towerplant, lb/hr. stgCMw[stgi].
tCnd	X	X	number	end of each subhour	heat rejection: water temp entering chiller condensers, last used TOWERPLANT.tpTs.
cpClf	X	X	integer number	end of each subhour	call-flag: set nz if must call cpCompute so it can test tr, etc to see if computation needed.
cpPtf	X	X	integer number	end of each subhour	compute-flag: set if must call cpCompute and it should unconditionally recompute this plant
cpMode	X	X	unrecognized	end of each subhour	mode this subhour: off or on: per cpSched; per demand for AVAIL. Set in cpEstimate, cpCompute.
qLoadNx	X	X	number	end of each subhour	heat added to water by loads. Negative. Believe need in rec only for debug/reporting.
qLoad	X	X	number	end of each subhour	load: sum of coil Btuh's, pipe loss. Negative. May be used in cpEstimate.
tr	X	X	number	end of each subhour	load: return water temp from coils, incl pipe loss, assuming no mw overrun.
stgi	X	X	integer number	end of each subhour	stage in use, 0-6 for cpStage1-7.
qNeed	X	X	number	end of each subhour	power needed from coolPlant to deliver water at setpoint: (cpTsSp - tr) * mw[stg]. negative.
cap	X	X	number	end of each subhour	curr capac of stgi chillers @ ts & tCnd, Btuh, incl pump heat, set by capStg().
plr	X	X	number	end of each subhour	part load ratio

Name	Input?	Runtime?	Type	Variability	Description
puteTs	X	X	number	end of each subhour	cpCompute's supply temp: cpTs, but not overwritten by cpEstimate, for debug aid/probes/reports.
cpTsSpPr	X	X	number	end of each subhour	for cpEstimate
cpTsEstPr	X	X	number	end of each subhour	for cpEstimate
cpModePr	X	X	unrecognized	end of each subhour	for cpCompute
trMxPr	X	X	number	end of each subhour	for cpCompute: tr-assuming-max-flow when last computed
qLoadPr	X	X	number	end of each subhour	for cpCompute
mwTowPr	X	X	number	end of each subhour	for cpCompute, set by tpCompute
tTowPr	X	X	number	end of each subhour	for cpCompute, set by tpCompute

6.7 @DHWDayUse[1..].

Name	Input?	Runtime?	Type	Variability	Description
name	X	X	string	constant	–
mult	X	X	number	hourly	multiplier applied to all child DHWUSE wuFlows

6.8 @DHWHeater[1..]. (owner: DHWSys)

Name	Input?	Runtime?	Type	Variability	Description
name	X	X	string	constant	–
mult	X	X	integer number	input time	count of identical water heaters (default 1)
heatSrc	X	X	unrecognized	input time	heat source
type	X	X	unrecognized	input time	heater type
desc	X	X	string	input time	probe-able description text
ashpTy	X	X	unrecognized	input time	air source heat pump type, required iff ASHPX, else ignored
znTi	X	X	integer number	input time	DHWHEATER location zone re tank loss

Name	Input?	Runtime?	Type	Variability	Description
tEx	X	X	number	subhourly	surrounding temperature, F for tank loss
ashpSrcZnTi	X	X	integer number	input time	ASHP source zone
ashpTSrc	X	X	number	subhourly	ASHP source temperature, F
ashpResUse	X	X	number	input time	resistance heat parameter for
vol	X	X	number	input time	storage tank vol, gal
EF	X	X	number	input time	rated energy factor
LDEF	X	X	number	input time	load-dependent energy factor
eff	X	X	number	input time	efficiency
HPAF	X	X	number	input time	heat pump adjustment factor
SBL	X	X	number	input time	standby loss, Btuh
pilotPwr	X	X	number	hourly	pilot light power, Btuh
parElec	X	X	number	hourly	parasitic electric use, W
tHWOut	X	X	number	hourly	average hot water temp, F (at water heater)
mixDownF	X	X	number	hourly	factor for draw adjustment re HPWH setpoint > DHWSYS::ws_tUse
elecMtri	X	X	integer number	input time	meter for system electricity use (default = parent ws_elecMtri)
fuelMtri	X	X	integer number	input time	meter for system fuel use (default = parent ws_electMtri)
inElec	X	X	number	end of each hour	primary electricity (including wh_parElec) (note not kWh)
inElecBU	X	X	number	end of each hour	backup electricity (>0 only for HPWH resistance heat)
inFuel	X	X	number	end of each hour	fuel (including wh_pilotPwr)
input	X	X	number	input time	rated input, Btuh (future use?)

Name	Input?	Runtime?	Type	Variability	Description
resHtPwr	X	X	number	input time	upper element resistance heating power, W
resHtPwr2	X	X	number	input time	lower element resistance heating power, W
HPWHHSCount	X	X	unrecognized	run start time (of each phase, autoSize or simulate)	# of HPWH heatsources in use for
HPWHUse[0]	X	X	number	end of each subhour	–
HPWHUse[1]	X	X	number	end of each subhour	–
HPWHxBU	X	X	number	run start time (of each phase, autoSize or simulate)	current step HPWH add'l backup resistance heat, Btu
HPWHTankTempSet	X	X	unrecognized	end of each hour	nz iff tank temp has been initialized
totHARL	X	X	number	hourly	cumumlative recovery load at heater, Btu
hrCount	X	X	unrecognized	hourly	# of hourly values included in wh_totHARL
totOut	X	X	number	hourly	total heat delivered to hot water, Btu
unMetSh	X	X	unrecognized	end of each hour	HPWH: count of subhrs in this hour
unMetHrs	X	X	unrecognized	end of run (of each phase, autoSize or simulate)	HPHW: annual count of hrs having

6.9 @DHWLoop[1..]. (owner: DHWSys)

Name	Input?	Runtime?	Type	Variability	Description
name	X	X	string	constant	–
mult	X	X	integer number	input time	multiplier: number of identical loops
wbCount	X	X	number	run start time (of each phase, autoSize or simulate)	total # of DHWLOOP-BRANCHs on all DHWLOOPSEGs
wlpCount	X	X	integer number	run start time (of each phase, autoSize or simulate)	total # of child DHWLOOPPUMPs

Name	Input?	Runtime?	Type	Variability	Description
flow	X	X	number	hourly	current loop recirculation max flow, gpm
runF	X	X	number	hourly	current hour recirculation operation fraction
tIn1	X	X	number	hourly	entering temperature at 1st DHWLOOPSEG
fUA	X	X	number	input time	UA adjustment factor for child DHWLOOPSEGs
HRLL	X	X	number	end of each hour	current hour loop seg pipe loss, Btu
HRBL	X	X	number	end of each hour	current hour branch pipe loss, But

6.10 @DHWLoopBranch[1..]. (owner: DHWLoopSeg)

Name	Input?	Runtime?	Type	Variability	Description
name	X	X	string	constant	–
len	X	X	number	input time	–
size	X	X	number	input time	–
insulK	X	X	number	input time	–
insulThk	X	X	number	input time	–
exH	X	X	number	input time	–
exT	X	X	number	hourly	–
vol	X	X	number	run start time (of each phase, autoSize or simulate)	–
UA	X	X	number	run start time (of each phase, autoSize or simulate)	–
fRhoCpX	X	X	number	run start time (of each phase, autoSize or simulate)	–
fvf	X	X	number	end of each hour	–
tIn	X	X	number	end of each hour	–
tOut	X	X	number	end of each hour	–
PLWF	X	X	number	end of each hour	–
PLCD	X	X	number	end of each hour	–
PL	X	X	number	end of each hour	–
mult	X	X	number	input time	# of identical branches
fWaste	X	X	number	hourly	waste fraction
flow	X	X	number	hourly	assumed flow during use, gpm
HBUL	X	X	number	end of each hour	... when water in use
HBWL	X	X	number	end of each hour	... waste loss

6.11 @DHWLoopPump[1..]. (owner: DHWLoop)

Name	Input?	Runtime?	Type	Variability	Description
name	X	X	string	constant	–
mult	X	X	integer number	input time	–
elecMtri	X	X	integer number	input time	–
pwr	X	X	number	hourly	–
inElec	X	X	number	end of each hour	–

6.12 @DHWLoopSeg[1..]. (owner: DHWLoop)

Name	Input?	Runtime?	Type	Variability	Description
name	X	X	string	constant	–
len	X	X	number	input time	–
size	X	X	number	input time	–
insulK	X	X	number	input time	–
insulThk	X	X	number	input time	–
exH	X	X	number	input time	–
exT	X	X	number	hourly	–
vol	X	X	number	run start time (of each phase, autoSize or simulate)	–
UA	X	X	number	run start time (of each phase, autoSize or simulate)	–
fRhoCpX	X	X	number	run start time (of each phase, autoSize or simulate)	–
fvf	X	X	number	end of each hour	–
tIn	X	X	number	end of each hour	–
tOut	X	X	number	end of each hour	–
PLWF	X	X	number	end of each hour	–
PLCD	X	X	number	end of each hour	–
PL	X	X	number	end of each hour	–
ty	X	X	unrecognized	input time	type: C_DHWLSEGTYCH_SUP / _RET
wbCount	X	X	number	run start time (of each phase, autoSize or simulate)	total # of child DHWLOOPBRANCHs
fNoDraw	X	X	number	hourly	fraction of hour when there is no draw
BL	X	X	number	hourly	current hour child DHWLOOPBRANCH losses, Btu

6.13 @DHWmeter[1..].

Name	Input?	Runtime?	Type	Variability	Description
name	X	X	string	constant	–
Y.total	X	X	number	end of run (of each phase, autoSize or simulate)	–
Y.unknown	X	X	number	end of run (of each phase, autoSize or simulate)	–
Y.faucet	X	X	number	end of run (of each phase, autoSize or simulate)	–
Y.shower	X	X	number	end of run (of each phase, autoSize or simulate)	–
Y.bath	X	X	number	end of run (of each phase, autoSize or simulate)	–
Y.cwashr	X	X	number	end of run (of each phase, autoSize or simulate)	–
Y.dwashr	X	X	number	end of run (of each phase, autoSize or simulate)	–
M.total	X	X	number	end of each month	–
M.unknown	X	X	number	end of each month	–
M.faucet	X	X	number	end of each month	–
M.shower	X	X	number	end of each month	–
M.bath	X	X	number	end of each month	–
M.cwashr	X	X	number	end of each month	–
M.dwashr	X	X	number	end of each month	–
D.total	X	X	number	end of each day	–
D.unknown	X	X	number	end of each day	–
D.faucet	X	X	number	end of each day	–
D.shower	X	X	number	end of each day	–
D.bath	X	X	number	end of each day	–
D.cwashr	X	X	number	end of each day	–
D.dwashr	X	X	number	end of each day	–
H.total	X	X	number	end of each hour	–
H.unknown	X	X	number	end of each hour	–
H.faucet	X	X	number	end of each hour	–
H.shower	X	X	number	end of each hour	–
H.bath	X	X	number	end of each hour	–
H.cwashr	X	X	number	end of each hour	–
H.dwashr	X	X	number	end of each hour	–

6.14 @DHWPump[1..]. (owner: DHWSys)

Name	Input?	Runtime?	Type	Variability	Description
name	X	X	string	constant	–
mult	X	X	integer number	input time	count of identical DHW pumps (default 1)
elecMtri	X	X	integer number	input time	meter for pump electricity use (default = parent ws_elecMtri)
pwr	X	X	number	hourly	pump power, W

Name	Input?	Runtime?	Type	Variability	Description
inElec	X	X	number	end of each hour	electricity (note not kWh)

6.15 @DHWSys[1..].

Name	Input?	Runtime?	Type	Variability	Description
name	X	X	string	constant	–
calcMode	X	X	unrecognized	input time	calculation mode
centralDHWSYSi	X	X	integer number	input time	index of central (parent) DHWSYS, 0 if none
loadShareDHWSYSi	X	X	integer number	input time	index of DHWSYS with which this DHWSYS shares load
mult	X	X	integer number	input time	multiplier
elecMtri	X	X	integer number	input time	meter for system electricity use
fuelMtri	X	X	integer number	input time	meter for system fuel use
inElec	X	X	number	end of each hour	electricity (note not kWh)
inFuel	X	X	number	end of each hour	fuel (for generality, always 0?)
tInlet	X	X	number	end of each hour	current hour inlet (cold water mains) temp, F
hwUse	X	X	number	hourly	current hour hot water use (at fixtures), gal
WHhwMtri	X	X	integer number	input time	DHWMTR for hot water use at water heater(s), gal
FXhwMtri	X	X	integer number	input time	DHWMTR for hot water use at fixtures, gal
fxUseHot	X	X	number	end of each hour	total hot water use at fixtures for hour, gal
fxUseMix.total	X	X	number	end of each hour	–
fxUseMix.unknown	X	X	number	end of each hour	–
fxUseMix.faucet	X	X	number	end of each hour	–
fxUseMix.shower	X	X	number	end of each hour	–
fxUseMix.bath	X	X	number	end of each hour	–
fxUseMix.cwashr	X	X	number	end of each hour	–

Name	Input?	Runtime?	Type	Variability	Description
fxUseMix.dwashr	X	X	number	end of each hour	–
fxUseMixLH.total	X	X	number	hourly	–
fxUseMixLH.unknown	X	X	number	hourly	–
fxUseMixLH.faucet	X	X	number	hourly	–
fxUseMixLH.shower	X	X	number	hourly	–
fxUseMixLH.bath	X	X	number	hourly	–
fxUseMixLH.cwashr	X	X	number	hourly	–
fxUseMixLH.dwashr	X	X	number	hourly	–
whUse.total	X	X	number	end of each hour	–
whUse.unknown	X	X	number	end of each hour	–
whUse.faucet	X	X	number	end of each hour	–
whUse.shower	X	X	number	end of each hour	–
whUse.bath	X	X	number	end of each hour	–
whUse.cwashr	X	X	number	end of each hour	–
whUse.dwashr	X	X	number	end of each hour	–
tUse	X	X	number	hourly	hot water use temp, F
tSetpoint	X	X	number	hourly	water heater set point, F
dayUsei	X	X	integer number	daily	idx of DHWDAYUSE
dayUseName	X	X	string	daily	name of DHWDAYUSE (resolved at runtime)
parElec	X	X	number	hourly	electrical parasitic power, W
SDLM	X	X	number	input time	standard distribution loss multiplier
DSM	X	X	number	input time	distribution system multiplier (AppE table RE-2)
SSF	X	X	number	hourly	solar savings fraction
WF	X	X	number	hourly	waste factor applied to ws_hwUse and DHWUSEs
whDrawF[0]	X	X	number	end of each hour	–
whDrawF[1]	X	X	number	end of each hour	–
whDrawF[2]	X	X	number	end of each hour	–
whDrawF[3]	X	X	number	end of each hour	–
whDrawF[4]	X	X	number	end of each hour	–

Name	Input?	Runtime?	Type	Variability	Description
whDrawF[5]	X	X	number	end of each hour	–
simMeth	X	X	unrecognized	run start time (of each phase, autoSize or simulate)	simulation method (see ws_DetermineSimMeth())
whCount	X	X	unrecognized	run start time (of each phase, autoSize or simulate)	# of DHWHEATERs serving this DHWSYS
wtCount	X	X	unrecognized	run start time (of each phase, autoSize or simulate)	ditto DHWTANKs
wpCount	X	X	unrecognized	run start time (of each phase, autoSize or simulate)	ditto DHWPUMPs
wlCount	X	X	unrecognized	run start time (of each phase, autoSize or simulate)	ditto DHWLOOPs (aka NLOOPk)
wbCount	X	X	number	run start time (of each phase, autoSize or simulate)	total DHWLOOP-BRANCHs, all loops
loadShareCount	X	X	unrecognized	run start time (of each phase, autoSize or simulate)	# of DHWSYSs sharing common load this group
loadShareIdx	X	X	unrecognized	run start time (of each phase, autoSize or simulate)	0-based index of this DHWSYS in shared group
loadShareWS0[0]	X	X	unrecognized	end of each hour	–
loadShareWS0[1]	X	X	unrecognized	end of each hour	–
loadShareWS0[2]	X	X	unrecognized	end of each hour	–
loadShareWS0[3]	X	X	unrecognized	end of each hour	–
loadShareWS0[4]	X	X	unrecognized	end of each hour	–

Name	Input?	Runtime?	Type	Variability	Description
loadShareWS0[5]	X	X	unrecognized	end of each hour	–
HHWO	X	X	number	end of each hour	hourly hot water output (at water heater), Btu
DLM	X	X	number	end of each hour	distribution loss multiplier (calc'd)
HRDL	X	X	number	end of each hour	hourly recirculation loss, Btuh
HJL	X	X	number	end of each hour	hourly jacket loss, Btuh
HARL	X	X	number	end of each hour	hourly adjusted recovery load, Btu

6.16 @DHWTank[1..]. (owner: DHWSys)

Name	Input?	Runtime?	Type	Variability	Description
name	X	X	string	constant	–
mult	X	X	integer number	input time	count of identical DHW tanks (default 1)
UA	X	X	number	input time	tank water-to-air UA, Btuh/F
vol	X	X	number	input time	tank volume, gal
insulR	X	X	number	input time	total tank insulation resistance, hr-F/Btuh
tTank	X	X	number	hourly	assumed tank water temperature, F
tEx	X	X	number	hourly	tank surrounding air temp, F
xLoss	X	X	number	hourly	other tank temp-independent losses, Btuh
qLoss	X	X	number	end of each hour	current hour's total loss, Btu

6.17 @DHWUse[1..]. (owner: DHWDayUse)

Name	Input?	Runtime?	Type	Variability	Description
name	X	X	string	constant	–
hwEndUse	X	X	integer number	input time	hot water end use
eventID	X	X	integer number	input time	user-defined index that identifies DHWUSEs belonging to a single
start	X	X	number	hourly	draw starting hour of day, 0 - 23.999
dur	X	X	number	hourly	flow duration, min
flow	X	X	number	hourly	mixed flow rate, gpm

Name	Input?	Runtime?	Type	Variability	Description
hotF	X	X	number	hourly	fraction hot water, default = 1
temp	X	X	number	hourly	use temperature, F. If given,
heatRecEF	X	X	number	hourly	heat recovery effectiveness

6.18 @door[1..]. (owner: surface)

Name	Input?	Runtime?	Type	Variability	Description
name	X	–	string	constant	–
ty	X	–	integer number	input time	–
area	X	–	number	run start time (of each phase, autoSize or simulate)	–
azm	X	–	number	run start time (of each phase, autoSize or simulate)	–
tilt	X	–	number	run start time (of each phase, autoSize or simulate)	–
dircos[0]	X	–	number	run start time (of each phase, autoSize or simulate)	–
dircos[1]	X	–	number	run start time (of each phase, autoSize or simulate)	–
dircos[2]	X	–	number	run start time (of each phase, autoSize or simulate)	–
depthBG	X	–	number	run start time (of each phase, autoSize or simulate)	–
model	X	–	integer number	input time	–
modelr	X	–	integer number	run start time (of each phase, autoSize or simulate)	–
lThkF	X	–	number	run start time (of each phase, autoSize or simulate)	–

Name	Input?	Runtime?	Type	Variability	Description
gti	X	–	integer number	run start time (of each phase, autoSize or simulate)	–
sco	X	–	number	monthly-hourly	–
scc	X	–	number	monthly-hourly	–
sbcI.absSlr	X	–	number	monthly-hourly	–
sbcI.awAbsSlr	X	–	number	monthly-hourly	–
sbcI.epsLW	X	–	number	run start time (of each phase, autoSize or simulate)	–
sbcI.zi	X	–	integer number	run start time (of each phase, autoSize or simulate)	–
sbcI.F	X	–	number	run start time (of each phase, autoSize or simulate)	–
sbcI.Fp	X	–	number	run start time (of each phase, autoSize or simulate)	–
sbcI.frRad	X	–	number	run start time (of each phase, autoSize or simulate)	–
sbcI.fSky	X	–	number	run start time (of each phase, autoSize or simulate)	–
sbcI.fAir	X	–	number	run start time (of each phase, autoSize or simulate)	–
sbcI.hcNat	X	–	number	end of each subhour	–
sbcI.hcFrc	X	–	number	end of each subhour	–
sbcI.hcMult	X	–	number	end of each subhour	–
sbcI.hxa	X	–	number	end of each subhour	–
sbcI.hxr	X	–	number	end of each subhour	–
sbcI.hxtot	X	–	number	end of each subhour	–
sbcI.uRat	X	–	number	end of each subhour	–
sbcI.fRat	X	–	number	end of each subhour	–

Name	Input?	Runtime?	Type	Variability	Description
sbcI.cx	X	–	number	end of each subhour	–
sbcI.sgTarg.bm	X	–	number	end of each subhour	–
sbcI.sgTarg.df	X	–	number	end of each subhour	–
sbcI.sgTarg.tot	X	–	number	end of each subhour	–
sbcI.sg	X	–	number	end of each subhour	–
sbcI.tSrf	X	–	number	end of each subhour	–
sbcI.tSrfls	X	–	number	subhourly	–
sbcI.qrAbs	X	–	number	end of each subhour	–
sbcI.txa	X	–	number	end of each subhour	–
sbcI.txr	X	–	number	end of each subhour	–
sbcI.txe	X	–	number	end of each subhour	–
sbcI.w	X	–	number	end of each subhour	–
sbcI.qSrf	X	–	number	end of each subhour	–
sbcI.pXS	X	–	unrecognized	run start time (of each phase, autoSize or simulate)	–
sbcI.si	X	–	unrecognized	run start time (of each phase, autoSize or simulate)	–
sbcI.fcWind	X	–	number	run start time (of each phase, autoSize or simulate)	–
sbcI.fcWind2	X	–	number	run start time (of each phase, autoSize or simulate)	–
sbcI.eta	X	–	number	end of each subhour	–
sbcI.widNom	X	–	number	run start time (of each phase, autoSize or simulate)	–
sbcI.lenNom	X	–	number	run start time (of each phase, autoSize or simulate)	–

Name	Input?	Runtime?	Type	Variability	Description
sbcI.lenCharNat	X	–	number	run start time (of each phase, autoSize or simulate)	–
sbcI.lenEffWink	X	–	number	run start time (of each phase, autoSize or simulate)	–
sbcI.atvDeg	X	–	number	run start time (of each phase, autoSize or simulate)	–
sbcI.cosAtv	X	–	number	run start time (of each phase, autoSize or simulate)	–
sbcI.hcModel	X	–	unrecognized	run start time (of each phase, autoSize or simulate)	–
sbcI.hcLChar	X	–	number	run start time (of each phase, autoSize or simulate)	–
sbcI.hcConst[0]	X	–	number	run start time (of each phase, autoSize or simulate)	–
sbcI.hcConst[1]	X	–	number	run start time (of each phase, autoSize or simulate)	–
sbcI.hcConst[2]	X	–	number	run start time (of each phase, autoSize or simulate)	–
sbcI.groundModel	X	–	unrecognized	run start time (of each phase, autoSize or simulate)	–
sbcI.cTaDbAvgYr	X	–	number	run start time (of each phase, autoSize or simulate)	–
sbcI.cTaDbAvg31	X	–	number	run start time (of each phase, autoSize or simulate)	–
sbcI.cTaDbAvg14	X	–	number	run start time (of each phase, autoSize or simulate)	–

Name	Input?	Runtime?	Type	Variability	Description
sbcI.cTaDbAvg07	X	–	number	run start time (of each phase, autoSize or simulate)	–
sbcI.cTGrnd	X	–	number	run start time (of each phase, autoSize or simulate)	–
sbcI.rGrnd	X	–	number	run start time (of each phase, autoSize or simulate)	–
sbcI.rConGrnd	X	–	number	run start time (of each phase, autoSize or simulate)	–
sbcO.absSlr	X	–	number	monthly-hourly	–
sbcO.awAbsSlr	X	–	number	monthly-hourly	–
sbcO.epsLW	X	–	number	run start time (of each phase, autoSize or simulate)	–
sbcO.zi	X	–	integer number	run start time (of each phase, autoSize or simulate)	–
sbcO.F	X	–	number	run start time (of each phase, autoSize or simulate)	–
sbcO.Fp	X	–	number	run start time (of each phase, autoSize or simulate)	–
sbcO.frRad	X	–	number	run start time (of each phase, autoSize or simulate)	–
sbcO.fSky	X	–	number	run start time (of each phase, autoSize or simulate)	–
sbcO.fAir	X	–	number	run start time (of each phase, autoSize or simulate)	–
sbcO.hcNat	X	–	number	end of each subhour	–
sbcO.hcFrc	X	–	number	end of each subhour	–
sbcO.hcMult	X	–	number	end of each subhour	–

Name	Input?	Runtime?	Type	Variability	Description
sbcO.hxa	X	–	number	end of each subhour	–
sbcO.hxr	X	–	number	end of each subhour	–
sbcO.hxtot	X	–	number	end of each subhour	–
sbcO.uRat	X	–	number	end of each subhour	–
sbcO.fRat	X	–	number	end of each subhour	–
sbcO.cx	X	–	number	end of each subhour	–
sbcO.sgTarg.bm	X	–	number	end of each subhour	–
sbcO.sgTarg.df	X	–	number	end of each subhour	–
sbcO.sgTarg.tot	X	–	number	end of each subhour	–
sbcO.sg	X	–	number	end of each subhour	–
sbcO.tSrf	X	–	number	end of each subhour	–
sbcO.tSrfls	X	–	number	subhourly	–
sbcO.qrAbs	X	–	number	end of each subhour	–
sbcO.txa	X	–	number	end of each subhour	–
sbcO.txr	X	–	number	end of each subhour	–
sbcO.txe	X	–	number	end of each subhour	–
sbcO.w	X	–	number	end of each subhour	–
sbcO.qSrf	X	–	number	end of each subhour	–
sbcO.pXS	X	–	unrecognized	run start time (of each phase, autoSize or simulate)	–
sbcO.si	X	–	unrecognized	run start time (of each phase, autoSize or simulate)	–
sbcO.fcWind	X	–	number	run start time (of each phase, autoSize or simulate)	–
sbcO.fcWind2	X	–	number	run start time (of each phase, autoSize or simulate)	–

Name	Input?	Runtime?	Type	Variability	Description
sbcO.eta	X	–	number	end of each subhour	–
sbcO.widNom	X	–	number	run start time (of each phase, autoSize or simulate)	–
sbcO.lenNom	X	–	number	run start time (of each phase, autoSize or simulate)	–
sbcO.lenCharNat	X	–	number	run start time (of each phase, autoSize or simulate)	–
sbcO.lenEffWink	X	–	number	run start time (of each phase, autoSize or simulate)	–
sbcO.atvDeg	X	–	number	run start time (of each phase, autoSize or simulate)	–
sbcO.cosAtv	X	–	number	run start time (of each phase, autoSize or simulate)	–
sbcO.hcModel	X	–	unrecognized	run start time (of each phase, autoSize or simulate)	–
sbcO.hcLChar	X	–	number	run start time (of each phase, autoSize or simulate)	–
sbcO.hcConst[0]	X	–	number	run start time (of each phase, autoSize or simulate)	–
sbcO.hcConst[1]	X	–	number	run start time (of each phase, autoSize or simulate)	–
sbcO.hcConst[2]	X	–	number	run start time (of each phase, autoSize or simulate)	–
sbcO.groundModel	X	–	unrecognized	run start time (of each phase, autoSize or simulate)	–

Name	Input?	Runtime?	Type	Variability	Description
sbcO.cTaDbAvgYr	X	–	number	run start time (of each phase, autoSize or simulate)	–
sbcO.cTaDbAvg31	X	–	number	run start time (of each phase, autoSize or simulate)	–
sbcO.cTaDbAvg14	X	–	number	run start time (of each phase, autoSize or simulate)	–
sbcO.cTaDbAvg07	X	–	number	run start time (of each phase, autoSize or simulate)	–
sbcO.cTGrnd	X	–	number	run start time (of each phase, autoSize or simulate)	–
sbcO.rGrnd	X	–	number	run start time (of each phase, autoSize or simulate)	–
sbcO.rConGrnd	X	–	number	run start time (of each phase, autoSize or simulate)	–
fenModel	X	–	unrecognized	input time	–
SHGC	X	–	number	input time	–
fMult	X	–	number	run start time (of each phase, autoSize or simulate)	–
UNFRC	X	–	number	input time	–
NGlz	X	–	integer number	input time	–
exShd	X	–	unrecognized	input time	–
inShd	X	–	unrecognized	input time	–
dirtLoss	X	–	number	run start time (of each phase, autoSize or simulate)	–
sfExCnd	X	–	integer number	run start time (of each phase, autoSize or simulate)	–
sfExT	X	–	number	subhourly	–
sfAdjZi	X	–	integer number	input time	–
uI	X	–	number	run start time (of each phase, autoSize or simulate)	–

Name	Input?	Runtime?	Type	Variability	Description
uC	X	–	number	run start time (of each phase, autoSize or simulate)	–
uX	X	–	number	run start time (of each phase, autoSize or simulate)	–
Rf	X	–	number	run start time (of each phase, autoSize or simulate)	–
grndRefl	X	–	number	monthly-hourly	–
vfSkyDf	X	–	number	monthly-hourly	–
vfGrndDf	X	–	number	monthly-hourly	–
vfSkyLW	X	–	number	run start time (of each phase, autoSize or simulate)	–
vfGrndLW	X	–	number	run start time (of each phase, autoSize or simulate)	–
uval	X	–	number	run start time (of each phase, autoSize or simulate)	–
UNom	X	–	number	run start time (of each phase, autoSize or simulate)	–
UANom	X	–	number	run start time (of each phase, autoSize or simulate)	–
rSrfNom[0]	X	–	number	run start time (of each phase, autoSize or simulate)	–
rSrfNom[1]	X	–	number	run start time (of each phase, autoSize or simulate)	–
hSrfNom[0]	X	–	number	run start time (of each phase, autoSize or simulate)	–
hSrfNom[1]	X	–	number	run start time (of each phase, autoSize or simulate)	–

Name	Input?	Runtime?	Type	Variability	Description
cFctr	X	–	number	run start time (of each phase, autoSize or simulate)	–
iwshad	X	–	integer number	run start time (of each phase, autoSize or simulate)	–
msi	X	–	integer number	run start time (of each phase, autoSize or simulate)	–
tLrB[0]	X	–	number	end of each hour	–
tLrB[1]	X	–	number	end of each hour	–
tLrB[2]	X	–	number	end of each hour	–
tLrB[3]	X	–	number	end of each hour	–
tLrB[4]	X	–	number	end of each hour	–
tLrB[5]	X	–	number	end of each hour	–
tLrB[6]	X	–	number	end of each hour	–
tLrB[7]	X	–	number	end of each hour	–
tLrB[8]	X	–	number	end of each hour	–
tLrB[9]	X	–	number	end of each hour	–
nsgdist	X	–	integer number	run start time (of each phase, autoSize or simulate)	–
sgdist[0].targTy	X	–	integer number	run start time (of each phase, autoSize or simulate)	–
sgdist[0].targTi	X	–	integer number	run start time (of each phase, autoSize or simulate)	–
sgdist[0].FSO	X	–	number	monthly-hourly	–
sgdist[0].FSC	X	–	number	monthly-hourly	–
sgdist[1].targTy	X	–	integer number	run start time (of each phase, autoSize or simulate)	–
sgdist[1].targTi	X	–	integer number	run start time (of each phase, autoSize or simulate)	–
sgdist[1].FSO	X	–	number	monthly-hourly	–
sgdist[1].FSC	X	–	number	monthly-hourly	–
sgdist[2].targTy	X	–	integer number	run start time (of each phase, autoSize or simulate)	–

Name	Input?	Runtime?	Type	Variability	Description
sgdist[2].targTi	X	–	integer number	run start time (of each phase, autoSize or simulate)	–
sgdist[2].FSO	X	–	number	monthly-hourly	–
sgdist[2].FSC	X	–	number	monthly-hourly	–
sgdist[3].targTy	X	–	integer number	run start time (of each phase, autoSize or simulate)	–
sgdist[3].targTi	X	–	integer number	run start time (of each phase, autoSize or simulate)	–
sgdist[3].FSO	X	–	number	monthly-hourly	–
sgdist[3].FSC	X	–	number	monthly-hourly	–
sgdist[4].targTy	X	–	integer number	run start time (of each phase, autoSize or simulate)	–
sgdist[4].targTi	X	–	integer number	run start time (of each phase, autoSize or simulate)	–
sgdist[4].FSO	X	–	number	monthly-hourly	–
sgdist[4].FSC	X	–	number	monthly-hourly	–
sgdist[5].targTy	X	–	integer number	run start time (of each phase, autoSize or simulate)	–
sgdist[5].targTi	X	–	integer number	run start time (of each phase, autoSize or simulate)	–
sgdist[5].FSO	X	–	number	monthly-hourly	–
sgdist[5].FSC	X	–	number	monthly-hourly	–
sgdist[6].targTy	X	–	integer number	run start time (of each phase, autoSize or simulate)	–
sgdist[6].targTi	X	–	integer number	run start time (of each phase, autoSize or simulate)	–
sgdist[6].FSO	X	–	number	monthly-hourly	–
sgdist[6].FSC	X	–	number	monthly-hourly	–
sgdist[7].targTy	X	–	integer number	run start time (of each phase, autoSize or simulate)	–

Name	Input?	Runtime?	Type	Variability	Description
sgdist[7].targTi	X	–	integer number	run start time (of each phase, autoSize or simulate)	–
sgdist[7].FSO	X	–	number	monthly-hourly	–
sgdist[7].FSC	X	–	number	monthly-hourly	–
sfClass	X	–	unrecognized	input time	–
sfArea	X	–	number	input time	–
sfU	X	–	number	input time	–
sfCon	X	–	integer number	input time	–
sfTy	X	–	integer number	constant	–
width	X	–	number	input time	–
height	X	–	number	input time	–
mult	X	–	number	input time	–
xi	X	–	integer number	run start time (of each phase, autoSize or simulate)	–
msi	X	–	integer number	run start time (of each phase, autoSize or simulate)	–

6.19 @DuctSeg[1..]. (owner: RSYS)

Name	Input?	Runtime?	Type	Variability	Description
name	X	X	string	constant	–
ty	X	X	unrecognized	input time	effective duct insulation resistance; typ value = 0.4
absSlr	X	X	number	subhourly	–
awAbsSlr	X	X	number	subhourly	–
epsLW	X	X	number	subhourly	–
zi	X	X	integer number	subhourly	–
F	X	X	number	subhourly	–
Fp	X	X	number	subhourly	–
frRad	X	X	number	subhourly	–
fSky	X	X	number	subhourly	–
fAir	X	X	number	subhourly	–
hcNat	X	X	number	end of each subhour	–
hcFrc	X	X	number	end of each subhour	–
hcMult	X	X	number	end of each subhour	–
hxa	X	X	number	end of each subhour	–
hxr	X	X	number	end of each subhour	–

Name	Input?	Runtime?	Type	Variability	Description
hxtot	X	X	number	end of each subhour	–
uRat	X	X	number	end of each subhour	–
fRat	X	X	number	end of each subhour	–
cx	X	X	number	end of each subhour	–
sgTarg.bm	X	X	number	end of each subhour	–
sgTarg.df	X	X	number	end of each subhour	–
sgTarg.tot	X	X	number	end of each subhour	–
sg	X	X	number	end of each subhour	–
tSrf	X	X	number	end of each subhour	–
tSrfls	X	X	number	subhourly	–
qrAbs	X	X	number	end of each subhour	–
txa	X	X	number	end of each subhour	–
txr	X	X	number	end of each subhour	–
txe	X	X	number	end of each subhour	–
w	X	X	number	end of each subhour	–
qSrf	X	X	number	end of each subhour	–
pDS	X	X	unrecognized	subhourly	type: C_DUCTTYCH_RET / _SUP
exArea	X	X	number	input time	Never 0 assuming rs_RconvIn > 0
diam	X	X	number	input time	–
len	X	X	number	input time	effective insulation resistance, ft2-F/Btuh
inArea	X	X	number	input time	–
insulR	X	X	number	input time	cur step total conductance between duct air
insulMati	X	X	integer number	input time	and surrounding equivalent temp, Btuh/F
insulKA	X	X	number	run start time (of each phase, autoSize or simulate)	cur step conduction loss parameter (1 - effectiveness)

Name	Input?	Runtime?	Type	Variability	Description
insulKB	X	X	number	run start time (of each phase, autoSize or simulate)	depends only on ds_uaTot and air mass flow
insulThk	X	X	number	run start time (of each phase, autoSize or simulate)	–
insulThkEff	X	X	number	run start time (of each phase, autoSize or simulate)	cur step air states
RconvIn	X	X	number	autosize and simulate phase start time	[2]=average (consistent w/ conduction loss)
Rduct	X	X	number	end of each hour	dry air mass flow rate at full load, lbm/hr
Uduct	X	X	number	end of each hour	ds_qCondAirFL + ds_qCondRadFL = ds_qCondFL
insulREff	X	X	number	end of each hour	DUCTSEG
exCnd	X	X	integer number	input time	–
leakF	X	X	number	input time	*excon // explicit constructor
uaTot	X	X	number	end of each subhour	explicit destructor
beta	X	X	number	end of each subhour	–
air[0].tdb	X	X	number	end of each subhour	–
air[0].w	X	X	number	end of each subhour	–
air[1].tdb	X	X	number	end of each subhour	–
air[1].w	X	X	number	end of each subhour	–
air[2].tdb	X	X	number	end of each subhour	–
air[2].w	X	X	number	end of each subhour	–
air[3].tdb	X	X	number	end of each subhour	–
air[3].w	X	X	number	end of each subhour	–
amfFL	X	X	number	end of each subhour	–
qCondFL	X	X	number	end of each subhour	–
qCondAirFL	X	X	number	end of each subhour	–

Name	Input?	Runtime?	Type	Variability	Description
qCondRadFL	X	X	number	end of each subhour	–

6.20 @export[1..]. (owner: exportFile)

Name	Input?	Runtime?	Type	Variability	Description
name	X	–	string	constant	–
zi	X	–	integer number	input time	–
mtri	X	–	integer number	input time	–
ahi	X	–	integer number	input time	–
tui	X	–	integer number	input time	–
dhwMtri	X	–	integer number	input time	–
isExport	X	–	integer number	input time	–
rpTy	X	–	integer number	constant	–
rpFreq	X	–	integer number	constant	–
rpDayBeg	X	–	integer number	input time	–
rpDayEnd	X	–	integer number	input time	–
rpBtuSf	X	–	number	input time	–
rpCond	X	–	number	end of each subhour	–
rpTitle	X	–	string	input time	–
rpCpl	X	–	integer number	input time	–
rpHeader	X	–	unrecognized	input time	–
rpFooter	X	–	integer number	input time	–
coli	X	–	integer number	run start time (of each phase, autoSize or simulate)	–
nCol	X	–	integer number	run start time (of each phase, autoSize or simulate)	–
wid	X	–	integer number	run start time (of each phase, autoSize or simulate)	–
vrh	X	–	unrecognized	run start time (of each phase, autoSize or simulate)	–

6.21 @exportCol[1..]. (owner: export)

Name	Input?	Runtime?	Type	Variability	Description
name	X	X	string	constant	–
colHead	X	X	string	input time	–
colGap	X	X	integer number	input time	–
colWid	X	X	integer number	input time	–
colDec	X	X	integer number	input time	–
colJust	X	X	integer number	input time	–
colVal	X	X	un-probe-able	end of each subhour	–
nxColi	X	X	integer number	constant	–

6.22 @exportFile[1..].

Name	Input?	Runtime?	Type	Variability	Description
name	X	–	string	constant	–
fileName	X	–	string	input time	–
fileStat	X	–	integer number	run start time (of each phase, autoSize or simulate)	–
pageFmt	X	–	integer number	input time	–
fileStatChecked	X	–	integer number	run start time (of each phase, autoSize or simulate)	–
overWrite	X	–	integer number	run start time (of each phase, autoSize or simulate)	–

6.23 @gain[1..]. (owner: zone)

Name	Input?	Runtime?	Type	Variability	Description
name	X	X	string	constant	–
gnPower	X	X	number	hourly	amount of gain (demand – b4 reduction by gnDlFrPow), Btuh, hourly expression
mtri	X	X	integer number	input time	meter to which gain is charged
gnEndUse	X	X	integer number	autosize and simulate phase start time	end use of energy: cooling, heating, receptacles, etc. reqd if gnMeter != none, else disallowed.
gnFrLat	X	X	number	hourly	fraction of gain which is latent (0 - 1, hourly expression)
gnFrRad	X	X	number	hourly	fraction of gain which is radiant, added 11-95
gnFrZn	X	X	number	hourly	fraction of gain going to zone (0 - 1, hourly expression)
gnFrPl	X	X	number	hourly	fraction of gain going to plenum (0 - 1, hourly expression)
gnFrRtn	X	X	number	hourly	fraction of gain going to return (0 - 1, hourly expression)
gnDlFrPow	X	X	number	hourly	fraction power on for daylighting, 0-1, default 1.0, hourly expression
dhwsysi	X	X	integer number	input time	controlling DHWSYS, 0 if none

Name	Input?	Runtime?	Type	Variability	Description
dhwEndUse	X	X	integer number	input time	with gn_dhwsysi, specifies controlling HW end use

6.24 @glazeType[1..].

Name	Input?	Runtime?	Type	Variability	Description
name	X	X	string	constant	–
gtSHGC	X	X	number	input time	–
gtSMSO	X	X	number	monthly-hourly	–
gtSMSC	X	X	number	monthly-hourly	–
gtFMult	X	X	number	input time	–
gtPySHGC.k[0]	X	X	number	autosize and simulate phase start time	–
gtPySHGC.k[1]	X	X	number	autosize and simulate phase start time	–
gtPySHGC.k[2]	X	X	number	autosize and simulate phase start time	–
gtPySHGC.k[3]	X	X	number	autosize and simulate phase start time	–
gtPySHGC.k[4]	X	X	number	autosize and simulate phase start time	–
gtPySHGC.k[5]	X	X	number	autosize and simulate phase start time	–
gtDMSHGC	X	X	number	input time	–
gtDMRBSol	X	X	number	input time	–
gtU	X	X	number	input time	–
gtUNFRC	X	X	number	input time	–
gtNGlz	X	X	integer number	input time	–
gtFenModel	X	X	unrecognized	input time	–
gtExShd	X	X	unrecognized	input time	–
gtInShd	X	X	unrecognized	input time	–
gtDirtLoss	X	X	number	input time	–

6.25 @heatPlant[1..].

Name	Input?	Runtime?	Type	Variability	Description
name	X	X	string	constant	–
hpSched	X	X	unrecognized	hourly	hourly choice of OFF, AVAIL (default; plant runs on demand), or ON (at least 1st stage runs).
hpPipeLossF	X	X	number	autosize and simulate phase start time	pipe loss, default .01, fraction of largest stage boiler capac whenever any boiler running

Name	Input?	Runtime?	Type	Variability	Description
hpStage1[0]	X	X	integer number	autosize and simulate phase start time	–
hpStage1[1]	X	X	integer number	autosize and simulate phase start time	–
hpStage1[2]	X	X	integer number	autosize and simulate phase start time	–
hpStage1[3]	X	X	integer number	autosize and simulate phase start time	–
hpStage1[4]	X	X	integer number	autosize and simulate phase start time	–
hpStage1[5]	X	X	integer number	autosize and simulate phase start time	–
hpStage1[6]	X	X	integer number	autosize and simulate phase start time	–
hpStage1[7]	X	X	integer number	autosize and simulate phase start time	–
hpStage2[0]	X	X	integer number	autosize and simulate phase start time	–
hpStage2[1]	X	X	integer number	autosize and simulate phase start time	–
hpStage2[2]	X	X	integer number	autosize and simulate phase start time	–
hpStage2[3]	X	X	integer number	autosize and simulate phase start time	–
hpStage2[4]	X	X	integer number	autosize and simulate phase start time	–
hpStage2[5]	X	X	integer number	autosize and simulate phase start time	–
hpStage2[6]	X	X	integer number	autosize and simulate phase start time	–
hpStage2[7]	X	X	integer number	autosize and simulate phase start time	–
hpStage3[0]	X	X	integer number	autosize and simulate phase start time	–

Name	Input?	Runtime?	Type	Variability	Description
hpStage3[1]	X	X	integer number	autosize and simulate phase start time	–
hpStage3[2]	X	X	integer number	autosize and simulate phase start time	–
hpStage3[3]	X	X	integer number	autosize and simulate phase start time	–
hpStage3[4]	X	X	integer number	autosize and simulate phase start time	–
hpStage3[5]	X	X	integer number	autosize and simulate phase start time	–
hpStage3[6]	X	X	integer number	autosize and simulate phase start time	–
hpStage3[7]	X	X	integer number	autosize and simulate phase start time	–
hpStage4[0]	X	X	integer number	autosize and simulate phase start time	–
hpStage4[1]	X	X	integer number	autosize and simulate phase start time	–
hpStage4[2]	X	X	integer number	autosize and simulate phase start time	–
hpStage4[3]	X	X	integer number	autosize and simulate phase start time	–
hpStage4[4]	X	X	integer number	autosize and simulate phase start time	–
hpStage4[5]	X	X	integer number	autosize and simulate phase start time	–
hpStage4[6]	X	X	integer number	autosize and simulate phase start time	–
hpStage4[7]	X	X	integer number	autosize and simulate phase start time	–
hpStage5[0]	X	X	integer number	autosize and simulate phase start time	–
hpStage5[1]	X	X	integer number	autosize and simulate phase start time	–

Name	Input?	Runtime?	Type	Variability	Description
hpStage5[2]	X	X	integer number	autosize and simulate phase start time	–
hpStage5[3]	X	X	integer number	autosize and simulate phase start time	–
hpStage5[4]	X	X	integer number	autosize and simulate phase start time	–
hpStage5[5]	X	X	integer number	autosize and simulate phase start time	–
hpStage5[6]	X	X	integer number	autosize and simulate phase start time	–
hpStage5[7]	X	X	integer number	autosize and simulate phase start time	–
hpStage6[0]	X	X	integer number	autosize and simulate phase start time	–
hpStage6[1]	X	X	integer number	autosize and simulate phase start time	–
hpStage6[2]	X	X	integer number	autosize and simulate phase start time	–
hpStage6[3]	X	X	integer number	autosize and simulate phase start time	–
hpStage6[4]	X	X	integer number	autosize and simulate phase start time	–
hpStage6[5]	X	X	integer number	autosize and simulate phase start time	–
hpStage6[6]	X	X	integer number	autosize and simulate phase start time	–
hpStage6[7]	X	X	integer number	autosize and simulate phase start time	–
hpStage7[0]	X	X	integer number	autosize and simulate phase start time	–
hpStage7[1]	X	X	integer number	autosize and simulate phase start time	–
hpStage7[2]	X	X	integer number	autosize and simulate phase start time	–

Name	Input?	Runtime?	Type	Variability	Description
hpStage7[3]	X	X	integer number	autosize and simulate phase start time	–
hpStage7[4]	X	X	integer number	autosize and simulate phase start time	–
hpStage7[5]	X	X	integer number	autosize and simulate phase start time	–
hpStage7[6]	X	X	integer number	autosize and simulate phase start time	–
hpStage7[7]	X	X	integer number	autosize and simulate phase start time	–
blr1	X	X	integer number	run start time (of each phase, autoSize or simulate)	subscript of 1st BOILER for this HEATPLANT. Next is BOILER.nxBlr4hp.
tu1	X	X	integer number	run start time (of each phase, autoSize or simulate)	subscript of 1st TU with HW coil served by this HEATPLANT. Next is TU.tuhc.nxTu4hp.
ah1	X	X	integer number	run start time (of each phase, autoSize or simulate)	subscript of 1st AH with HW coil served by this HEATPLANT. Next is AH.ahhc.nxAh4hp.
hl1	X	X	integer number	run start time (of each phase, autoSize or simulate)	subscript of 1st HPLOOP with HX for this HEATPLANT
qPipeLoss	X	X	number	run start time (of each phase, autoSize or simulate)	pipe loss power: hpPipeLossF * capStg[stgMxQ]
stgCap[0]	X	X	number	run start time (of each phase, autoSize or simulate)	–
stgCap[1]	X	X	number	run start time (of each phase, autoSize or simulate)	–
stgCap[2]	X	X	number	run start time (of each phase, autoSize or simulate)	–

Name	Input?	Runtime?	Type	Variability	Description
stgCap[3]	X	X	number	run start time (of each phase, autoSize or simulate)	–
stgCap[4]	X	X	number	run start time (of each phase, autoSize or simulate)	–
stgCap[5]	X	X	number	run start time (of each phase, autoSize or simulate)	–
stgCap[6]	X	X	number	run start time (of each phase, autoSize or simulate)	–
stgPQ[0]	X	X	number	run start time (of each phase, autoSize or simulate)	–
stgPQ[1]	X	X	number	run start time (of each phase, autoSize or simulate)	–
stgPQ[2]	X	X	number	run start time (of each phase, autoSize or simulate)	–
stgPQ[3]	X	X	number	run start time (of each phase, autoSize or simulate)	–
stgPQ[4]	X	X	number	run start time (of each phase, autoSize or simulate)	–
stgPQ[5]	X	X	number	run start time (of each phase, autoSize or simulate)	–
stgPQ[6]	X	X	number	run start time (of each phase, autoSize or simulate)	–
stgN	X	X	integer number	run start time (of each phase, autoSize or simulate)	max+1 used stage subscript 1-7 (used stages need not be contiguous)
stgMxQ	X	X	integer number	run start time (of each phase, autoSize or simulate)	most powerful stage subscript 0-6

Name	Input?	Runtime?	Type	Variability	Description
hpClf	X	X	integer number	end of each subhour	call-flag: set nz if must call hpCompute so it can test tr, etc to see if computation needed.
hpPtf	X	X	integer number	end of each subhour	compute-flag: set if must call hpCompute and it should unconditionally recompute this plant.
hpMode	X	X	unrecognized	end of each subhour	mode this subhour: off or on: per hpSched; per demand for AVAIL. Set in hpEstimate, hpCompute.
capF	X	X	number	end of each subhour	1.0 or, when overloaded, derating fraction for capacity of each coil/hx.
stgi	X	X	integer number	end of each subhour	stage in use, 0-6 for hpStage1-7.
qNx	X	X	number	end of each subhour	latest coil/hx load, copied to .q at decision to compute, else may remain slightly different.
q	X	X	number	end of each subhour	–
qPk	X	X	number	end of each subhour	peak load re error autosizing overload message
qPkAs	X	X	number	end of each subhour	peak load on a converged autoSizing design day re error autosizing overload message
hpModePr	X	X	unrecognized	end of each subhour	–
qPr	X	X	number	end of each subhour	–
capFPr	X	X	number	end of each subhour	–

6.26 @holiday[1..].

Name	Input?	Runtime?	Type	Variability	Description
name	X	–	string	constant	–
hdDateTrue	X	–	integer number	input time	–
hdDateObs	X	–	integer number	input time	–

Name	Input?	Runtime?	Type	Variability	Description
hdOnMonday	X	–	integer number	input time	–
hdCase	X	–	unrecognized	input time	–
hdDow	X	–	integer number	input time	–
hdMon	X	–	unrecognized	input time	–

6.27 @impFileFldNames[1..].

Name	Input?	Runtime?	Type	Variability	Description
name	–	X	string	constant	–
impfi	–	X	integer number	input time	–
fnmiN	–	X	integer number	input time	–

6.28 @importFile[1..].

Name	Input?	Runtime?	Type	Variability	Description
name	X	X	string	constant	–
fileName	X	X	string	autosize and simulate phase start time	–
imTitle	X	X	string	autosize and simulate phase start time	–
imPhaseSpare	X	X	integer number	constant	–
imFreq	X	X	integer number	input time	–
hasHeader	X	X	integer number	autosize and simulate phase start time	–
iffnmi	X	X	integer number	run start time (of each phase, autoSize or simulate)	–
isOpen	X	X	integer number	run start time (of each phase, autoSize or simulate)	–
fh	X	X	integer number	run start time (of each phase, autoSize or simulate)	–
posEndHdr	X	X	number	run start time (of each phase, autoSize or simulate)	–
bufSz	X	X	integer number	run start time (of each phase, autoSize or simulate)	–
bufN	X	X	integer number	hourly	–
eofRead	X	X	integer number	hourly	–
eof	X	X	integer number	hourly	–

Name	Input?	Runtime?	Type	Variability	Description
bufI1	X	X	integer number	hourly	–
bufI2	X	X	integer number	hourly	–
lineNo	X	X	integer number	hourly	–
lineNoEndHdr	X	X	integer number	run start time (of each phase, autoSize or simulate)	–
nFieldsScanned	X	X	integer number	end of each hour	–
eorScanned	X	X	integer number	end of each hour	–

6.29 @izXfer[1..].

Name	Input?	Runtime?	Type	Variability	Description
name	X	X	string	constant	–
zi1	X	X	integer number	input time	–
zi2	X	X	integer number	input time	–
ua	X	X	number	hourly	–
nvcntrl	X	X	integer number	input time	–
a1	X	X	number	hourly	–
a2	X	X	number	hourly	–
L1	X	X	number	input time	–
L2	X	X	number	input time	–
hz	X	X	number	input time	–
stairAngle	X	X	number	input time	–
cd	X	X	number	input time	–
exp	X	X	number	run start time (of each phase, autoSize or simulate)	–
cpr	X	X	number	input time	–
vfMin	X	X	number	subhourly	–
vfMax	X	X	number	subhourly	–
ASEF	X	X	number	subhourly	–
LEF	X	X	number	subhourly	–
vfExhRat	X	X	number	subhourly	–
EATR	X	X	number	subhourly	–
fan.fanTy	X	X	unrecognized	autosize and simulate phase start time	–
fan.vfDs	X	X	number	end of each subhour	–
fan.vfDs_As	X	X	number	autosize and simulate phase start time	–
fan.vfDs_AsNov	X	X	number	autosize and simulate phase start time	–

Name	Input?	Runtime?	Type	Variability	Description
fan.vfMxF	X	X	number	autosize and simulate phase start time	–
fan.press	X	X	number	run start time (of each phase, autoSize or simulate)	–
fan.eff	X	X	number	run start time (of each phase, autoSize or simulate)	–
fan.shaftPwr	X	X	number	run start time (of each phase, autoSize or simulate)	–
fan.elecPwr	X	X	number	run start time (of each phase, autoSize or simulate)	–
fan.motTy	X	X	unrecognized	run start time (of each phase, autoSize or simulate)	–
fan.motEff	X	X	number	autosize and simulate phase start time	–
fan.motPos	X	X	unrecognized	autosize and simulate phase start time	–
fan.curvePy.k[0]	X	X	number	autosize and simulate phase start time	–
fan.curvePy.k[1]	X	X	number	autosize and simulate phase start time	–
fan.curvePy.k[2]	X	X	number	autosize and simulate phase start time	–
fan.curvePy.k[3]	X	X	number	autosize and simulate phase start time	–
fan.curvePy.k[4]	X	X	number	autosize and simulate phase start time	–
fan.curvePy.k[5]	X	X	number	autosize and simulate phase start time	–
fan.mtri	X	X	integer number	input time	–
fan.endUse	X	X	integer number	autosize and simulate phase start time	–

Name	Input?	Runtime?	Type	Variability	Description
fan.ausz	X	X	integer number	run start time (of each phase, autoSize or simulate)	–
fan.outPower	X	X	number	subhourly	–
fan.airPower	X	X	number	subhourly	–
fan.cMx	X	X	number	end of each subhour	–
fan.c	X	X	number	end of each subhour	–
fan.t	X	X	number	end of each subhour	–
fan.frOn	X	X	number	end of each subhour	–
fan.p	X	X	number	end of each subhour	–
fan.q	X	X	number	end of each subhour	–
fan.dT	X	X	number	end of each subhour	–
fan.qAround	X	X	number	end of each subhour	–
nvcoeff	X	X	number	run start time (of each phase, autoSize or simulate)	–
air1.tdb	X	X	number	end of each subhour	–
air1.w	X	X	number	end of each subhour	–
air2.tdb	X	X	number	end of each subhour	–
air2.w	X	X	number	end of each subhour	–
rho1	X	X	number	subhourly	–
rho2	X	X	number	subhourly	–
ad[0].Ae	X	X	number	end of each subhour	–
ad[0].AeLin	X	X	number	end of each subhour	–
ad[0].delP	X	X	number	end of each subhour	–
ad[0].mdotP	X	X	number	end of each subhour	–
ad[0].dmdp	X	X	number	end of each subhour	–
ad[0].mdotB	X	X	number	end of each subhour	–
ad[0].mdotX	X	X	number	end of each subhour	–
ad[0].xDelpF	X	X	number	end of each subhour	–

Name	Input?	Runtime?	Type	Variability	Description
ad[0].xMbm	X	X	number	end of each subhour	–
ad[0].tdFan	X	X	number	end of each subhour	–
ad[0].pFan	X	X	number	end of each subhour	–
ad[1].Ae	X	X	number	end of each subhour	–
ad[1].AeLin	X	X	number	end of each subhour	–
ad[1].delP	X	X	number	end of each subhour	–
ad[1].mdotP	X	X	number	end of each subhour	–
ad[1].dmdp	X	X	number	end of each subhour	–
ad[1].mdotB	X	X	number	end of each subhour	–
ad[1].mdotX	X	X	number	end of each subhour	–
ad[1].xDelpF	X	X	number	end of each subhour	–
ad[1].xMbm	X	X	number	end of each subhour	–
ad[1].tdFan	X	X	number	end of each subhour	–
ad[1].pFan	X	X	number	end of each subhour	–
amfNom	X	X	number	end of each subhour	–

6.30 @layer[1..]. (owner: construction)

Name	Input?	Runtime?	Type	Variability	Description
name	X	–	string	constant	–
thk	X	–	number	input time	–
mati	X	–	integer number	input time	–
frmMati	X	–	integer number	input time	–
frmFrac	X	–	number	input time	–
uvy	X	–	number	run start time (of each phase, autoSize or simulate)	–
r	X	–	number	run start time (of each phase, autoSize or simulate)	–
vhc	X	–	number	run start time (of each phase, autoSize or simulate)	–

6.31 @mass[1..].

Name	Input?	Runtime?	Type	Variability	Description
name	–	X	string	constant	–
sfi	–	X	integer number	run start time (of each phase, autoSize or simulate)	–
sfClass	–	X	unrecognized	run start time (of each phase, autoSize or simulate)	–
xri	–	X	integer number	run start time (of each phase, autoSize or simulate)	–
area	–	X	number	run start time (of each phase, autoSize or simulate)	–
isSubhrly	–	X	integer number	run start time (of each phase, autoSize or simulate)	–
isFD	–	X	integer number	run start time (of each phase, autoSize or simulate)	–
inside.msi	–	X	integer number	run start time (of each phase, autoSize or simulate)	–
inside.ty	–	X	integer number	run start time (of each phase, autoSize or simulate)	–
inside.zi	–	X	integer number	run start time (of each phase, autoSize or simulate)	–
inside.exTa	–	X	number	hourly	–
inside.exTr	–	X	number	hourly	–
inside.rsurf	–	X	number	run start time (of each phase, autoSize or simulate)	–
inside.h	–	X	number	run start time (of each phase, autoSize or simulate)	–

Name	Input?	Runtime?	Type	Variability	Description
inside.ha	–	X	number	run start time (of each phase, autoSize or simulate)	–
inside.rIg	–	X	unrecognized	hourly	–
inside.qxhnet	–	X	number	end of each hour	–
inside.qxdnet	–	X	number	end of each day	–
inside.qxmnet	–	X	number	end of each month	–
inside.qxhtot	–	X	number	end of each hour	–
inside.qxdtot	–	X	number	end of each day	–
inside.qxmtot	–	X	number	end of each month	–
inside.surfTemp	–	X	number	end of each subhour	–
outside.msi	–	X	integer number	run start time (of each phase, autoSize or simulate)	–
outside.ty	–	X	integer number	run start time (of each phase, autoSize or simulate)	–
outside.zi	–	X	integer number	run start time (of each phase, autoSize or simulate)	–
outside.exTa	–	X	number	hourly	–
outside.exTr	–	X	number	hourly	–
outside.rsurf	–	X	number	run start time (of each phase, autoSize or simulate)	–
outside.h	–	X	number	run start time (of each phase, autoSize or simulate)	–
outside.ha	–	X	number	run start time (of each phase, autoSize or simulate)	–
outside.rIg	–	X	unrecognized	hourly	–
outside.qxhnet	–	X	number	end of each hour	–
outside.qxdnet	–	X	number	end of each day	–
outside.qxmnet	–	X	number	end of each month	–
outside.qxhtot	–	X	number	end of each hour	–
outside.qxdtot	–	X	number	end of each day	–
outside.qxmtot	–	X	number	end of each month	–
outside.surfTemp	–	X	number	end of each subhour	–

Name	Input?	Runtime?	Type	Variability	Description
UNom	–	X	number	run start time (of each phase, autoSize or simulate)	–
tc	–	X	number	run start time (of each phase, autoSize or simulate)	–
pMM	–	X	un-probe-able	run start time (of each phase, autoSize or simulate)	–

6.32 @material[1..].

Name	Input?	Runtime?	Type	Variability	Description
name	X	–	string	constant	–
thk	X	–	number	input time	–
cond	X	–	number	input time	–
condTRat	X	–	number	input time	–
condCT	X	–	number	input time	–
spHt	X	–	number	input time	–
dens	X	–	number	input time	–
rNom	X	–	number	input time	–
vhc	X	–	number	run start time (of each phase, autoSize or simulate)	–

6.33 @meter[1..].

Name	Input?	Runtime?	Type	Variability	Description
name	X	X	string	constant	–
rate	X	X	number	input time	–
dmdRate	X	X	number	input time	–
Y.tot	X	X	number	end of run (of each phase, autoSize or simulate)	–
Y.clg	X	X	number	end of run (of each phase, autoSize or simulate)	–
Y.htg	X	X	number	end of run (of each phase, autoSize or simulate)	–
Y.hp	X	X	number	end of run (of each phase, autoSize or simulate)	–
Y.dhw	X	X	number	end of run (of each phase, autoSize or simulate)	–

Name	Input?	Runtime?	Type	Variability	Description
Y.dhwBU	X	X	number	end of run (of each phase, autoSize or simulate)	–
Y.fanC	X	X	number	end of run (of each phase, autoSize or simulate)	–
Y.fanH	X	X	number	end of run (of each phase, autoSize or simulate)	–
Y.fanV	X	X	number	end of run (of each phase, autoSize or simulate)	–
Y.fan	X	X	number	end of run (of each phase, autoSize or simulate)	–
Y.aux	X	X	number	end of run (of each phase, autoSize or simulate)	–
Y.proc	X	X	number	end of run (of each phase, autoSize or simulate)	–
Y.lit	X	X	number	end of run (of each phase, autoSize or simulate)	–
Y.rcp	X	X	number	end of run (of each phase, autoSize or simulate)	–
Y.ext	X	X	number	end of run (of each phase, autoSize or simulate)	–
Y.refr	X	X	number	end of run (of each phase, autoSize or simulate)	–
Y.dish	X	X	number	end of run (of each phase, autoSize or simulate)	–
Y.dry	X	X	number	end of run (of each phase, autoSize or simulate)	–
Y.wash	X	X	number	end of run (of each phase, autoSize or simulate)	–
Y.cook	X	X	number	end of run (of each phase, autoSize or simulate)	–
Y.usr1	X	X	number	end of run (of each phase, autoSize or simulate)	–
Y.usr2	X	X	number	end of run (of each phase, autoSize or simulate)	–

Name	Input?	Runtime?	Type	Variability	Description
Y.pv	X	X	number	end of run (of each phase, autoSize or simulate)	–
Y.cost	X	X	number	end of run (of each phase, autoSize or simulate)	–
Y.dmdCost	X	X	number	end of run (of each phase, autoSize or simulate)	–
Y.dmd	X	X	number	end of run (of each phase, autoSize or simulate)	–
Y.dmdShoy	X	X	unrecognized	end of run (of each phase, autoSize or simulate)	–
M.tot	X	X	number	end of each month	–
M.clg	X	X	number	end of each month	–
M.htg	X	X	number	end of each month	–
M.hp	X	X	number	end of each month	–
M.dhw	X	X	number	end of each month	–
M.dhwBU	X	X	number	end of each month	–
M.fanC	X	X	number	end of each month	–
M.fanH	X	X	number	end of each month	–
M.fanV	X	X	number	end of each month	–
M.fan	X	X	number	end of each month	–
M.aux	X	X	number	end of each month	–
M.proc	X	X	number	end of each month	–
M.lit	X	X	number	end of each month	–
M.rcp	X	X	number	end of each month	–
M.ext	X	X	number	end of each month	–
M.refr	X	X	number	end of each month	–
M.dish	X	X	number	end of each month	–
M.dry	X	X	number	end of each month	–
M.wash	X	X	number	end of each month	–
M.cook	X	X	number	end of each month	–
M.usr1	X	X	number	end of each month	–
M.usr2	X	X	number	end of each month	–
M.pv	X	X	number	end of each month	–
M.cost	X	X	number	end of each month	–
M.dmdCost	X	X	number	end of each month	–
M.dmd	X	X	number	end of each month	–
M.dmdShoy	X	X	unrecognized	end of each month	–
D.tot	X	X	number	end of each day	–
D.clg	X	X	number	end of each day	–
D.htg	X	X	number	end of each day	–
D.hp	X	X	number	end of each day	–
D.dhw	X	X	number	end of each day	–
D.dhwBU	X	X	number	end of each day	–
D.fanC	X	X	number	end of each day	–
D.fanH	X	X	number	end of each day	–
D.fanV	X	X	number	end of each day	–
D.fan	X	X	number	end of each day	–

Name	Input?	Runtime?	Type	Variability	Description
D.aux	X	X	number	end of each day	–
D.proc	X	X	number	end of each day	–
D.lit	X	X	number	end of each day	–
D.rcp	X	X	number	end of each day	–
D.ext	X	X	number	end of each day	–
D.refr	X	X	number	end of each day	–
D.dish	X	X	number	end of each day	–
D.dry	X	X	number	end of each day	–
D.wash	X	X	number	end of each day	–
D.cook	X	X	number	end of each day	–
D.usr1	X	X	number	end of each day	–
D.usr2	X	X	number	end of each day	–
D.pv	X	X	number	end of each day	–
D.cost	X	X	number	end of each day	–
D.dmdCost	X	X	number	end of each day	–
D.dmd	X	X	number	end of each day	–
D.dmdShoy	X	X	unrecognized	end of each day	–
H.tot	X	X	number	end of each hour	–
H.clg	X	X	number	end of each hour	–
H.htg	X	X	number	end of each hour	–
H.hp	X	X	number	end of each hour	–
H.dhw	X	X	number	end of each hour	–
H.dhwBU	X	X	number	end of each hour	–
H.fanC	X	X	number	end of each hour	–
H.fanH	X	X	number	end of each hour	–
H.fanV	X	X	number	end of each hour	–
H.fan	X	X	number	end of each hour	–
H.aux	X	X	number	end of each hour	–
H.proc	X	X	number	end of each hour	–
H.lit	X	X	number	end of each hour	–
H.rcp	X	X	number	end of each hour	–
H.ext	X	X	number	end of each hour	–
H.refr	X	X	number	end of each hour	–
H.dish	X	X	number	end of each hour	–
H.dry	X	X	number	end of each hour	–
H.wash	X	X	number	end of each hour	–
H.cook	X	X	number	end of each hour	–
H.usr1	X	X	number	end of each hour	–
H.usr2	X	X	number	end of each hour	–
H.pv	X	X	number	end of each hour	–
H.cost	X	X	number	end of each hour	–
H.dmdCost	X	X	number	end of each hour	–
H.dmd	X	X	number	end of each hour	–
H.dmdShoy	X	X	unrecognized	end of each hour	–

6.34 @perimeter[1..]. (owner: zone)

Name	Input?	Runtime?	Type	Variability	Description
name	X	–	string	constant	–
prLen	X	–	number	input time	–

Name	Input?	Runtime?	Type	Variability	Description
prF2	X	–	number	input time	–
xi	X	–	integer number	run start time (of each phase, autoSize or simulate)	–

6.35 @PVArray[1..].

Name	Input?	Runtime?	Type	Variability	Description
name	X	X	string	constant	–
elecMtri	X	X	integer number	input time	meter for system electricity production
endUse	X	X	integer number	autosize and simulate phase start time	end use of energy. defataults to "PV"
dcCap	X	X	number	input time	system capacity/size (DC nameplate), kW
usePVWattsDLL	X	X	integer number	input time	use PVWatts DLL instead of CSE calculations
moduleType	X	X	unrecognized	input time	type of module (Standard, Premium, ThinFilm)
tempCoeff	X	X	number	input time	temperature coefficient, 1/F
covRefrInd	X	X	number	input time	refraction index for coating applied to cover
arrayType	X	X	unrecognized	input time	type of array (Fixed, FixedRoof, 1Axis, Backtracked, 2Axis)
tilt	X	X	number	hourly	Array tilt, radians (input as degrees)
azm	X	X	number	hourly	Array azimuth, radians (input as degrees)
grndRefl	X	X	number	hourly	ground reflectance
gcr	X	X	number	input time	ground coverage ratio (what fraction of the ground is covered by the array). 1.0 implies no spacing.
dcacRat	X	X	number	input time	DC to AC ratio
invEff	X	X	number	input time	inverter efficiency at rated power

Name	Input?	Runtime?	Type	Variability	Description
sysLoss	X	X	number	hourly	system losses
tCell	X	X	number	end of each hour	cell temperature, F
aoi	X	X	number	end of each hour	angle of incidence (radians)
panelTilt	X	X	number	end of each hour	tilt of pv panel (different from array tilt for tracking systems), radians
panelAzm	X	X	number	end of each hour	azimuth of pv panel (different from array tilt for tracking systems), radians
poa	X	X	number	end of each hour	plane of array incidence, Btu/h-ft2
poaT	X	X	number	end of each hour	transmitted plane of array incidence, Btu/h-ft2
dcOut	X	X	number	end of each hour	DC power output, Btu
acOut	X	X	number	end of each hour	AC power output, Btu
tauNorm	X	X	number	run start time (of each phase, autoSize or simulate)	transmittance at normal incidence
inoct	X	X	number	run start time (of each phase, autoSize or simulate)	installed nominal operating cell temperature, F
convRatio	X	X	number	run start time (of each phase, autoSize or simulate)	ratio of back convection to front convection
tGrndRatio	X	X	number	run start time (of each phase, autoSize or simulate)	ratio of ground-cell temperature diff. to air-cell temperature diff.
poaPv	X	X	number	end of each hour	previous timestep plane of array incidence, Btu/h-ft2
tCellPv	X	X	number	end of each hour	previous timestep cell temperature, F

6.36 @report[1..]. (owner: reportFile)

Name	Input?	Runtime?	Type	Variability	Description
name	X	–	string	constant	–
zi	X	–	integer number	input time	–
mtri	X	–	integer number	input time	–
ahi	X	–	integer number	input time	–
tui	X	–	integer number	input time	–
dhwMtri	X	–	integer number	input time	–
isExport	X	–	integer number	input time	–
rpTy	X	–	integer number	constant	–
rpFreq	X	–	integer number	constant	–
rpDayBeg	X	–	integer number	input time	–
rpDayEnd	X	–	integer number	input time	–
rpBtuSf	X	–	number	input time	–
rpCond	X	–	number	end of each subhour	–
rpTitle	X	–	string	input time	–
rpCpl	X	–	integer number	input time	–
rpHeader	X	–	unrecognized	input time	–
rpFooter	X	–	integer number	input time	–
coli	X	–	integer number	run start time (of each phase, autoSize or simulate)	–
nCol	X	–	integer number	run start time (of each phase, autoSize or simulate)	–
wid	X	–	integer number	run start time (of each phase, autoSize or simulate)	–
vrh	X	–	unrecognized	run start time (of each phase, autoSize or simulate)	–

6.37 @reportCol[1..]. (owner: report)

Name	Input?	Runtime?	Type	Variability	Description
name	X	X	string	constant	–
colHead	X	X	string	input time	–
colGap	X	X	integer number	input time	–
colWid	X	X	integer number	input time	–
colDec	X	X	integer number	input time	–
colJust	X	X	integer number	input time	–
colVal	X	X	un-probe-able	end of each subhour	–
nxColi	X	X	integer number	constant	–

6.38 @reportFile[1..].

Name	Input?	Runtime?	Type	Variability	Description
name	X	–	string	constant	–
fileName	X	–	string	input time	–

Name	Input?	Runtime?	Type	Variability	Description
fileStat	X	–	integer number	run start time (of each phase, autoSize or simulate)	–
pageFmt	X	–	integer number	input time	–
fileStatChecked	X	–	integer number	run start time (of each phase, autoSize or simulate)	–
overWrite	X	–	integer number	run start time (of each phase, autoSize or simulate)	–

6.39 @RSYS[1..].

Name	Input?	Runtime?	Type	Variability	Description
name	X	X	string	constant	–
type	X	X	unrecognized	input time	leaving air state at plenum
desc	X	X	string	input time	NOT including any aux heat
perfMap	X	X	integer number	input time	leaving air state at plenum for ASHP heating (else 0)
areaServed	X	X	number	run start time (of each phase, autoSize or simulate)	NOT including any DSE or supply duct losses
zonesServed	X	X	unrecognized	run start time (of each phase, autoSize or simulate)	–
elecMtri	X	X	integer number	input time	… at full capacity under current conditions
fuelMtri	X	X	integer number	input time	… at full cap + auxiliary (ASHP only, else unused)
parElec	X	X	number	hourly	if <0, use DUCTSEG else apply DSE
parFuel	X	X	number	hourly	heating

Name	Input?	Runtime?	Type	Variability	Description
fan.fanTy	X	X	unrecognized	run start time (of each phase, autoSize or simulate)	–
fan.vfDs	X	X	number	end of each subhour	–
fan.vfDs_As	X	X	number	run start time (of each phase, autoSize or simulate)	–
fan.vfDs_AsNov	X	X	number	run start time (of each phase, autoSize or simulate)	–
fan.vfMxF	X	X	number	run start time (of each phase, autoSize or simulate)	–
fan.press	X	X	number	run start time (of each phase, autoSize or simulate)	–
fan.eff	X	X	number	run start time (of each phase, autoSize or simulate)	–

Name	Input?	Runtime?	Type	Variability	Description
fan.shaftPwr	X	X	number	run start time (of each phase, autoSize or simulate)	–
fan.elecPwr	X	X	number	run start time (of each phase, autoSize or simulate)	–
fan.motTy	X	X	unrecognized	run start time (of each phase, autoSize or simulate)	–
fan.motEff	X	X	number	run start time (of each phase, autoSize or simulate)	–
fan.motPos	X	X	unrecognized	run start time (of each phase, autoSize or simulate)	–
fan.curvePy.k[0]	X	X	number	run start time (of each phase, autoSize or simulate)	–

Name	Input?	Runtime?	Type	Variability	Description
fan.curvePy.k[1]	X	X	number	run start time (of each phase, autoSize or simulate)	–
fan.curvePy.k[2]	X	X	number	run start time (of each phase, autoSize or simulate)	–
fan.curvePy.k[3]	X	X	number	run start time (of each phase, autoSize or simulate)	–
fan.curvePy.k[4]	X	X	number	run start time (of each phase, autoSize or simulate)	–
fan.curvePy.k[5]	X	X	number	run start time (of each phase, autoSize or simulate)	–
fan.mtri	X	X	integer number	run start time (of each phase, autoSize or simulate)	–

Name	Input?	Runtime?	Type	Variability	Description
fan.endUse	X	X	integer number	run start time (of each phase, autoSize or simulate)	–
fan.ausz	X	X	integer number	run start time (of each phase, autoSize or simulate)	–
fan.outPower	X	X	number	subhourly	–
fan.airPower	X	X	number	subhourly	–
fan.cMx	X	X	number	end of each subhour	–
fan.c	X	X	number	end of each subhour	–
fan.t	X	X	number	end of each subhour	–
fan.frOn	X	X	number	end of each subhour	–
fan.p	X	X	number	end of each subhour	–
fan.q	X	X	number	end of each subhour	–
fan.dT	X	X	number	end of each subhour	–
fan.qAround	X	X	number	end of each subhour	–
asRet.tdb	X	X	number	end of each subhour	–
asRet.w	X	X	number	end of each subhour	–
asIn.tdb	X	X	number	end of each subhour	–

Name	Input?	Runtime?	Type	Variability	Description
asIn.w	X	X	number	end of each subhour	–
twbIn	X	X	number	end of each subhour	design temperature difference (rise) across RSYS for heating
asOut.tdb	X	X	number	end of each subhour	–
asOut.w	X	X	number	end of each subhour	–
asOutAux.tdb	X	X	number	end of each subhour	–
asOutAux.w	X	X	number	end of each subhour	–
asSup.tdb	X	X	number	end of each subhour	–
asSup.w	X	X	number	end of each subhour	–
asSupAux.tdb	X	X	number	end of each subhour	–
asSupAux.w	X	X	number	end of each subhour	–
tSupLs	X	X	number	subhourly	target excess capacity factor for heating autosize
DSEH	X	X	number	hourly	working excess capacity factor for cooling autosize
DSEC	X	X	number	hourly	ensures sufficient capacity to meet load
isAuszH	X	X	unrecognized	run start time (of each phase, autoSize or simulate)	ditto cooling
isAuszC	X	X	unrecognized	run start time (of each phase, autoSize or simulate)	ASHP heating (all value net (including rated fan heat / power)

Name	Input?	Runtime?	Type	Variability	Description
tdDesH	X	X	number	run start time (of each phase, autoSize or simulate)	rated HSPF, Btuh/W
tdDesC	X	X	number	run start time (of each phase, autoSize or simulate)	COP at ODB=47 F
fxCap[0]	X	X	number	end of each subhour	–
fxCap[1]	X	X	number	end of each subhour	–
fxCapCDay	X	X	number	end of each hour	heating cycling degradation factor
fxCapHDay	X	X	number	end of each hour	ditto 35 F
fxCapHTarg	X	X	number	run start time (of each phase, autoSize or simulate)	ditto 17 F
fxCapHAsF	X	X	number	run start time (of each phase, autoSize or simulate)	ASHP constants [0]=non-defrost, [1]=defrost
fxCapCTarg	X	X	number	run start time (of each phase, autoSize or simulate)	input slope: inp(T) = inp17 + InpF*(T - 17)

Name	Input?	Runtime?	Type	Variability	Description
fxCapCAsF	X	X	number	run start time (of each phase, autoSize or simulate)	auxiliary heating capacity (NOT including fan heat), Btuh
fxCapAuxHTarg	X	X	number	autosize and simulate phase start time	rs_capAuxH as input (may be AUTOSIZE)
auszH.az_active	X	X	integer number	run start time (of each phase, autoSize or simulate)	–
auszH.az_a	X	X	number	end of each subhour	–
auszH.az_b	X	X	number	end of each subhour	–
auszH.ldPk	X	X	number	end of each subhour	–
auszH.ldPkAs	X	X	number	end of each day	–
auszH.ldPkAs1	X	X	number	end of each day	–
auszH.plrPk	X	X	number	end of each subhour	–
auszH.plrPkAs	X	X	number	end of each day	–
auszH.xPk	X	X	number	end of each subhour	–
auszH.xPkAs	X	X	number	end of each day	–

Name	Input?	Runtime?	Type	Variability	Description
auszC.az_active	X	X	integer number	run start time (of each phase, autoSize or simulate)	–
auszC.az_a	X	X	number	end of each subhour	–
auszC.az_b	X	X	number	end of each subhour	–
auszC.ldPk	X	X	number	end of each subhour	–
auszC.ldPkAs	X	X	number	end of each day	–
auszC.ldPkAs1	X	X	number	end of each day	–
auszC.plrPk	X	X	number	end of each subhour	–
auszC.plrPkAs	X	X	number	end of each day	–
auszC.xPk	X	X	number	end of each subhour	–
auszC.xPkAs	X	X	number	end of each day	–
HSPF	X	X	number	run start time (of each phase, autoSize or simulate)	–
cap47	X	X	number	end of each phase (autosize or simulate)	non-ASHP heating

Name	Input?	Runtime?	Type	Variability	Description
COP47	X	X	number	end of each phase (autosize or simulate)	heating system rated AFUE, 0 < AFUE <= 1
cap35	X	X	number	end of each phase (autosize or simulate)	–
COP35	X	X	number	end of each phase (autosize or simulate)	rated heating output (including fan), Btuh
cap17	X	X	number	end of each phase (autosize or simulate)	… as autoSized
COP17	X	X	number	end of each phase (autosize or simulate)	… raw autoSized w/o oversizing
CdH	X	X	number	end of each phase (autosize or simulate)	autoSize code ASSUMES x, x_As, x_AsNov together for access thru one ptr. cuprobe.cpp's name search also requires together.
inp47	X	X	number	end of each phase (autosize or simulate)	fan heat included in ASHP rated cap/COP/HSPF, Btuh
inp35	X	X	number	end of each phase (autosize or simulate)	(generally estimated from rs_fanHRtdC)

Name	Input?	Runtime?	Type	Variability	Description
inp17	X	X	number	end of each phase (autosize or simulate)	heating fan power, W/cfm
ASHPCapF[0]	X	X	number	run start time (of each phase, autoSize or simulate)	–
ASHPCapF[1]	X	X	number	run start time (of each phase, autoSize or simulate)	–
ASHPInpF[0]	X	X	number	run start time (of each phase, autoSize or simulate)	–
ASHPInpF[1]	X	X	number	run start time (of each phase, autoSize or simulate)	–
capAuxH	X	X	number	end of each phase (autosize or simulate)	current step heating capacity (including fan and ASHP defrost heat), Btuh
capAuxHInp	X	X	number	end of each phase (autosize or simulate)	current step defrost heating capacity, Btuh

Name	Input?	Runtime?	Type	Variability	Description
COPAuxH	X	X	number	autosize and simulate phase start time	0 if not ASHP or no defrost active
ASHPLockOutT	X	X	number	hourly	efficiency degradation due to cycling
AFUE	X	X	number	autosize and simulate phase start time	cooling AHRI rated SEER, Btuh/W
capH	X	X	number	end of each phase (autosize or simulate)	rated total cooling capacity at 95 F, Btuh TODO: decide on sign
capH_As	X	X	number	end of each phase (autosize or simulate)	… as autoSized
capH_AsNov	X	X	number	end of each phase (autosize or simulate)	… raw autoSized w/o oversizing
fanHRtdH	X	X	number	autosize and simulate phase start time	air flow ratio, cfm/ton (= cfm/(rs_cap95/12000))
fanPwrH	X	X	number	autosize and simulate phase start time	cooling fan operating electrical power, Btuh
fanHeatH	X	X	number	end of each phase (autosize or simulate)	used re both electricity use and air heat gain

Name	Input?	Runtime?	Type	Variability	Description
amfH	X	X	number	end of each phase (autosize or simulate)	constant even if capacity is altered during autosize
effHt	X	X	number	end of each subhour	Why: air flow is function of rated capacity
capHt	X	X	number	end of each subhour	cooling cycling degradation factor
capDefrostHt	X	X	number	end of each subhour	plenum entering air relnum, 0-1
PLF	X	X	number	end of each subhour	coil entering dry bulb, F (ditto)
SEER	X	X	number	autosize and simulate phase start time	refrigerant charge factor (default 1, 0.9 or 0.96 for CA compliance)
EER95	X	X	number	autosize and simulate phase start time	compressor sizing factor (default 1, 0.95 or 1 for CA compliance)
cap95	X	X	number	end of each phase (autosize or simulate)	fan heat included in rated rs_cap95, Btuh
cap95_As	X	X	number	end of each phase (autosize or simulate)	constant for rs_capCt calc
cap95_AsNov	X	X	number	end of each phase (autosize or simulate)	–

Name	Input?	Runtime?	Type	Variability	Description
vfPerTon	X	X	number	autosize and simulate phase start time	–
fanPwrC	X	X	number	autosize and simulate phase start time	conditions factor, capacity
fanHeatC	X	X	number	end of each phase (autosize or simulate)	conditions factor, SEER
fanDeltaTC	X	X	number	end of each phase (autosize or simulate)	–
amfC	X	X	number	end of each phase (autosize or simulate)	EER w/o fan power
CdC	X	X	number	end of each phase (autosize or simulate)	compressor EER, Btuh/W (temperature weighted mix of
rhInTest	X	X	number	end of each hour	rs_SEERnf and rs_EERnf)
rhIn	X	X	number	end of each subhour	temp adjusted compressor efficiency (= CEt in ACM)
twbCoilIn	X	X	number	end of each subhour	coil total cooling capacity at current conditions, Btuh (<0)
tdbCoilIn	X	X	number	end of each subhour	coil latent cooling capacity at current conditions, Btuh (<0)
wetCoil	X	X	unrecognized	end of each subhour	coil sensibgle cooling capacity at current conditions, Btuh (<0)

Name	Input?	Runtime?	Type	Variability	Description
SHR	X	X	number	end of each subhour	Central outside air vent (aka OAV)
fChg	X	X	number	autosize and simulate phase start time	OAV relief zone index
fSize	X	X	number	autosize and simulate phase start time	OAV inlet dry-bulb temp, F
fanHRtdC	X	X	number	autosize and simulate phase start time	default = from project weather data source (generally weather file)
capnfX	X	X	number	autosize and simulate phase start time	note: default varies subhourly but input expression is hourly
capAdjF	X	X	number	autosize and simulate phase start time	OAV temperature differential, F
SEERnfX	X	X	number	end of each phase (auto-size or simulate)	OAV design air flow rate, cfm actual air
EERnfX	X	X	number	end of each phase (auto-size or simulate)	OAV design fan power (based on rs_OAVVfDs), W/cfm
fCondCap	X	X	number	end of each subhour	–
fCondSEER	X	X	number	end of each subhour	OAV current air volume flow, cfm (set at beg of each day)

Name	Input?	Runtime?	Type	Variability	Description
fCondEER	X	X	number	end of each subhour	ditto fan power, Btuh
SEERnf	X	X	number	end of each subhour	DUCTSEG linkage
EERnf	X	X	number	end of each subhour	–
EERt	X	X	number	end of each subhour	idx of associated DUCTSEGs, 0 if none
effCt	X	X	number	end of each subhour	nonleak fraction = (1 - ds_leakF) [0] = sup, [1] = ret
capTotCt	X	X	number	end of each subhour	[0]=htg [1] = clg
capLatCt	X	X	number	end of each subhour	–
capSenCt	X	X	number	end of each subhour	0: use htg ducts, 1: use clg
OAVType	X	X	unrecognized	input time	outdoor dry-bulb temp at condensor or other outdoor components, F
OAVReliefZi	X	X	integer number	input time	default = from project weather data source (generally weather file)
OAVTdbInlet	X	X	number	subhourly	note: default varies subhourly but input expression is hourly
OAVTdiff	X	X	number	hourly	last step mode (rsmOFF, rsmHEAT, rsmCOOL, rsmOAV)
OAVAvfDs	X	X	number	input time	full-load (maximum) dry air mass flow rate, lbm/hr
OAVFanPwr	X	X	number	input time	= flow at blower / coil / furnace HX etc.
OAVAvfMinF	X	X	number	input time	set per rsMode from rs_amfH, rs_amfC, or OAV algorithm
avfOAV	X	X	number	daily	[0] = main source (compressor or burner)
fanHeatOAV	X	X	number	daily	[1] = main+aux, ASHP heating only else 0
amfOAV	X	X	number	daily	run fraction
tdbOut	X	X	number	subhourly	fan electricity input, Btuh (not kWh)
modeCtrl	X	X	unrecognized	hourly	–
mode	X	X	unrecognized	end of each subhour	RSYS
modeLs	X	X	unrecognized	subhourly	==
amf	X	X	number	end of each subhour	RSYSresult substruct for RSYSRES

Name	Input?	Runtime?	Type	Variability	Description
amfReq[0]	X	X	number	end of each subhour	–
amfReq[1]	X	X	number	end of each subhour	–
runF	X	X	number	end of each subhour	outdoor temp (for convience for reporting; same for all ah) (Top.tDbO, .wO)
runFAux	X	X	number	end of each subhour	return air (AH.tr,.wr)
outSen	X	X	number	end of each subhour	mixed air (.tmix,.wmix)
outLat	X	X	number	end of each subhour	fraction of time fan on if ahFanCycles, else 1.0
outFan	X	X	number	end of each subhour	flow (at supply fan) (.cmix)
outDefrost	X	X	number	end of each subhour	float members to add: qh is first, hrsOn is last. CAUTION: q's here are energy not power.
outAux	X	X	number	end of each subhour	net energy taken from outside air (possible future impl); fan heat energy
inPrimary	X	X	number	end of each subhour	unbalance, should be near 0: qh+qc+qO+qFan+qLoss+qLoad.
inFan	X	X	number	end of each subhour	heat and cool coil input energy, from meter or (probably) from plant
inDefrost	X	X	number	end of each subhour	heat and cool aux energy
inAux	X	X	number	end of each subhour	fans input energy

6.40 @RSYSRes[1..].

Name	Input?	Runtime?	Type	Variability	Description
name	–	X	string	constant	–
Y.n	–	X	unrecognized	end of run (of each phase, autoSize or simulate)	–
M.n	–	X	unrecognized	end of each month	–
D.n	–	X	unrecognized	end of each day	–
H.n	–	X	unrecognized	end of each hour	–
S.n	–	X	unrecognized	end of each subhour	–

6.41 @sgdist[1..]. (owner: window)

Name	Input?	Runtime?	Type	Variability	Description
name	X	–	string	constant	–
sgSide	X	–	integer number	input time	–
targTy	X	–	integer number	run start time (of each phase, autoSize or simulate)	–
targTi	X	–	integer number	input time	–
FSO	X	–	number	monthly-hourly	–
FSC	X	–	number	monthly-hourly	–

6.42 @shade[1..]. (owner: window)

Name	Input?	Runtime?	Type	Variability	Description
name	X	X	string	constant	–
wWidth	X	X	number	run start time (of each phase, autoSize or simulate)	–
wHeight	X	X	number	run start time (of each phase, autoSize or simulate)	–
ohDepth	X	X	number	monthly-hourly	–
ohDistUp	X	X	number	monthly-hourly	–
ohExL	X	X	number	monthly-hourly	–
ohExR	X	X	number	monthly-hourly	–
ohFlap	X	X	number	monthly-hourly	–
lfDepth	X	X	number	monthly-hourly	–
lfTopUp	X	X	number	monthly-hourly	–
lfDistL	X	X	number	monthly-hourly	–
lfBotUp	X	X	number	monthly-hourly	–
rfDepth	X	X	number	monthly-hourly	–
rfTopUp	X	X	number	monthly-hourly	–
rfDistR	X	X	number	monthly-hourly	–
rfBotUp	X	X	number	monthly-hourly	–

6.43 @surface[1..]. (owner: zone)

Name	Input?	Runtime?	Type	Variability	Description
name	X	–	string	constant	–
ty	X	–	integer number	input time	–
area	X	–	number	run start time (of each phase, autoSize or simulate)	–
azm	X	–	number	run start time (of each phase, autoSize or simulate)	–

Name	Input?	Runtime?	Type	Variability	Description
tilt	X	–	number	run start time (of each phase, autoSize or simulate)	–
dircos[0]	X	–	number	run start time (of each phase, autoSize or simulate)	–
dircos[1]	X	–	number	run start time (of each phase, autoSize or simulate)	–
dircos[2]	X	–	number	run start time (of each phase, autoSize or simulate)	–
depthBG	X	–	number	run start time (of each phase, autoSize or simulate)	–
model	X	–	integer number	input time	–
modelr	X	–	integer number	run start time (of each phase, autoSize or simulate)	–
lThkF	X	–	number	run start time (of each phase, autoSize or simulate)	–
gti	X	–	integer number	run start time (of each phase, autoSize or simulate)	–
sco	X	–	number	monthly-hourly	–
scc	X	–	number	monthly-hourly	–
sbcI.absSlr	X	–	number	monthly-hourly	–
sbcI.awAbsSlr	X	–	number	monthly-hourly	–
sbcI.epsLW	X	–	number	run start time (of each phase, autoSize or simulate)	–
sbcI.zi	X	–	integer number	run start time (of each phase, autoSize or simulate)	–
sbcI.F	X	–	number	run start time (of each phase, autoSize or simulate)	–

Name	Input?	Runtime?	Type	Variability	Description
sbcI.Fp	X	–	number	run start time (of each phase, autoSize or simulate)	–
sbcI.frRad	X	–	number	run start time (of each phase, autoSize or simulate)	–
sbcI.fSky	X	–	number	run start time (of each phase, autoSize or simulate)	–
sbcI.fAir	X	–	number	run start time (of each phase, autoSize or simulate)	–
sbcI.hcNat	X	–	number	end of each subhour	–
sbcI.hcFrc	X	–	number	end of each subhour	–
sbcI.hcMult	X	–	number	end of each subhour	–
sbcI.hxa	X	–	number	end of each subhour	–
sbcI.hxr	X	–	number	end of each subhour	–
sbcI.hxtot	X	–	number	end of each subhour	–
sbcI.uRat	X	–	number	end of each subhour	–
sbcI.fRat	X	–	number	end of each subhour	–
sbcI.cx	X	–	number	end of each subhour	–
sbcI.sgTarg.bm	X	–	number	end of each subhour	–
sbcI.sgTarg.df	X	–	number	end of each subhour	–
sbcI.sgTarg.tot	X	–	number	end of each subhour	–
sbcI.sg	X	–	number	end of each subhour	–
sbcI.tSrf	X	–	number	end of each subhour	–
sbcI.tSrfls	X	–	number	subhourly	–
sbcI.qrAbs	X	–	number	end of each subhour	–
sbcI.txa	X	–	number	end of each subhour	–
sbcI.txr	X	–	number	end of each subhour	–

Name	Input?	Runtime?	Type	Variability	Description
sbcI.txe	X	–	number	end of each subhour	–
sbcI.w	X	–	number	end of each subhour	–
sbcI.qSrf	X	–	number	end of each subhour	–
sbcI.pXS	X	–	unrecognized	run start time (of each phase, autoSize or simulate)	–
sbcI.si	X	–	unrecognized	run start time (of each phase, autoSize or simulate)	–
sbcI.fcWind	X	–	number	run start time (of each phase, autoSize or simulate)	–
sbcI.fcWind2	X	–	number	run start time (of each phase, autoSize or simulate)	–
sbcI.eta	X	–	number	end of each subhour	–
sbcI.widNom	X	–	number	run start time (of each phase, autoSize or simulate)	–
sbcI.lenNom	X	–	number	run start time (of each phase, autoSize or simulate)	–
sbcI.lenCharNat	X	–	number	run start time (of each phase, autoSize or simulate)	–
sbcI.lenEffWink	X	–	number	run start time (of each phase, autoSize or simulate)	–
sbcI.atvDeg	X	–	number	run start time (of each phase, autoSize or simulate)	–
sbcI.cosAtv	X	–	number	run start time (of each phase, autoSize or simulate)	–
sbcI.hcModel	X	–	unrecognized	run start time (of each phase, autoSize or simulate)	–

Name	Input?	Runtime?	Type	Variability	Description
sbcI.hcLChar	X	–	number	run start time (of each phase, autoSize or simulate)	–
sbcI.hcConst[0]	X	–	number	run start time (of each phase, autoSize or simulate)	–
sbcI.hcConst[1]	X	–	number	run start time (of each phase, autoSize or simulate)	–
sbcI.hcConst[2]	X	–	number	run start time (of each phase, autoSize or simulate)	–
sbcI.groundModel	X	–	unrecognized	run start time (of each phase, autoSize or simulate)	–
sbcI.cTaDbAvgYr	X	–	number	run start time (of each phase, autoSize or simulate)	–
sbcI.cTaDbAvg31	X	–	number	run start time (of each phase, autoSize or simulate)	–
sbcI.cTaDbAvg14	X	–	number	run start time (of each phase, autoSize or simulate)	–
sbcI.cTaDbAvg07	X	–	number	run start time (of each phase, autoSize or simulate)	–
sbcI.cTGrnd	X	–	number	run start time (of each phase, autoSize or simulate)	–
sbcI.rGrnd	X	–	number	run start time (of each phase, autoSize or simulate)	–
sbcI.rConGrnd	X	–	number	run start time (of each phase, autoSize or simulate)	–
sbcO.absSlr	X	–	number	monthly-hourly	–
sbcO.awAbsSlr	X	–	number	monthly-hourly	–

Name	Input?	Runtime?	Type	Variability	Description
sbcO.epsLW	X	–	number	run start time (of each phase, autoSize or simulate)	–
sbcO.zi	X	–	integer number	run start time (of each phase, autoSize or simulate)	–
sbcO.F	X	–	number	run start time (of each phase, autoSize or simulate)	–
sbcO.Fp	X	–	number	run start time (of each phase, autoSize or simulate)	–
sbcO.frRad	X	–	number	run start time (of each phase, autoSize or simulate)	–
sbcO.fSky	X	–	number	run start time (of each phase, autoSize or simulate)	–
sbcO.fAir	X	–	number	run start time (of each phase, autoSize or simulate)	–
sbcO.hcNat	X	–	number	end of each subhour	–
sbcO.hcFrc	X	–	number	end of each subhour	–
sbcO.hcMult	X	–	number	end of each subhour	–
sbcO.hxa	X	–	number	end of each subhour	–
sbcO.hxr	X	–	number	end of each subhour	–
sbcO.hxtot	X	–	number	end of each subhour	–
sbcO.uRat	X	–	number	end of each subhour	–
sbcO.fRat	X	–	number	end of each subhour	–
sbcO.cx	X	–	number	end of each subhour	–
sbcO.sgTarg.bm	X	–	number	end of each subhour	–
sbcO.sgTarg.df	X	–	number	end of each subhour	–
sbcO.sgTarg.tot	X	–	number	end of each subhour	–

Name	Input?	Runtime?	Type	Variability	Description
sbcO.sg	X	–	number	end of each subhour	–
sbcO.tSrf	X	–	number	end of each subhour	–
sbcO.tSrfls	X	–	number	subhourly	–
sbcO.qrAbs	X	–	number	end of each subhour	–
sbcO.txa	X	–	number	end of each subhour	–
sbcO.txr	X	–	number	end of each subhour	–
sbcO.txe	X	–	number	end of each subhour	–
sbcO.w	X	–	number	end of each subhour	–
sbcO.qSrf	X	–	number	end of each subhour	–
sbcO.pXS	X	–	unrecognized	run start time (of each phase, autoSize or simulate)	–
sbcO.si	X	–	unrecognized	run start time (of each phase, autoSize or simulate)	–
sbcO.fcWind	X	–	number	run start time (of each phase, autoSize or simulate)	–
sbcO.fcWind2	X	–	number	run start time (of each phase, autoSize or simulate)	–
sbcO.eta	X	–	number	end of each subhour	–
sbcO.widNom	X	–	number	run start time (of each phase, autoSize or simulate)	–
sbcO.lenNom	X	–	number	run start time (of each phase, autoSize or simulate)	–
sbcO.lenCharNat	X	–	number	run start time (of each phase, autoSize or simulate)	–
sbcO.lenEffWink	X	–	number	run start time (of each phase, autoSize or simulate)	–

Name	Input?	Runtime?	Type	Variability	Description
sbcO.atvDeg	X	–	number	run start time (of each phase, autoSize or simulate)	–
sbcO.cosAtv	X	–	number	run start time (of each phase, autoSize or simulate)	–
sbcO.hcModel	X	–	unrecognized	run start time (of each phase, autoSize or simulate)	–
sbcO.hcLChar	X	–	number	run start time (of each phase, autoSize or simulate)	–
sbcO.hcConst[0]	X	–	number	run start time (of each phase, autoSize or simulate)	–
sbcO.hcConst[1]	X	–	number	run start time (of each phase, autoSize or simulate)	–
sbcO.hcConst[2]	X	–	number	run start time (of each phase, autoSize or simulate)	–
sbcO.groundModel	X	–	unrecognized	run start time (of each phase, autoSize or simulate)	–
sbcO.cTaDbAvgYr	X	–	number	run start time (of each phase, autoSize or simulate)	–
sbcO.cTaDbAvg31	X	–	number	run start time (of each phase, autoSize or simulate)	–
sbcO.cTaDbAvg14	X	–	number	run start time (of each phase, autoSize or simulate)	–
sbcO.cTaDbAvg07	X	–	number	run start time (of each phase, autoSize or simulate)	–
sbcO.cTGrnd	X	–	number	run start time (of each phase, autoSize or simulate)	–

Name	Input?	Runtime?	Type	Variability	Description
sbcO.rGrnd	X	–	number	run start time (of each phase, autoSize or simulate)	–
sbcO.rConGrnd	X	–	number	run start time (of each phase, autoSize or simulate)	–
fenModel	X	–	unrecognized	input time	–
SHGC	X	–	number	input time	–
fMult	X	–	number	run start time (of each phase, autoSize or simulate)	–
UNFRC	X	–	number	input time	–
NGlz	X	–	integer number	input time	–
exShd	X	–	unrecognized	input time	–
inShd	X	–	unrecognized	input time	–
dirtLoss	X	–	number	run start time (of each phase, autoSize or simulate)	–
sfExCnd	X	–	integer number	run start time (of each phase, autoSize or simulate)	–
sfExT	X	–	number	subhourly	–
sfAdjZi	X	–	integer number	input time	–
uI	X	–	number	run start time (of each phase, autoSize or simulate)	–
uC	X	–	number	run start time (of each phase, autoSize or simulate)	–
uX	X	–	number	run start time (of each phase, autoSize or simulate)	–
Rf	X	–	number	run start time (of each phase, autoSize or simulate)	–
grndRefl	X	–	number	monthly-hourly	–
vfSkyDf	X	–	number	monthly-hourly	–
vfGrndDf	X	–	number	monthly-hourly	–
vfSkyLW	X	–	number	run start time (of each phase, autoSize or simulate)	–

Name	Input?	Runtime?	Type	Variability	Description
vfGrndLW	X	–	number	run start time (of each phase, autoSize or simulate)	–
uval	X	–	number	run start time (of each phase, autoSize or simulate)	–
UNom	X	–	number	run start time (of each phase, autoSize or simulate)	–
UANom	X	–	number	run start time (of each phase, autoSize or simulate)	–
rSrfNom[0]	X	–	number	run start time (of each phase, autoSize or simulate)	–
rSrfNom[1]	X	–	number	run start time (of each phase, autoSize or simulate)	–
hSrfNom[0]	X	–	number	run start time (of each phase, autoSize or simulate)	–
hSrfNom[1]	X	–	number	run start time (of each phase, autoSize or simulate)	–
cFctr	X	–	number	run start time (of each phase, autoSize or simulate)	–
iwshad	X	–	integer number	run start time (of each phase, autoSize or simulate)	–
msi	X	–	integer number	run start time (of each phase, autoSize or simulate)	–
tLrB[0]	X	–	number	end of each hour	–
tLrB[1]	X	–	number	end of each hour	–
tLrB[2]	X	–	number	end of each hour	–
tLrB[3]	X	–	number	end of each hour	–
tLrB[4]	X	–	number	end of each hour	–
tLrB[5]	X	–	number	end of each hour	–
tLrB[6]	X	–	number	end of each hour	–
tLrB[7]	X	–	number	end of each hour	–

Name	Input?	Runtime?	Type	Variability	Description
tLrB[8]	X	–	number	end of each hour	–
tLrB[9]	X	–	number	end of each hour	–
nsgdist	X	–	integer number	run start time (of each phase, autoSize or simulate)	–
sgdist[0].targTy	X	–	integer number	run start time (of each phase, autoSize or simulate)	–
sgdist[0].targTi	X	–	integer number	run start time (of each phase, autoSize or simulate)	–
sgdist[0].FSO	X	–	number	monthly-hourly	–
sgdist[0].FSC	X	–	number	monthly-hourly	–
sgdist[1].targTy	X	–	integer number	run start time (of each phase, autoSize or simulate)	–
sgdist[1].targTi	X	–	integer number	run start time (of each phase, autoSize or simulate)	–
sgdist[1].FSO	X	–	number	monthly-hourly	–
sgdist[1].FSC	X	–	number	monthly-hourly	–
sgdist[2].targTy	X	–	integer number	run start time (of each phase, autoSize or simulate)	–
sgdist[2].targTi	X	–	integer number	run start time (of each phase, autoSize or simulate)	–
sgdist[2].FSO	X	–	number	monthly-hourly	–
sgdist[2].FSC	X	–	number	monthly-hourly	–
sgdist[3].targTy	X	–	integer number	run start time (of each phase, autoSize or simulate)	–
sgdist[3].targTi	X	–	integer number	run start time (of each phase, autoSize or simulate)	–
sgdist[3].FSO	X	–	number	monthly-hourly	–
sgdist[3].FSC	X	–	number	monthly-hourly	–
sgdist[4].targTy	X	–	integer number	run start time (of each phase, autoSize or simulate)	–

Name	Input?	Runtime?	Type	Variability	Description
sgdist[4].targTi	X	–	integer number	run start time (of each phase, autoSize or simulate)	–
sgdist[4].FSO	X	–	number	monthly-hourly	–
sgdist[4].FSC	X	–	number	monthly-hourly	–
sgdist[5].targTy	X	–	integer number	run start time (of each phase, autoSize or simulate)	–
sgdist[5].targTi	X	–	integer number	run start time (of each phase, autoSize or simulate)	–
sgdist[5].FSO	X	–	number	monthly-hourly	–
sgdist[5].FSC	X	–	number	monthly-hourly	–
sgdist[6].targTy	X	–	integer number	run start time (of each phase, autoSize or simulate)	–
sgdist[6].targTi	X	–	integer number	run start time (of each phase, autoSize or simulate)	–
sgdist[6].FSO	X	–	number	monthly-hourly	–
sgdist[6].FSC	X	–	number	monthly-hourly	–
sgdist[7].targTy	X	–	integer number	run start time (of each phase, autoSize or simulate)	–
sgdist[7].targTi	X	–	integer number	run start time (of each phase, autoSize or simulate)	–
sgdist[7].FSO	X	–	number	monthly-hourly	–
sgdist[7].FSC	X	–	number	monthly-hourly	–
sfClass	X	–	unrecognized	input time	–
sfArea	X	–	number	input time	–
sfU	X	–	number	input time	–
sfCon	X	–	integer number	input time	–
sfTy	X	–	integer number	constant	–
width	X	–	number	input time	–
height	X	–	number	input time	–
mult	X	–	number	input time	–
xi	X	–	integer number	run start time (of each phase, autoSize or simulate)	–
msi	X	–	integer number	run start time (of each phase, autoSize or simulate)	–

6.44 @terminal[1..]. (owner: zone)

Name	Input?	Runtime?	Type	Variability	Description
name	X	X	string	constant	–
tuVfMxHC	X	X	unrecognized	autosize and simulate phase start time	–
tuOversize	X	X	number	autosize and simulate phase start time	–
asHcSame	X	X	integer number	run start time (of each phase, autoSize or simulate)	–
asKVol	X	X	integer number	run start time (of each phase, autoSize or simulate)	–
hcAs.az_active	X	X	integer number	run start time (of each phase, autoSize or simulate)	–
hcAs.az_a	X	X	number	end of each subhour	–
hcAs.az_b	X	X	number	end of each subhour	–
hcAs.ldPk	X	X	number	end of each subhour	–
hcAs.ldPkAs	X	X	number	end of each day	–
hcAs.ldPkAs1	X	X	number	end of each day	–
hcAs.plrPk	X	X	number	end of each subhour	–
hcAs.plrPkAs	X	X	number	end of each day	–
hcAs.xPk	X	X	number	end of each subhour	–
hcAs.xPkAs	X	X	number	end of each day	–
vhAs.az_active	X	X	integer number	run start time (of each phase, autoSize or simulate)	–
vhAs.az_a	X	X	number	end of each subhour	–
vhAs.az_b	X	X	number	end of each subhour	–
vhAs.ldPk	X	X	number	end of each subhour	–
vhAs.ldPkAs	X	X	number	end of each day	–
vhAs.ldPkAs1	X	X	number	end of each day	–
vhAs.plrPk	X	X	number	end of each subhour	–
vhAs.plrPkAs	X	X	number	end of each day	–
vhAs.xPk	X	X	number	end of each subhour	–

Name	Input?	Runtime?	Type	Variability	Description
vhAs.xPkAs	X	X	number	end of each day	–
vcAs.az_active	X	X	integer number	run start time (of each phase, autoSize or simulate)	–
vcAs.az_a	X	X	number	end of each subhour	–
vcAs.az_b	X	X	number	end of each subhour	–
vcAs.ldPk	X	X	number	end of each subhour	–
vcAs.ldPkAs	X	X	number	end of each day	–
vcAs.ldPkAs1	X	X	number	end of each day	–
vcAs.plrPk	X	X	number	end of each subhour	–
vcAs.plrPkAs	X	X	number	end of each day	–
vcAs.xPk	X	X	number	end of each subhour	–
vcAs.xPkAs	X	X	number	end of each day	–
qhPk	X	X	number	end of each subhour	–
qcPk	X	X	number	end of each subhour	–
qhPkAs	X	X	number	end of each subhour	–
qcPkAs	X	X	number	end of each subhour	–
bVfMn	X	X	number	end of each subhour	–
bVfMxH	X	X	number	end of each subhour	–
bVfMxC	X	X	number	end of each subhour	–
dtLoHSh	X	X	integer number	end of each subhour	–
dtLoCSh	X	X	integer number	end of each subhour	–
aDtLoHSh	X	X	integer number	end of each subhour	–
aDtLoCSh	X	X	integer number	end of each subhour	–
aDtLoTem	X	X	integer number	end of each subhour	–
dtLoH	X	X	integer number	end of each subhour	–
dtLoC	X	X	integer number	end of each subhour	–
dtLoHAs	X	X	integer number	end of each day	–
dtLoCAs	X	X	integer number	end of each subhour	–
tuTLh	X	X	number	hourly	–
tuQMnLh	X	X	number	hourly	–

Name	Input?	Runtime?	Type	Variability	Description
tuQMxLh	X	X	number	hourly	–
tuPriLh	X	X	integer number	autosize and simulate phase start time	–
tuLhNeedsFlow	X	X	integer number	autosize and simulate phase start time	–
tuhc.coilTy	X	X	unrecognized	run start time (of each phase, autoSize or simulate)	–
tuhc.sched	X	X	unrecognized	hourly	–
tuhc.captRat	X	X	number	end of each subhour	–
tuhc.captRat_As	X	X	number	autosize and simulate phase start time	–
tuhc.captRat_AsNov	X	X	number	autosize and simulate phase start time	–
tuhc.bCaptRat	X	X	number	end of each subhour	–
tuhc.eirRat	X	X	number	hourly	–
tuhc.mtri	X	X	integer number	autosize and simulate phase start time	–
tuhc.auxOn	X	X	number	hourly	–
tuhc.auxOnMtri	X	X	integer number	autosize and simulate phase start time	–
tuhc.auxOff	X	X	number	hourly	–
tuhc.auxOffMtri	X	X	integer number	autosize and simulate phase start time	–
tuhc.auxOnAtall	X	X	number	hourly	–
tuhc.auxOnAtallMtri	X	X	integer number	autosize and simulate phase start time	–
tuhc.auxFullOff	X	X	number	hourly	–
tuhc.auxFullOffMtri	X	X	integer number	autosize and simulate phase start time	–
tuhc.q	X	X	number	end of each subhour	–
tuhc.qPr	X	X	number	end of each subhour	–
tuhc.p	X	X	number	end of each subhour	–
tuhc.plr	X	X	number	end of each subhour	–
tuhc.plrAv	X	X	number	end of each subhour	–

Name	Input?	Runtime?	Type	Variability	Description
tuhc.eir	X	X	number	end of each subhour	–
tuhc.pAuxOn	X	X	number	end of each subhour	–
tuhc.pAuxOff	X	X	number	end of each subhour	–
tuhc.pAuxOnAtall	X	X	number	end of each subhour	–
tuhc.pAuxFullOff	X	X	number	end of each subhour	–
tuhc.effRat	X	X	number	autosize and simulate phase start time	–
tuhc.pyEi.k[0]	X	X	number	autosize and simulate phase start time	–
tuhc.pyEi.k[1]	X	X	number	autosize and simulate phase start time	–
tuhc.pyEi.k[2]	X	X	number	autosize and simulate phase start time	–
tuhc.pyEi.k[3]	X	X	number	autosize and simulate phase start time	–
tuhc.pyEi.k[4]	X	X	number	autosize and simulate phase start time	–
tuhc.stackEffect	X	X	number	hourly	–
tuhc.hpi	X	X	integer number	autosize and simulate phase start time	–
tuhc.nxTu4hp	X	X	integer number	run start time (of each phase, autoSize or simulate)	–
tuhc.nxAh4hp	X	X	integer number	run start time (of each phase, autoSize or simulate)	–
tuhc.flueLoss	X	X	number	end of each subhour	–
tuhc.qWant	X	X	number	end of each subhour	–
tuTH	X	X	number	hourly	–
tuTC	X	X	number	hourly	–
tuVfMn	X	X	number	end of each subhour	–
tuVfMn_As	X	X	number	autosize and simulate phase start time	–

Name	Input?	Runtime?	Type	Variability	Description
tuVfMn_AsNov	X	X	number	autosize and simulate phase start time	–
ai	X	X	integer number	input time	–
tuVfMxH	X	X	number	end of each subhour	–
tuVfMxH_As	X	X	number	autosize and simulate phase start time	–
tuVfMxH_AsNov	X	X	number	autosize and simulate phase start time	–
tuVfMxC	X	X	number	end of each subhour	–
tuVfMxC_As	X	X	number	autosize and simulate phase start time	–
tuVfMxC_AsNov	X	X	number	autosize and simulate phase start time	–
tuVfDs	X	X	number	run start time (of each phase, autoSize or simulate)	–
tuPriH	X	X	integer number	autosize and simulate phase start time	–
tuPriC	X	X	integer number	autosize and simulate phase start time	–
tuSRLeak	X	X	number	autosize and simulate phase start time	–
tuSRLoss	X	X	number	run start time (of each phase, autoSize or simulate)	–
tfanSch	X	X	unrecognized	run start time (of each phase, autoSize or simulate)	–
tfanOffLeak	X	X	number	run start time (of each phase, autoSize or simulate)	–
tfan.fanTy	X	X	unrecognized	autosize and simulate phase start time	–
tfan.vfDs	X	X	number	end of each subhour	–

Name	Input?	Runtime?	Type	Variability	Description
tfan.vfDs_As	X	X	number	autosize and simulate phase start time	–
tfan.vfDs_AsNov	X	X	number	autosize and simulate phase start time	–
tfan.vfMxF	X	X	number	autosize and simulate phase start time	–
tfan.press	X	X	number	run start time (of each phase, autoSize or simulate)	–
tfan.eff	X	X	number	run start time (of each phase, autoSize or simulate)	–
tfan.shaftPwr	X	X	number	run start time (of each phase, autoSize or simulate)	–
tfan.elecPwr	X	X	number	run start time (of each phase, autoSize or simulate)	–
tfan.motTy	X	X	unrecognized	run start time (of each phase, autoSize or simulate)	–
tfan.motEff	X	X	number	autosize and simulate phase start time	–
tfan.motPos	X	X	unrecognized	autosize and simulate phase start time	–
tfan.curvePy.k[0]	X	X	number	autosize and simulate phase start time	–
tfan.curvePy.k[1]	X	X	number	autosize and simulate phase start time	–
tfan.curvePy.k[2]	X	X	number	autosize and simulate phase start time	–
tfan.curvePy.k[3]	X	X	number	autosize and simulate phase start time	–
tfan.curvePy.k[4]	X	X	number	autosize and simulate phase start time	–

Name	Input?	Runtime?	Type	Variability	Description
tfan.curvePy.k[5]	X	X	number	autosize and simulate phase start time	–
tfan.mtri	X	X	integer number	input time	–
tfan.endUse	X	X	integer number	autosize and simulate phase start time	–
tfan.ausz	X	X	integer number	run start time (of each phase, autoSize or simulate)	–
tfan.outPower	X	X	number	subhourly	–
tfan.airPower	X	X	number	subhourly	–
tfan.cMx	X	X	number	end of each subhour	–
tfan.c	X	X	number	end of each subhour	–
tfan.t	X	X	number	end of each subhour	–
tfan.frOn	X	X	number	end of each subhour	–
tfan.p	X	X	number	end of each subhour	–
tfan.q	X	X	number	end of each subhour	–
tfan.dT	X	X	number	end of each subhour	–
tfan.qAround	X	X	number	end of each subhour	–
nxTu4z	X	X	integer number	run start time (of each phase, autoSize or simulate)	–
nxTu4a	X	X	integer number	run start time (of each phase, autoSize or simulate)	–
xiLh	X	X	integer number	run start time (of each phase, autoSize or simulate)	–
xiArH	X	X	integer number	run start time (of each phase, autoSize or simulate)	–
xiArC	X	X	integer number	run start time (of each phase, autoSize or simulate)	–

Name	Input?	Runtime?	Type	Variability	Description
cmLh	X	X	unrecognized	run start time (of each phase, autoSize or simulate)	–
cmAr	X	X	unrecognized	run start time (of each phase, autoSize or simulate)	–
ctrlsAi	X	X	integer number	run start time (of each phase, autoSize or simulate)	–
wantMd	X	X	unrecognized	end of each subhour	–
lhMn	X	X	number	end of each subhour	–
lhMx	X	X	number	end of each subhour	–
lhMxMx	X	X	number	end of each subhour	–
cMn	X	X	number	end of each subhour	–
cMxH	X	X	number	end of each subhour	–
cMxC	X	X	number	end of each subhour	–
useLh	X	X	unrecognized	end of each subhour	–
useAr	X	X	unrecognized	end of each subhour	–
qLhWant	X	X	number	end of each subhour	–
cv	X	X	number	end of each subhour	–
cz	X	X	number	end of each subhour	–
aCv	X	X	number	end of each subhour	–
tfanRunning	X	X	integer number	end of each subhour	–
tfanBkC	X	X	number	end of each subhour	–

6.45 @top.

Top level fields

Name	Input?	Runtime?	Type	Variability	Description
name	X	X	string	constant	Name of the object.

Name	Input?	Runtime?	Type	Variability	Description
bAutoSizeCmd	X	X	integer number	input time	Non-0 if any AUTOSIZE commands seen in input
chAutoSize	X	X	integer number	run start time (of each phase, autoSize or simulate)	Whether to do autosizing, default per bAutoSizeCmd
chSimulate	X	X	integer number	input time	Whether to do main simulation, default is TRUE; input FALSE for autosizing only.
begDay	X	X	integer number	input time	First 1-based Julian day of year of run.
endDay	X	X	integer number	input time	Last 1-based Julian day of year of run, inclusive
nDays	X	X	integer number	run start time (of each phase, autoSize or simulate)	Number of days in run
jan1DoW	X	X	integer number	input time	January 1 day of week: sun=1
year	X	X	integer number	run start time (of each phase, autoSize or simulate)	Generic non-leap year. -1 = Jan 1 is Monday .. -7 Jan 1 is Sunday
wuDays	X	X	integer number	input time	Number of warm-up days
nSubSteps	X	X	integer number	input time	Subhours per hour, determines subhour duration
skipDayStart	X	X	integer number	input time	Number of days to skip at beginning of year (not beginning of run), default 0.
skipDayStep	X	X	integer number	input time	Number of days in each step through year, default 1
wfName	X	X	string	autosize and simulate phase start time	Weather file name string.
elevation	X	X	number	run start time (of each phase, autoSize or simulate)	Site elevation in feet (for determining air density)

Name	Input?	Runtime?	Type	Variability	Description
refTemp	X	X	number	autosize and simulate phase start time	Temperature for computing the humidity ratio (w). Used in air-density calculations, default 70 F.
refRH	X	X	number	autosize and simulate phase start time	Relative humidity (as a fraction); default 0.6 (60%).
grndRefl	X	X	number	monthly-hourly	Ground surface reflectivity, regarding solar gain.
soilDiff	X	X	number	input time	Local soil diffusivity in ft^2/hr. Relates to annual deep ground temperature cycle estimation.
tol	X	X	number	input time	(Relative) tolerance used in many HVAC calculations, default 0.001 or as changed.
humTolF	X	X	number	input time	Humidity ratio (w) change to consider as important as 1F temp regarding convergedness.
ebTolMon	X	X	number	input time	Monthly tolerance.
ebTolDay	X	X	number	input time	Daily tolerance.
ebTolHour	X	X	number	input time	Hourly tolerance.
ebTolSubhr	X	X	number	input time	Subhourly tolerance.
AWTrigT	X	X	number	input time	ASHWAT: inside or outside environmental temperature, F (default = 1)
AWTrigSlr	X	X	number	input time	ASHWAT: incident solar, fraction (default = 0.05)

Name	Input?	Runtime?	Type	Variability	Description
AWTrigH	X	X	number	input time	ASHWAT: total surface coefficient (conv+rad), fraction (default = 0.1)
ANTolAbs	X	X	number	input time	AirNet: absolute tolerance, lbm/sec, dflt=0.00125 (about 1 cfm).
ANTolRel	X	X	number	input time	AirNet: relative tolerance, dflt = .0001.
bldgAzm	X	X	number	input time	Angle to add to all zone/surface azimuths
skyModel	X	X	integer number	input time	Sky model: C__..__ISO or _ANISO.
skyModelLW	X	X	unrecognized	input time	Long-wave sky model.
dhwModel	X	X	unrecognized	input time	Runtime DHW model selection.
humMeth	X	X	unrecognized	input time	Humidity calculation method: Rob (w = wa/wb) or Phil (central difference)
dflExH	X	X	number	input time	Default external (air film) conductivity.
workDayMask	X	X	integer number	input time	Mask with bits set for “work” days, clear for “non-work” days. Default Mon..Fri. Bits: Sun=1, Mon=2, Tu=4, W=8, Th=16, F=32, Sat=64, holidays=128, heatDsn-Day=256, coolDsnDays=512.
DT	X	X	integer number	input time	YES (default) to enable daylight saving time

Name	Input?	Runtime?	Type	Variability	Description
DTBegDay	X	X	integer number	run start time (of each phase, autoSize or simulate)	Daylight saving start day, 1-365, default 1st Sun (Sun after 1st Sat?) in April.
DTEndDay	X	X	integer number	run start time (of each phase, autoSize or simulate)	Daylight saving end day, 1-365. Default is last Sunday in October
windSpeedMin	X	X	number	input time	Minimum wind speed in miles-per-hour (default = 0.5)
windF	X	X	number	input time	Wind factor (default=1)
terrainClass	X	X	integer number	input time	Terrain class (1-5) for wind speed adjustment.
radBeamF	X	X	number	input time	Beam radiation factor. appl sees ANISO(<fileVal>) * BeamRadFactor.
radDiffF	X	X	number	input time	Diffuse variant of BeamRadFactor.
ventAvail	X	X	unrecognized	hourly	All-zone ventilation availability (default = WholeHouse)
verbose	X	X	integer number	autosize and simulate phase start time	Screen messages: autosizing: 0 none, 1 some, 2-5 more
dbgPrintMask	X	X	number	hourly	Debug print mask, controls printing. Schedulable via standard capabilities.
dbgPrintMaskC	X	X	number	input time	Constant portion (value known during setup) of debug print mask.
auszTol	X	X	number	input time	Autosizing result tolerance, default: 0.005

Name	Input?	Runtime?	Type	Variability	Description
heatDsTDbO	X	X	number	hourly	Heat design outdoor temperature. Default per the weather file header.
heatDsTWbO	X	X	number	hourly	Heating design outdoor wetbulb temperature. Default: 70% RH @ heatDsTDbO.
coolDsMo[0]	X	X	integer number	input time	Cooling design month(s) 1-12 + 0 terminator. Default per ET1 weather file header.
coolDsMo[1]	X	X	integer number	input time	As per coolDsMo[0]
coolDsMo[2]	X	X	integer number	input time	As per coolDsMo[0]
coolDsMo[3]	X	X	integer number	input time	As per coolDsMo[0]
coolDsMo[4]	X	X	integer number	input time	As per coolDsMo[0]
coolDsMo[5]	X	X	integer number	input time	As per coolDsMo[0]
coolDsMo[6]	X	X	integer number	input time	As per coolDsMo[0]
coolDsMo[7]	X	X	integer number	input time	As per coolDsMo[0]
coolDsMo[8]	X	X	integer number	input time	As per coolDsMo[0]
coolDsMo[9]	X	X	integer number	input time	As per coolDsMo[0]
coolDsMo[10]	X	X	integer number	input time	As per coolDsMo[0]
coolDsMo[11]	X	X	integer number	input time	As per coolDsMo[0]
coolDsMo[12]	X	X	integer number	input time	As per coolDsMo[0]
coolDsDay[0]	X	X	integer number	input time	–
coolDsDay[1]	X	X	integer number	input time	–
coolDsDay[2]	X	X	integer number	input time	–
coolDsDay[3]	X	X	integer number	input time	–
coolDsDay[4]	X	X	integer number	input time	–
coolDsDay[5]	X	X	integer number	input time	–
coolDsDay[6]	X	X	integer number	input time	–
coolDsDay[7]	X	X	integer number	input time	–
coolDsDay[8]	X	X	integer number	input time	–
coolDsDay[9]	X	X	integer number	input time	–

Name	Input?	Runtime?	Type	Variability	Description
coolDsDay[10]	X	X	integer number	input time	–
coolDsDay[11]	X	X	integer number	input time	–
coolDsDay[12]	X	X	integer number	input time	–
exePath	X	X	string	Run start time (of each phase, autoSize or simulate).	–
exeInfo	X	X	string	run start time (of each phase, autoSize or simulate)	–
exeCodeSize	X	X	unrecognized	run start time (of each phase, autoSize or simulate)	–
progVersion	X	X	string	run start time (of each phase, autoSize or simulate)	–
HPWHVersion	X	X	string	run start time (of each phase, autoSize or simulate)	–
runSerial	X	X	integer number	input time	–
runTitle	X	X	string	input time	–
runDateTime	X	X	string	run start time (of each phase, autoSize or simulate)	–
brs	X	X	integer number	run start time (of each phase, autoSize or simulate)	–
brHrly	X	X	integer number	run start time (of each phase, autoSize or simulate)	–
brFileName	X	X	string	input time	–
brMem	X	X	integer number	run start time (of each phase, autoSize or simulate)	Put binary results in Windows global memory and return handles; do not write file.
brDiscardable	X	X	integer number	run start time (of each phase, autoSize or simulate)	Put binary results in discardable memory as well as file, return handles. Overrides brMem.

Name	Input?	Runtime?	Type	Variability	Description
repHdrL	X	X	string	input time	User-specified text for left end of report header line.
repHdrR	X	X	string	input time	User-specified text for right end of report header line.
repCpl	X	X	integer number	input time	Report characters per line.
repLpp	X	X	integer number	input time	Total number of lines per page (paper size).
repTopM	X	X	integer number	input time	Top margin in lines; Number of newlines written above header.
repBotM	X	X	integer number	input time	Bottom margin in lines; not actually output.
repTestPfx	X	X	string	input time	Prefix pre-pended to e.g. footer lines regarding hiding lines and/or automated testing. “” for normal runs, “!” for 3-2010 test framework.
latitude	X	X	number	run start time (of each phase, autoSize or simulate)	Degrees north.
longitude	X	X	number	run start time (of each phase, autoSize or simulate)	Degress west.
timeZone	X	X	number	run start time (of each phase, autoSize or simulate)	Hours west (fraction ok).
presAtm	X	X	number	run start time (of each phase, autoSize or simulate)	Nominal atmospheric pressure at Top.elevation (in Hg). Constant for simulation.

Name	Input?	Runtime?	Type	Variability	Description
refW	X	X	number	run start time (of each phase, autoSize or simulate)	Humidity ratio for refTemp, refRH (ratio).
refWX	X	X	number	run start time (of each phase, autoSize or simulate)	1 / (1.0 + refW).
airSH	X	X	number	run start time (of each phase, autoSize or simulate)	Air specific heat (Btu/lbDryAir-F) @ refW.
airVK	X	X	number	run start time (of each phase, autoSize or simulate)	Specific volume per temperature (ft^3/lb-F): multiply by absolute temperature.
airRhoK	X	X	number	run start time (of each phase, autoSize or simulate)	density * temp (lb-F/ft^3): divide by absolute temperature to get density.
airVshK	X	X	number	run start time (of each phase, autoSize or simulate)	Volumetric specific heat / temperature (Btu/ft3-F): divide by absolute temperature for heat capacity per ft^3
airXK	X	X	number	run start time (of each phase, autoSize or simulate)	Divide by absolute temperature for specific heat of flow (Btuh/cfm-F)
hConvF	X	X	number	run start time (of each phase, autoSize or simulate)	Convective coefficient pressure modification factor.
nDesDays	X	X	unrecognized	run start time (of each phase, autoSize or simulate)	Number of design days: 1 for heating + number of non-0 coolDsMo's.
auszSmTol	X	X	number	run start time (of each phase, autoSize or simulate)	–

Name	Input?	Runtime?	Type	Variability	Description
auszTol2	X	X	number	run start time (of each phase, autoSize or simulate)	–
auszHiTol2	X	X	number	run start time (of each phase, autoSize or simulate)	–
vrSum	X	X	unrecognized	run start time (of each phase, autoSize or simulate)	–
dvriY	X	X	integer number	daily	–
dvriM	X	X	integer number	daily	–
dvriD	X	X	integer number	daily	–
dvriH	X	X	integer number	daily	–
dvriHS	X	X	integer number	daily	–
dvriS	X	X	integer number	daily	–
hrxFlg	X	X	integer number	daily	–
shrxFlg	X	X	integer number	daily	–
tmrInput	X	X	number	end of each day	–
tmrAusz	X	X	number	end of each day	–
tmrRun	X	X	number	end of each day	–
tmrTotal	X	X	number	end of each day	–
tmrAirNet	X	X	number	end of each day	–
tmrAWTot	X	X	number	end of each day	–
tmrAWCalc	X	X	number	end of each day	–
tmrCond	X	X	number	end of each day	–
tmrBC	X	X	number	end of each day	–
tmrZone	X	X	number	end of each day	–
subhrDur	X	X	number	subhourly	–
nSubhrTicks	X	X	integer number	run start time (of each phase, autoSize or simulate)	–
subhrTickDur	X	X	number	run start time (of each phase, autoSize or simulate)	–
subhrWSCount	X	X	unrecognized	run start time (of each phase, autoSize or simulate)	–
monStr	X	X	string	monthly	–
dateStr	X	X	string	daily	–
date	X	X	un-probe-able	daily	–
jDay	X	X	integer number	daily	–
xJDay	X	X	integer number	daily	–
skipDay	X	X	integer number	daily	–
iHr	X	X	integer number	hourly	–
iSubhr	X	X	integer number	subhourly	–
shoy	X	X	unrecognized	subhourly	–

Name	Input?	Runtime?	Type	Variability	Description
isDT	X	X	integer number	hourly	–
iHrST	X	X	integer number	hourly	–
jDayST	X	X	integer number	hourly	–
autoSizing	X	X	integer number	autosize and simulate phase start time	–
pass1	X	X	integer number	daily	–
pass1A	X	X	integer number	daily	–
pass1B	X	X	integer number	daily	–
pass2	X	X	integer number	daily	–
sizing	X	X	integer number	daily	–
dsDayI	X	X	unrecognized	daily	–
dsDay	X	X	integer number	daily	–
auszMon	X	X	integer number	daily	–
ivl	X	X	integer number	subhourly	–
isBegOf	X	X	integer number	subhourly	–
isEndOf	X	X	integer number	subhourly	–
isBegRun	X	X	integer number	subhourly	–
isBegMainSim	X	X	integer number	subhourly	–
isFirstMon	X	X	integer number	monthly	–
isLastDay	X	X	integer number	daily	–
isLastWarmupDay	X	X	integer number	daily	–
isBegHour	X	X	integer number	subhourly	–
isEndHour	X	X	integer number	subhourly	–
isBegDay	X	X	integer number	hourly	–
isEndDay	X	X	integer number	hourly	–
isBegMonth	X	X	integer number	daily	–
isEndMonth	X	X	integer number	daily	–
isSolarCalcDay	X	X	integer number	daily	–
isWarmup	X	X	integer number	daily	–
dowh	X	X	integer number	daily	–
isHoliday	X	X	integer number	daily	–
isHoliTrue	X	X	integer number	daily	–
isWeHol	X	X	integer number	daily	–
isWeekend	X	X	integer number	daily	–
isBegWeek	X	X	integer number	daily	–
isWeekday	X	X	integer number	daily	–
isWorkDay	X	X	integer number	daily	–
isNonWorkDay	X	X	integer number	daily	–
isBegWorkWeek	X	X	integer number	daily	–
notDone	X	X	integer number	daily	–
dsDayNIt	X	X	unrecognized	daily	–
radBeamHrAv	X	X	number	hourly	–
radBeamPvHrAv	X	X	number	hourly	–
radBeamNxHrAv	X	X	number	hourly	–
radBeamShAv	X	X	number	subhourly	–
radBeamShSpare	X	X	number	constant	–
radDiffHrAv	X	X	number	hourly	–
radDiffPvHrAv	X	X	number	hourly	–
radDiffNxHrAv	X	X	number	hourly	–
radDiffShAv	X	X	number	subhourly	–
radDiffShSpare	X	X	number	constant	–

Name	Input?	Runtime?	Type	Variability	Description
tDbOHr	X	X	number	hourly	–
tDbOPvHr	X	X	number	hourly	–
tDbOHrAv	X	X	number	hourly	–
tDbOSh	X	X	number	subhourly	–
tDbOPvSh	X	X	number	subhourly	–
tDbOShAv	X	X	number	subhourly	–
tWbOHr	X	X	number	hourly	–
tWbOPvHr	X	X	number	hourly	–
tWbOHrAv	X	X	number	hourly	–
tWbOSh	X	X	number	subhourly	–
tSkyHr	X	X	number	hourly	–
tSkyPvHr	X	X	number	hourly	–
tSkySh	X	X	number	subhourly	–
windSpeedHr	X	X	number	hourly	–
windSpeedPvHr	X	X	number	hourly	–
windSpeedHrAv	X	X	number	hourly	–
windSpeedSh	X	X	number	subhourly	–
windSpeedSquaredSh	X	X	number	subhourly	–
windSpeedSqrtSh	X	X	number	subhourly	–
windSpeedPt8Sh	X	X	number	subhourly	–
windDirDegHr	X	X	number	hourly	–
wOHr	X	X	number	hourly	–
wOPvHr	X	X	number	hourly	–
wOHrAv	X	X	number	hourly	–
wOSh	X	X	number	subhourly	–
hOSh	X	X	number	subhourly	–
airxOSh	X	X	number	subhourly	–
rhoMoistOSh	X	X	number	subhourly	–
rhoDryOSh	X	X	number	subhourly	–
iter	X	X	integer number	subhourly	–
qcPeak	X	X	number	hourly	–
qcPeakH	X	X	integer number	hourly	–
qcPeakD	X	X	integer number	hourly	–
qcPeakM	X	X	integer number	hourly	–
qhPeak	X	X	number	hourly	–
qhPeakH	X	X	integer number	hourly	–
qhPeakD	X	X	integer number	hourly	–
qhPeakM	X	X	integer number	hourly	–
ck5aa5	X	X	integer number	run start time (of each phase, autoSize or simulate)	–

6.46 @towerPlant[1..].

Name	Input?	Runtime?	Type	Variability	Description
name	X	X	string	constant	–

Name	Input?	Runtime?	Type	Variability	Description
ctN	X	X	integer number	autosize and simulate phase start time	Number of towers. Niles' ctNo. default 1.
tpStg	X	X	unrecognized	autosize and simulate phase start time	–
tpTsSp	X	X	number	hourly	Towers delivered water setpoint temperature (Niles' twoSp). degrees F, hourly, default 85F.
tpMtr	X	X	integer number	input time	–
ctTy	X	X	unrecognized	autosize and simulate phase start time	Cooling tower fan control type choice: ONESPEED (default), TWOSPEED, or VARIABLE.
ctLoSpd	X	X	number	autosize and simulate phase start time	Low speed for a TWOSPEED fan, as a fraction of full cfm. default 0.5.
ctShaftPwr	X	X	number	autosize and simulate phase start time	–
ctMotEff	X	X	number	autosize and simulate phase start time	Motor (and drive, if any) efficiency, default 0.88
ctFcOne.k[0]	X	X	number	run start time (of each phase, autoSize or simulate)	–
ctFcOne.k[1]	X	X	number	run start time (of each phase, autoSize or simulate)	–
ctFcOne.k[2]	X	X	number	run start time (of each phase, autoSize or simulate)	–
ctFcLo.k[0]	X	X	number	run start time (of each phase, autoSize or simulate)	–

Name	Input?	Runtime?	Type	Variability	Description
ctFcLo.k[1]	X	X	number	run start time (of each phase, autoSize or simulate)	–
ctFcLo.k[2]	X	X	number	run start time (of each phase, autoSize or simulate)	–
ctFcHi.k[0]	X	X	number	run start time (of each phase, autoSize or simulate)	–
ctFcHi.k[1]	X	X	number	run start time (of each phase, autoSize or simulate)	–
ctFcHi.k[2]	X	X	number	run start time (of each phase, autoSize or simulate)	–
ctFcVar.k[0]	X	X	number	run start time (of each phase, autoSize or simulate)	–
ctFcVar.k[1]	X	X	number	run start time (of each phase, autoSize or simulate)	–
ctFcVar.k[2]	X	X	number	run start time (of each phase, autoSize or simulate)	–
ctFcVar.k[3]	X	X	number	run start time (of each phase, autoSize or simulate)	–
ctFcVar.k[4]	X	X	number	run start time (of each phase, autoSize or simulate)	–

Name	Input?	Runtime?	Type	Variability	Description
ctCapDs	X	X	number	run start time (of each phase, autoSize or simulate)	–
ctVfDs	X	X	number	autosize and simulate phase start time	Design air flow volume rate through tower / full speed fan flow??, cfm, RQD.
ctGpmDs	X	X	number	run start time (of each phase, autoSize or simulate)	Design water flow rate, gpm. default: sum of connected heat rejection pump capacities / ctN.
ctTDbODs	X	X	number	autosize and simulate phase start time	Design outdoor drybulb temperature, F, RQD. (only needed to convert ctVfDs from cfm to lb/hr).
ctTWbODs	X	X	number	autosize and simulate phase start time	Design outdoor wetbulb temperature, F, RQD.
ctTwoDs	X	X	number	autosize and simulate phase start time	Design leaving water temperature, F, default 85.
ctCapOd	X	X	number	run start time (of each phase, autoSize or simulate)	–
ctVfOd	X	X	number	autosize and simulate phase start time	Off-design air flow volume rate through one tower, cfm, must != ctVfDs.
ctGpmOd	X	X	number	run start time (of each phase, autoSize or simulate)	–
ctTDbOOd	X	X	number	autosize and simulate phase start time	Off-design outdoor drybulb temperature, F. (only needed to convert ctVfOd from cfm to lb/hr).
ctTWbOOd	X	X	number	autosize and simulate phase start time	Off-design outdoor wetbulb temperature, F.

Name	Input?	Runtime?	Type	Variability	Description
ctTwoOd	X	X	number	autosize and simulate phase start time	Off-design leaving water temperature, F, default 85.
ctK	X	X	number	run start time (of each phase, autoSize or simulate)	exponent in formula ntuA = const * (mw/ma)^ctK, as alternative to "off design" inputs.
ctStkFlFr	X	X	number	autosize and simulate phase start time	–
ctBldn	X	X	number	autosize and simulate phase start time	Blowdown rate: frac inflowing water bled down drain, to reduce impurities buildup. default .01.
ctDrft	X	X	number	autosize and simulate phase start time	Drift rate: frac inflowing water blown out of tower as droplets, w/o evaporating. default 0.
ctTWm	X	X	number	autosize and simulate phase start time	Temperature of water in mains, for makeup water. default 60.
cp1	X	X	integer number	run start time (of each phase, autoSize or simulate)	subscript of 1st COOLPLANT served by this TOWERPLANT. Next is COOLPLANT.nxCp4tp.
hl1	X	X	integer number	run start time (of each phase, autoSize or simulate)	subscript of 1st HPLOOP with hx served by this TOWERPLANT. Next is HPLOOP.nxHl4tp.
oneFanP	X	X	number	run start time (of each phase, autoSize or simulate)	full-speed motor input power for one fan (Btuh): ctShaftPwr / ctMotEff.
maDs	X	X	number	run start time (of each phase, autoSize or simulate)	design air flow into 1 tower, ctVfDs converted from cfm to lb/hr
maOd	X	X	number	run start time (of each phase, autoSize or simulate)	off-design air flow into 1 tower, ctVfOd converted from cfm to lb/hr

Name	Input?	Runtime?	Type	Variability	Description
mwDs	X	X	number	run start time (of each phase, autoSize or simulate)	design water flow into 1 tower, ctGpmDs converted from gpm to lb/hr
mwOd	X	X	number	run start time (of each phase, autoSize or simulate)	off-design water flow into 1 tower, ctGpmOd converted from gpm to lb/hr
maOverMwDs	X	X	number	run start time (of each phase, autoSize or simulate)	maDs/mwDs, precomputed in setup.
ntuADs	X	X	number	run start time (of each phase, autoSize or simulate)	number of transfer units for air side at design conditions (Niles ntuAd)
ntuAOd	X	X	number	run start time (of each phase, autoSize or simulate)	.. at off-design conditions, if given. member only as debug aid.
tpTs	X	X	number	end of each subhour	–
tpClf	X	X	integer number	end of each subhour	call-flag: set nz if must call tpCompute so it can test tr, etc to see if computation needed.
tpPtf	X	X	integer number	end of each subhour	compute-flag: set if must call tpCompute and it should unconditionally recompute.
trNx	X	X	number	end of each subhour	return water temp, F
mwAllNx	X	X	number	end of each subhour	return water flow===total water flow entering towers, sum of loads, lb/hr.
qLoadNx	X	X	number	end of each subhour	heat added to water by loads. Positive. Believe need in record only for debugging/reporting.
tr	X	X	number	end of each subhour	return water temp
mwAll	X	X	number	end of each subhour	return water flow (sum of loads)===total water flow into all towers, lb/hr.

Name	Input?	Runtime?	Type	Variability	Description
qLoad	X	X	number	end of each subhour	heat added to water by loads. Positive. Believe need in record only for debugging/reporting.
mwi1	X	X	number	end of each subhour	flow into 1 tower (less flows out due to evaporation & drift): mwAll / ctN.
qNeed	X	X	number	end of each subhour	power needed from tower plant to deliver water at setpoint: (tpTsSp - tr) * mwAll. negative.
qMax1	X	X	number	end of each subhour	max power of 1 tower under current conditions
qMin1	X	X	number	end of each subhour	power of 1 tower, fan off (stack effect only) under current conditions
towldCase	X	X	unrecognized	end of each subhour	–
qMaxGuess	X	X	number	end of each subhour	for internal values for towModel initial guess at next call for various towModel calls.
qMinGuess	X	X	number	end of each subhour	..
qLoGuess	X	X	number	end of each subhour	..
qVarGuess	X	X	number	end of each subhour	.., used via varSpeedF
qVarTem	X	X	number	end of each subhour	varSpeedF temporary that should be saved between calls (last q, used re initial f)
puteTs	X	X	number	end of each subhour	tpTs from tpCompute, not set by tpEstimate. debug aid.
nCtOp	X	X	integer number	end of each subhour	number of tower fans operating
f	X	X	number	end of each subhour	fraction of full speed (fraction on for one speed fan), for lead tower only if LEAD.
fanP	X	X	number	end of each subhour	plant's fan input pwr this subhour (Btuh!)
q	X	X	number	end of each subhour	power imparted to water, for change detection/probes/reports 10-19-92
tpTsSpPr	X	X	number	end of each subhour	for tpEstimate
tpTsEstPr	X	X	number	end of each subhour	for tpEstimate

Name	Input?	Runtime?	Type	Variability	Description
tpTsPr	X	X	number	end of each subhour	for tpCompute
tDbOShPr	X	X	number	end of each subhour	for tpCompute
wOShPr	X	X	number	end of each subhour	for tpCompute

6.47 @weather.

Name	Input?	Runtime?	Type	Variability	Description
name	–	X	string	constant	–
bmrad	–	X	number	hourly	–
dfrad	–	X	number	hourly	–
db	–	X	number	hourly	–
wb	–	X	number	hourly	–
wnddir	–	X	number	hourly	–
wndspd	–	X	number	hourly	–
glrad	–	X	number	hourly	–
cldCvr	–	X	number	hourly	–
tSky	–	X	number	hourly	–
tGrnd	–	X	number	hourly	–
taDp	–	X	number	hourly	–
tMains	–	X	number	hourly	–
taDbPvPk	–	X	number	hourly	–
taDbAvg01	–	X	number	hourly	–
taDbAvg07	–	X	number	hourly	–
taDbAvg14	–	X	number	hourly	–
taDbAvg31	–	X	number	hourly	–
taDbAvg	–	X	number	hourly	–
tdvElec	–	X	number	hourly	–
tdvNatGas	–	X	number	hourly	–
tdvPropane	–	X	number	hourly	–

6.48 @weatherFile.

Name	Input?	Runtime?	Type	Variability	Description
name	–	X	string	constant	–
wFileFormat	–	X	integer number	run start time (of each phase, autoSize or simulate)	–
loc	–	X	string	run start time (of each phase, autoSize or simulate)	–
lid	–	X	string	run start time (of each phase, autoSize or simulate)	–
yr	–	X	integer number	run start time (of each phase, autoSize or simulate)	–

Name	Input?	Runtime?	Type	Variability	Description
jd1	–	X	integer number	run start time (of each phase, autoSize or simulate)	–
jdl	–	X	integer number	run start time (of each phase, autoSize or simulate)	–
lat	–	X	number	run start time (of each phase, autoSize or simulate)	–
lon	–	X	number	run start time (of each phase, autoSize or simulate)	–
tz	–	X	number	run start time (of each phase, autoSize or simulate)	–
elev	–	X	number	run start time (of each phase, autoSize or simulate)	–
taDbAvgYr	–	X	number	run start time (of each phase, autoSize or simulate)	–
tMainsAvgYr	–	X	number	autosize and simulate phase start time	–
solartime	–	X	integer number	run start time (of each phase, autoSize or simulate)	–
loc2	–	X	string	run start time (of each phase, autoSize or simulate)	–
isLeap	–	X	integer number	run start time (of each phase, autoSize or simulate)	–
firstDdm	–	X	integer number	run start time (of each phase, autoSize or simulate)	–
lastDdm	–	X	integer number	run start time (of each phase, autoSize or simulate)	–
winMOE	–	X	integer number	run start time (of each phase, autoSize or simulate)	–
win99TDb	–	X	integer number	run start time (of each phase, autoSize or simulate)	–
win97TDb	–	X	integer number	run start time (of each phase, autoSize or simulate)	–
sum1TDb	–	X	integer number	run start time (of each phase, autoSize or simulate)	–

Name	Input?	Runtime?	Type	Variability	Description
sum1TWb	–	X	integer number	run start time (of each phase, autoSize or simulate)	–
sum2TDb	–	X	integer number	run start time (of each phase, autoSize or simulate)	–
sum2TWb	–	X	integer number	run start time (of each phase, autoSize or simulate)	–
sum5TDb	–	X	integer number	run start time (of each phase, autoSize or simulate)	–
sum5TWb	–	X	integer number	run start time (of each phase, autoSize or simulate)	–
range	–	X	integer number	run start time (of each phase, autoSize or simulate)	–
sumMonHi	–	X	integer number	run start time (of each phase, autoSize or simulate)	–

6.49 @weatherNextHour.

Name	Input?	Runtime?	Type	Variability	Description
name	–	X	string	constant	–
bmrad	–	X	number	hourly	–
dfrad	–	X	number	hourly	–
db	–	X	number	hourly	–
wb	–	X	number	hourly	–
wnddir	–	X	number	hourly	–
wndspd	–	X	number	hourly	–
glrad	–	X	number	hourly	–
cldCvr	–	X	number	hourly	–
tSky	–	X	number	hourly	–
tGrnd	–	X	number	hourly	–
taDp	–	X	number	hourly	–
tMains	–	X	number	hourly	–
taDbPvPk	–	X	number	hourly	–
taDbAvg01	–	X	number	hourly	–
taDbAvg07	–	X	number	hourly	–
taDbAvg14	–	X	number	hourly	–
taDbAvg31	–	X	number	hourly	–
taDbAvg	–	X	number	hourly	–
tdvElec	–	X	number	hourly	–
tdvNatGas	–	X	number	hourly	–
tdvPropane	–	X	number	hourly	–

6.50 @window[1..]. (owner: surface)

Name	Input?	Runtime?	Type	Variability	Description
name	X	–	string	constant	–
ty	X	–	integer number	input time	–
area	X	–	number	run start time (of each phase, autoSize or simulate)	–
azm	X	–	number	run start time (of each phase, autoSize or simulate)	–
tilt	X	–	number	run start time (of each phase, autoSize or simulate)	–
dircos[0]	X	–	number	run start time (of each phase, autoSize or simulate)	–
dircos[1]	X	–	number	run start time (of each phase, autoSize or simulate)	–
dircos[2]	X	–	number	run start time (of each phase, autoSize or simulate)	–
depthBG	X	–	number	run start time (of each phase, autoSize or simulate)	–
model	X	–	integer number	input time	–
modelr	X	–	integer number	run start time (of each phase, autoSize or simulate)	–
lThkF	X	–	number	run start time (of each phase, autoSize or simulate)	–
gti	X	–	integer number	run start time (of each phase, autoSize or simulate)	–
sco	X	–	number	monthly-hourly	–
scc	X	–	number	monthly-hourly	–
sbcI.absSlr	X	–	number	monthly-hourly	–
sbcI.awAbsSlr	X	–	number	monthly-hourly	–
sbcI.epsLW	X	–	number	run start time (of each phase, autoSize or simulate)	–

Name	Input?	Runtime?	Type	Variability	Description
sbcI.zi	X	–	integer number	run start time (of each phase, autoSize or simulate)	–
sbcI.F	X	–	number	run start time (of each phase, autoSize or simulate)	–
sbcI.Fp	X	–	number	run start time (of each phase, autoSize or simulate)	–
sbcI.frRad	X	–	number	run start time (of each phase, autoSize or simulate)	–
sbcI.fSky	X	–	number	run start time (of each phase, autoSize or simulate)	–
sbcI.fAir	X	–	number	run start time (of each phase, autoSize or simulate)	–
sbcI.hcNat	X	–	number	end of each subhour	–
sbcI.hcFrc	X	–	number	end of each subhour	–
sbcI.hcMult	X	–	number	end of each subhour	–
sbcI.hxa	X	–	number	end of each subhour	–
sbcI.hxr	X	–	number	end of each subhour	–
sbcI.hxtot	X	–	number	end of each subhour	–
sbcI.uRat	X	–	number	end of each subhour	–
sbcI.fRat	X	–	number	end of each subhour	–
sbcI.cx	X	–	number	end of each subhour	–
sbcI.sgTarg.bm	X	–	number	end of each subhour	–
sbcI.sgTarg.df	X	–	number	end of each subhour	–
sbcI.sgTarg.tot	X	–	number	end of each subhour	–
sbcI.sg	X	–	number	end of each subhour	–
sbcI.tSrf	X	–	number	end of each subhour	–

Name	Input?	Runtime?	Type	Variability	Description
sbcI.tSrfls	X	–	number	subhourly	–
sbcI.qrAbs	X	–	number	end of each subhour	–
sbcI.txa	X	–	number	end of each subhour	–
sbcI.txr	X	–	number	end of each subhour	–
sbcI.txe	X	–	number	end of each subhour	–
sbcI.w	X	–	number	end of each subhour	–
sbcI.qSrf	X	–	number	end of each subhour	–
sbcI.pXS	X	–	unrecognized	run start time (of each phase, autoSize or simulate)	–
sbcI.si	X	–	unrecognized	run start time (of each phase, autoSize or simulate)	–
sbcI.fcWind	X	–	number	run start time (of each phase, autoSize or simulate)	–
sbcI.fcWind2	X	–	number	run start time (of each phase, autoSize or simulate)	–
sbcI.eta	X	–	number	end of each subhour	–
sbcI.widNom	X	–	number	run start time (of each phase, autoSize or simulate)	–
sbcI.lenNom	X	–	number	run start time (of each phase, autoSize or simulate)	–
sbcI.lenCharNat	X	–	number	run start time (of each phase, autoSize or simulate)	–
sbcI.lenEffWink	X	–	number	run start time (of each phase, autoSize or simulate)	–
sbcI.atvDeg	X	–	number	run start time (of each phase, autoSize or simulate)	–

Name	Input?	Runtime?	Type	Variability	Description
sbcI.cosAtv	X	–	number	run start time (of each phase, autoSize or simulate)	–
sbcI.hcModel	X	–	unrecognized	run start time (of each phase, autoSize or simulate)	–
sbcI.hcLChar	X	–	number	run start time (of each phase, autoSize or simulate)	–
sbcI.hcConst[0]	X	–	number	run start time (of each phase, autoSize or simulate)	–
sbcI.hcConst[1]	X	–	number	run start time (of each phase, autoSize or simulate)	–
sbcI.hcConst[2]	X	–	number	run start time (of each phase, autoSize or simulate)	–
sbcI.groundModel	X	–	unrecognized	run start time (of each phase, autoSize or simulate)	–
sbcI.cTaDbAvgYr	X	–	number	run start time (of each phase, autoSize or simulate)	–
sbcI.cTaDbAvg31	X	–	number	run start time (of each phase, autoSize or simulate)	–
sbcI.cTaDbAvg14	X	–	number	run start time (of each phase, autoSize or simulate)	–
sbcI.cTaDbAvg07	X	–	number	run start time (of each phase, autoSize or simulate)	–
sbcI.cTGrnd	X	–	number	run start time (of each phase, autoSize or simulate)	–
sbcI.rGrnd	X	–	number	run start time (of each phase, autoSize or simulate)	–

Name	Input?	Runtime?	Type	Variability	Description
sbcI.rConGrnd	X	–	number	run start time (of each phase, autoSize or simulate)	–
sbcO.absSlr	X	–	number	monthly-hourly	–
sbcO.awAbsSlr	X	–	number	monthly-hourly	–
sbcO.epsLW	X	–	number	run start time (of each phase, autoSize or simulate)	–
sbcO.zi	X	–	integer number	run start time (of each phase, autoSize or simulate)	–
sbcO.F	X	–	number	run start time (of each phase, autoSize or simulate)	–
sbcO.Fp	X	–	number	run start time (of each phase, autoSize or simulate)	–
sbcO.frRad	X	–	number	run start time (of each phase, autoSize or simulate)	–
sbcO.fSky	X	–	number	run start time (of each phase, autoSize or simulate)	–
sbcO.fAir	X	–	number	run start time (of each phase, autoSize or simulate)	–
sbcO.hcNat	X	–	number	end of each subhour	–
sbcO.hcFrc	X	–	number	end of each subhour	–
sbcO.hcMult	X	–	number	end of each subhour	–
sbcO.hxa	X	–	number	end of each subhour	–
sbcO.hxr	X	–	number	end of each subhour	–
sbcO.hxtot	X	–	number	end of each subhour	–
sbcO.uRat	X	–	number	end of each subhour	–
sbcO.fRat	X	–	number	end of each subhour	–
sbcO.cx	X	–	number	end of each subhour	–

Name	Input?	Runtime?	Type	Variability	Description
sbcO.sgTarg.bm	X	–	number	end of each subhour	–
sbcO.sgTarg.df	X	–	number	end of each subhour	–
sbcO.sgTarg.tot	X	–	number	end of each subhour	–
sbcO.sg	X	–	number	end of each subhour	–
sbcO.tSrf	X	–	number	end of each subhour	–
sbcO.tSrfls	X	–	number	subhourly	–
sbcO.qrAbs	X	–	number	end of each subhour	–
sbcO.txa	X	–	number	end of each subhour	–
sbcO.txr	X	–	number	end of each subhour	–
sbcO.txe	X	–	number	end of each subhour	–
sbcO.w	X	–	number	end of each subhour	–
sbcO.qSrf	X	–	number	end of each subhour	–
sbcO.pXS	X	–	unrecognized	run start time (of each phase, autoSize or simulate)	–
sbcO.si	X	–	unrecognized	run start time (of each phase, autoSize or simulate)	–
sbcO.fcWind	X	–	number	run start time (of each phase, autoSize or simulate)	–
sbcO.fcWind2	X	–	number	run start time (of each phase, autoSize or simulate)	–
sbcO.eta	X	–	number	end of each subhour	–
sbcO.widNom	X	–	number	run start time (of each phase, autoSize or simulate)	–
sbcO.lenNom	X	–	number	run start time (of each phase, autoSize or simulate)	–

Name	Input?	Runtime?	Type	Variability	Description
sbcO.lenCharNat	X	–	number	run start time (of each phase, autoSize or simulate)	–
sbcO.lenEffWink	X	–	number	run start time (of each phase, autoSize or simulate)	–
sbcO.atvDeg	X	–	number	run start time (of each phase, autoSize or simulate)	–
sbcO.cosAtv	X	–	number	run start time (of each phase, autoSize or simulate)	–
sbcO.hcModel	X	–	unrecognized	run start time (of each phase, autoSize or simulate)	–
sbcO.hcLChar	X	–	number	run start time (of each phase, autoSize or simulate)	–
sbcO.hcConst[0]	X	–	number	run start time (of each phase, autoSize or simulate)	–
sbcO.hcConst[1]	X	–	number	run start time (of each phase, autoSize or simulate)	–
sbcO.hcConst[2]	X	–	number	run start time (of each phase, autoSize or simulate)	–
sbcO.groundModel	X	–	unrecognized	run start time (of each phase, autoSize or simulate)	–
sbcO.cTaDbAvgYr	X	–	number	run start time (of each phase, autoSize or simulate)	–
sbcO.cTaDbAvg31	X	–	number	run start time (of each phase, autoSize or simulate)	–
sbcO.cTaDbAvg14	X	–	number	run start time (of each phase, autoSize or simulate)	–

Name	Input?	Runtime?	Type	Variability	Description
sbcO.cTaDbAvg07	X	–	number	run start time (of each phase, autoSize or simulate)	–
sbcO.cTGrnd	X	–	number	run start time (of each phase, autoSize or simulate)	–
sbcO.rGrnd	X	–	number	run start time (of each phase, autoSize or simulate)	–
sbcO.rConGrnd	X	–	number	run start time (of each phase, autoSize or simulate)	–
fenModel	X	–	unrecognized	input time	–
SHGC	X	–	number	input time	–
fMult	X	–	number	run start time (of each phase, autoSize or simulate)	–
UNFRC	X	–	number	input time	–
NGlz	X	–	integer number	input time	–
exShd	X	–	unrecognized	input time	–
inShd	X	–	unrecognized	input time	–
dirtLoss	X	–	number	run start time (of each phase, autoSize or simulate)	–
sfExCnd	X	–	integer number	run start time (of each phase, autoSize or simulate)	–
sfExT	X	–	number	subhourly	–
sfAdjZi	X	–	integer number	input time	–
uI	X	–	number	run start time (of each phase, autoSize or simulate)	–
uC	X	–	number	run start time (of each phase, autoSize or simulate)	–
uX	X	–	number	run start time (of each phase, autoSize or simulate)	–
Rf	X	–	number	run start time (of each phase, autoSize or simulate)	–

Name	Input?	Runtime?	Type	Variability	Description
grndRefl	X	–	number	monthly-hourly	–
vfSkyDf	X	–	number	monthly-hourly	–
vfGrndDf	X	–	number	monthly-hourly	–
vfSkyLW	X	–	number	run start time (of each phase, autoSize or simulate)	–
vfGrndLW	X	–	number	run start time (of each phase, autoSize or simulate)	–
uval	X	–	number	run start time (of each phase, autoSize or simulate)	–
UNom	X	–	number	run start time (of each phase, autoSize or simulate)	–
UANom	X	–	number	run start time (of each phase, autoSize or simulate)	–
rSrfNom[0]	X	–	number	run start time (of each phase, autoSize or simulate)	–
rSrfNom[1]	X	–	number	run start time (of each phase, autoSize or simulate)	–
hSrfNom[0]	X	–	number	run start time (of each phase, autoSize or simulate)	–
hSrfNom[1]	X	–	number	run start time (of each phase, autoSize or simulate)	–
cFctr	X	–	number	run start time (of each phase, autoSize or simulate)	–
iwshad	X	–	integer number	run start time (of each phase, autoSize or simulate)	–
msi	X	–	integer number	run start time (of each phase, autoSize or simulate)	–
tLrB[0]	X	–	number	end of each hour	–

Name	Input?	Runtime?	Type	Variability	Description
tLrB[1]	X	–	number	end of each hour	–
tLrB[2]	X	–	number	end of each hour	–
tLrB[3]	X	–	number	end of each hour	–
tLrB[4]	X	–	number	end of each hour	–
tLrB[5]	X	–	number	end of each hour	–
tLrB[6]	X	–	number	end of each hour	–
tLrB[7]	X	–	number	end of each hour	–
tLrB[8]	X	–	number	end of each hour	–
tLrB[9]	X	–	number	end of each hour	–
nsgdist	X	–	integer number	run start time (of each phase, autoSize or simulate)	–
sgdist[0].targTy	X	–	integer number	run start time (of each phase, autoSize or simulate)	–
sgdist[0].targTi	X	–	integer number	run start time (of each phase, autoSize or simulate)	–
sgdist[0].FSO	X	–	number	monthly-hourly	–
sgdist[0].FSC	X	–	number	monthly-hourly	–
sgdist[1].targTy	X	–	integer number	run start time (of each phase, autoSize or simulate)	–
sgdist[1].targTi	X	–	integer number	run start time (of each phase, autoSize or simulate)	–
sgdist[1].FSO	X	–	number	monthly-hourly	–
sgdist[1].FSC	X	–	number	monthly-hourly	–
sgdist[2].targTy	X	–	integer number	run start time (of each phase, autoSize or simulate)	–
sgdist[2].targTi	X	–	integer number	run start time (of each phase, autoSize or simulate)	–
sgdist[2].FSO	X	–	number	monthly-hourly	–
sgdist[2].FSC	X	–	number	monthly-hourly	–
sgdist[3].targTy	X	–	integer number	run start time (of each phase, autoSize or simulate)	–
sgdist[3].targTi	X	–	integer number	run start time (of each phase, autoSize or simulate)	–
sgdist[3].FSO	X	–	number	monthly-hourly	–

Name	Input?	Runtime?	Type	Variability	Description
sgdist[3].FSC	X	–	number	monthly-hourly	–
sgdist[4].targTy	X	–	integer number	run start time (of each phase, autoSize or simulate)	–
sgdist[4].targTi	X	–	integer number	run start time (of each phase, autoSize or simulate)	–
sgdist[4].FSO	X	–	number	monthly-hourly	–
sgdist[4].FSC	X	–	number	monthly-hourly	–
sgdist[5].targTy	X	–	integer number	run start time (of each phase, autoSize or simulate)	–
sgdist[5].targTi	X	–	integer number	run start time (of each phase, autoSize or simulate)	–
sgdist[5].FSO	X	–	number	monthly-hourly	–
sgdist[5].FSC	X	–	number	monthly-hourly	–
sgdist[6].targTy	X	–	integer number	run start time (of each phase, autoSize or simulate)	–
sgdist[6].targTi	X	–	integer number	run start time (of each phase, autoSize or simulate)	–
sgdist[6].FSO	X	–	number	monthly-hourly	–
sgdist[6].FSC	X	–	number	monthly-hourly	–
sgdist[7].targTy	X	–	integer number	run start time (of each phase, autoSize or simulate)	–
sgdist[7].targTi	X	–	integer number	run start time (of each phase, autoSize or simulate)	–
sgdist[7].FSO	X	–	number	monthly-hourly	–
sgdist[7].FSC	X	–	number	monthly-hourly	–
sfClass	X	–	unrecognized	input time	–
sfArea	X	–	number	input time	–
sfU	X	–	number	input time	–
sfCon	X	–	integer number	input time	–
sfTy	X	–	integer number	constant	–
width	X	–	number	input time	–
height	X	–	number	input time	–
mult	X	–	number	input time	–

Name	Input?	Runtime?	Type	Variability	Description
xi	X	–	integer number	run start time (of each phase, autoSize or simulate)	–
msi	X	–	integer number	run start time (of each phase, autoSize or simulate)	–

6.51 @xsurf[1..].

Name	Input?	Runtime?	Type	Variability	Description
name	–	X	string	constant	–
nxXsurf	–	X	integer number	run start time (of each phase, autoSize or simulate)	–
nxXsSpecT	–	X	integer number	run start time (of each phase, autoSize or simulate)	–
ty	–	X	integer number	run start time (of each phase, autoSize or simulate)	–
area	–	X	number	run start time (of each phase, autoSize or simulate)	–
azm	–	X	number	run start time (of each phase, autoSize or simulate)	–
tilt	–	X	number	run start time (of each phase, autoSize or simulate)	–
dircos[0]	–	X	number	run start time (of each phase, autoSize or simulate)	–
dircos[1]	–	X	number	run start time (of each phase, autoSize or simulate)	–
dircos[2]	–	X	number	run start time (of each phase, autoSize or simulate)	–

Name	Input?	Runtime?	Type	Variability	Description
depthBG	–	X	number	run start time (of each phase, autoSize or simulate)	–
model	–	X	integer number	run start time (of each phase, autoSize or simulate)	–
modelr	–	X	integer number	run start time (of each phase, autoSize or simulate)	–
lThkF	–	X	number	run start time (of each phase, autoSize or simulate)	–
gti	–	X	integer number	run start time (of each phase, autoSize or simulate)	–
sco	–	X	number	monthly-hourly	–
scc	–	X	number	monthly-hourly	–
sbcI.absSlr	–	X	number	monthly-hourly	–
sbcI.awAbsSlr	–	X	number	monthly-hourly	–
sbcI.epsLW	–	X	number	run start time (of each phase, autoSize or simulate)	–
sbcI.zi	–	X	integer number	run start time (of each phase, autoSize or simulate)	–
sbcI.F	–	X	number	run start time (of each phase, autoSize or simulate)	–
sbcI.Fp	–	X	number	run start time (of each phase, autoSize or simulate)	–
sbcI.frRad	–	X	number	run start time (of each phase, autoSize or simulate)	–
sbcI.fSky	–	X	number	run start time (of each phase, autoSize or simulate)	–
sbcI.fAir	–	X	number	run start time (of each phase, autoSize or simulate)	–

Name	Input?	Runtime?	Type	Variability	Description
sbcI.hcNat	–	X	number	end of each subhour	–
sbcI.hcFrc	–	X	number	end of each subhour	–
sbcI.hcMult	–	X	number	end of each subhour	–
sbcI.hxa	–	X	number	end of each subhour	–
sbcI.hxr	–	X	number	end of each subhour	–
sbcI.hxtot	–	X	number	end of each subhour	–
sbcI.uRat	–	X	number	end of each subhour	–
sbcI.fRat	–	X	number	end of each subhour	–
sbcI.cx	–	X	number	end of each subhour	–
sbcI.sgTarg.bm	–	X	number	end of each subhour	–
sbcI.sgTarg.df	–	X	number	end of each subhour	–
sbcI.sgTarg.tot	–	X	number	end of each subhour	–
sbcI.sg	–	X	number	end of each subhour	–
sbcI.tSrf	–	X	number	end of each subhour	–
sbcI.tSrfls	–	X	number	subhourly	–
sbcI.qrAbs	–	X	number	end of each subhour	–
sbcI.txa	–	X	number	end of each subhour	–
sbcI.txr	–	X	number	end of each subhour	–
sbcI.txe	–	X	number	end of each subhour	–
sbcI.w	–	X	number	end of each subhour	–
sbcI.qSrf	–	X	number	end of each subhour	–
sbcI.pXS	–	X	unrecognized	run start time (of each phase, autoSize or simulate)	–
sbcI.si	–	X	unrecognized	run start time (of each phase, autoSize or simulate)	–

Name	Input?	Runtime?	Type	Variability	Description
sbcI.fcWind	–	X	number	run start time (of each phase, autoSize or simulate)	–
sbcI.fcWind2	–	X	number	run start time (of each phase, autoSize or simulate)	–
sbcI.eta	–	X	number	end of each subhour	–
sbcI.widNom	–	X	number	run start time (of each phase, autoSize or simulate)	–
sbcI.lenNom	–	X	number	run start time (of each phase, autoSize or simulate)	–
sbcI.lenCharNat	–	X	number	run start time (of each phase, autoSize or simulate)	–
sbcI.lenEffWink	–	X	number	run start time (of each phase, autoSize or simulate)	–
sbcI.atvDeg	–	X	number	run start time (of each phase, autoSize or simulate)	–
sbcI.cosAtv	–	X	number	run start time (of each phase, autoSize or simulate)	–
sbcI.hcModel	–	X	unrecognized	run start time (of each phase, autoSize or simulate)	–
sbcI.hcLChar	–	X	number	run start time (of each phase, autoSize or simulate)	–
sbcI.hcConst[0]	–	X	number	run start time (of each phase, autoSize or simulate)	–
sbcI.hcConst[1]	–	X	number	run start time (of each phase, autoSize or simulate)	–

Name	Input?	Runtime?	Type	Variability	Description
sbcI.hcConst[2]	–	X	number	run start time (of each phase, autoSize or simulate)	–
sbcI.groundModel	–	X	unrecognized	run start time (of each phase, autoSize or simulate)	–
sbcI.cTaDbAvgYr	–	X	number	run start time (of each phase, autoSize or simulate)	–
sbcI.cTaDbAvg31	–	X	number	run start time (of each phase, autoSize or simulate)	–
sbcI.cTaDbAvg14	–	X	number	run start time (of each phase, autoSize or simulate)	–
sbcI.cTaDbAvg07	–	X	number	run start time (of each phase, autoSize or simulate)	–
sbcI.cTGrnd	–	X	number	run start time (of each phase, autoSize or simulate)	–
sbcI.rGrnd	–	X	number	run start time (of each phase, autoSize or simulate)	–
sbcI.rConGrnd	–	X	number	run start time (of each phase, autoSize or simulate)	–
sbcO.absSlr	–	X	number	monthly-hourly	–
sbcO.awAbsSlr	–	X	number	monthly-hourly	–
sbcO.epsLW	–	X	number	run start time (of each phase, autoSize or simulate)	–
sbcO.zi	–	X	integer number	run start time (of each phase, autoSize or simulate)	–
sbcO.F	–	X	number	run start time (of each phase, autoSize or simulate)	–

Name	Input?	Runtime?	Type	Variability	Description
sbcO.Fp	–	X	number	run start time (of each phase, autoSize or simulate)	–
sbcO.frRad	–	X	number	run start time (of each phase, autoSize or simulate)	–
sbcO.fSky	–	X	number	run start time (of each phase, autoSize or simulate)	–
sbcO.fAir	–	X	number	run start time (of each phase, autoSize or simulate)	–
sbcO.hcNat	–	X	number	end of each subhour	–
sbcO.hcFrc	–	X	number	end of each subhour	–
sbcO.hcMult	–	X	number	end of each subhour	–
sbcO.hxa	–	X	number	end of each subhour	–
sbcO.hxr	–	X	number	end of each subhour	–
sbcO.hxtot	–	X	number	end of each subhour	–
sbcO.uRat	–	X	number	end of each subhour	–
sbcO.fRat	–	X	number	end of each subhour	–
sbcO.cx	–	X	number	end of each subhour	–
sbcO.sgTarg.bm	–	X	number	end of each subhour	–
sbcO.sgTarg.df	–	X	number	end of each subhour	–
sbcO.sgTarg.tot	–	X	number	end of each subhour	–
sbcO.sg	–	X	number	end of each subhour	–
sbcO.tSrf	–	X	number	end of each subhour	–
sbcO.tSrfls	–	X	number	subhourly	–
sbcO.qrAbs	–	X	number	end of each subhour	–
sbcO.txa	–	X	number	end of each subhour	–
sbcO.txr	–	X	number	end of each subhour	–

Name	Input?	Runtime?	Type	Variability	Description
sbcO.txe	–	X	number	end of each subhour	–
sbcO.w	–	X	number	end of each subhour	–
sbcO.qSrf	–	X	number	end of each subhour	–
sbcO.pXS	–	X	unrecognized	run start time (of each phase, autoSize or simulate)	–
sbcO.si	–	X	unrecognized	run start time (of each phase, autoSize or simulate)	–
sbcO.fcWind	–	X	number	run start time (of each phase, autoSize or simulate)	–
sbcO.fcWind2	–	X	number	run start time (of each phase, autoSize or simulate)	–
sbcO.eta	–	X	number	end of each subhour	–
sbcO.widNom	–	X	number	run start time (of each phase, autoSize or simulate)	–
sbcO.lenNom	–	X	number	run start time (of each phase, autoSize or simulate)	–
sbcO.lenCharNat	–	X	number	run start time (of each phase, autoSize or simulate)	–
sbcO.lenEffWink	–	X	number	run start time (of each phase, autoSize or simulate)	–
sbcO.atvDeg	–	X	number	run start time (of each phase, autoSize or simulate)	–
sbcO.cosAtv	–	X	number	run start time (of each phase, autoSize or simulate)	–
sbcO.hcModel	–	X	unrecognized	run start time (of each phase, autoSize or simulate)	–

Name	Input?	Runtime?	Type	Variability	Description
sbcO.hcLChar	–	X	number	run start time (of each phase, autoSize or simulate)	–
sbcO.hcConst[0]	–	X	number	run start time (of each phase, autoSize or simulate)	–
sbcO.hcConst[1]	–	X	number	run start time (of each phase, autoSize or simulate)	–
sbcO.hcConst[2]	–	X	number	run start time (of each phase, autoSize or simulate)	–
sbcO.groundModel	–	X	unrecognized	run start time (of each phase, autoSize or simulate)	–
sbcO.cTaDbAvgYr	–	X	number	run start time (of each phase, autoSize or simulate)	–
sbcO.cTaDbAvg31	–	X	number	run start time (of each phase, autoSize or simulate)	–
sbcO.cTaDbAvg14	–	X	number	run start time (of each phase, autoSize or simulate)	–
sbcO.cTaDbAvg07	–	X	number	run start time (of each phase, autoSize or simulate)	–
sbcO.cTGrnd	–	X	number	run start time (of each phase, autoSize or simulate)	–
sbcO.rGrnd	–	X	number	run start time (of each phase, autoSize or simulate)	–
sbcO.rConGrnd	–	X	number	run start time (of each phase, autoSize or simulate)	–
fenModel	–	X	unrecognized	run start time (of each phase, autoSize or simulate)	–

Name	Input?	Runtime?	Type	Variability	Description
SHGC	–	X	number	run start time (of each phase, autoSize or simulate)	–
fMult	–	X	number	run start time (of each phase, autoSize or simulate)	–
UNFRC	–	X	number	run start time (of each phase, autoSize or simulate)	–
NGlz	–	X	integer number	run start time (of each phase, autoSize or simulate)	–
exShd	–	X	unrecognized	run start time (of each phase, autoSize or simulate)	–
inShd	–	X	unrecognized	run start time (of each phase, autoSize or simulate)	–
dirtLoss	–	X	number	run start time (of each phase, autoSize or simulate)	–
sfExCnd	–	X	integer number	run start time (of each phase, autoSize or simulate)	–
sfExT	–	X	number	subhourly	–
sfAdjZi	–	X	integer number	run start time (of each phase, autoSize or simulate)	–
uI	–	X	number	run start time (of each phase, autoSize or simulate)	–
uC	–	X	number	run start time (of each phase, autoSize or simulate)	–
uX	–	X	number	run start time (of each phase, autoSize or simulate)	–

Name	Input?	Runtime?	Type	Variability	Description
Rf	–	X	number	run start time (of each phase, autoSize or simulate)	–
grndRefl	–	X	number	monthly-hourly	–
vfSkyDf	–	X	number	monthly-hourly	–
vfGrndDf	–	X	number	monthly-hourly	–
vfSkyLW	–	X	number	run start time (of each phase, autoSize or simulate)	–
vfGrndLW	–	X	number	run start time (of each phase, autoSize or simulate)	–
uval	–	X	number	run start time (of each phase, autoSize or simulate)	–
UNom	–	X	number	run start time (of each phase, autoSize or simulate)	–
UANom	–	X	number	run start time (of each phase, autoSize or simulate)	–
rSrfNom[0]	–	X	number	run start time (of each phase, autoSize or simulate)	–
rSrfNom[1]	–	X	number	run start time (of each phase, autoSize or simulate)	–
hSrfNom[0]	–	X	number	run start time (of each phase, autoSize or simulate)	–
hSrfNom[1]	–	X	number	run start time (of each phase, autoSize or simulate)	–
cFctr	–	X	number	run start time (of each phase, autoSize or simulate)	–
iwshad	–	X	integer number	run start time (of each phase, autoSize or simulate)	–

Name	Input?	Runtime?	Type	Variability	Description
msi	–	X	integer number	run start time (of each phase, autoSize or simulate)	–
tLrB[0]	–	X	number	end of each hour	–
tLrB[1]	–	X	number	end of each hour	–
tLrB[2]	–	X	number	end of each hour	–
tLrB[3]	–	X	number	end of each hour	–
tLrB[4]	–	X	number	end of each hour	–
tLrB[5]	–	X	number	end of each hour	–
tLrB[6]	–	X	number	end of each hour	–
tLrB[7]	–	X	number	end of each hour	–
tLrB[8]	–	X	number	end of each hour	–
tLrB[9]	–	X	number	end of each hour	–
nsgdist	–	X	integer number	run start time (of each phase, autoSize or simulate)	–
sgdist[0].targTy	–	X	integer number	run start time (of each phase, autoSize or simulate)	–
sgdist[0].targTi	–	X	integer number	run start time (of each phase, autoSize or simulate)	–
sgdist[0].FSO	–	X	number	monthly-hourly	–
sgdist[0].FSC	–	X	number	monthly-hourly	–
sgdist[1].targTy	–	X	integer number	run start time (of each phase, autoSize or simulate)	–
sgdist[1].targTi	–	X	integer number	run start time (of each phase, autoSize or simulate)	–
sgdist[1].FSO	–	X	number	monthly-hourly	–
sgdist[1].FSC	–	X	number	monthly-hourly	–
sgdist[2].targTy	–	X	integer number	run start time (of each phase, autoSize or simulate)	–
sgdist[2].targTi	–	X	integer number	run start time (of each phase, autoSize or simulate)	–
sgdist[2].FSO	–	X	number	monthly-hourly	–
sgdist[2].FSC	–	X	number	monthly-hourly	–
sgdist[3].targTy	–	X	integer number	run start time (of each phase, autoSize or simulate)	–

Name	Input?	Runtime?	Type	Variability	Description
sgdist[3].targTi	–	X	integer number	run start time (of each phase, autoSize or simulate)	–
sgdist[3].FSO	–	X	number	monthly-hourly	–
sgdist[3].FSC	–	X	number	monthly-hourly	–
sgdist[4].targTy	–	X	integer number	run start time (of each phase, autoSize or simulate)	–
sgdist[4].targTi	–	X	integer number	run start time (of each phase, autoSize or simulate)	–
sgdist[4].FSO	–	X	number	monthly-hourly	–
sgdist[4].FSC	–	X	number	monthly-hourly	–
sgdist[5].targTy	–	X	integer number	run start time (of each phase, autoSize or simulate)	–
sgdist[5].targTi	–	X	integer number	run start time (of each phase, autoSize or simulate)	–
sgdist[5].FSO	–	X	number	monthly-hourly	–
sgdist[5].FSC	–	X	number	monthly-hourly	–
sgdist[6].targTy	–	X	integer number	run start time (of each phase, autoSize or simulate)	–
sgdist[6].targTi	–	X	integer number	run start time (of each phase, autoSize or simulate)	–
sgdist[6].FSO	–	X	number	monthly-hourly	–
sgdist[6].FSC	–	X	number	monthly-hourly	–
sgdist[7].targTy	–	X	integer number	run start time (of each phase, autoSize or simulate)	–
sgdist[7].targTi	–	X	integer number	run start time (of each phase, autoSize or simulate)	–
sgdist[7].FSO	–	X	number	monthly-hourly	–
sgdist[7].FSC	–	X	number	monthly-hourly	–

6.52 @zhx[1..]. (owner: zone)

Name	Input?	Runtime?	Type	Variability	Description
name	–	X	string	constant	–

Name	Input?	Runtime?	Type	Variability	Description
zhxTy	–	X	unrecognized	run start time (of each phase, autoSize or simulate)	zhx type (cndtypes.def): LhSo, LhStH, ArSo, ArStH, ArStC, or (future) nv.
sp	–	X	number	hourly	setpoint if heat xfer is tstat controlled (SETTMP), else unused (hourly variability)
spPri	–	X	integer number	run start time (of each phase, autoSize or simulate)	setpoint priority: low #'s used first if setpoints equal, so can eg peg air heat b4 using local heat.
ui	–	X	integer number	run start time (of each phase, autoSize or simulate)	terminal TU subscript if a term cap type
zi	–	X	integer number	run start time (of each phase, autoSize or simulate)	zone ZNR subscript always – for term cap or vent zhx. When stable, just use ownTi?
ai	–	X	integer number	run start time (of each phase, autoSize or simulate)	0 or AH ss (subscript) of air handler supplying Ar zhx (copied from tu).
xiLh	–	X	integer number	run start time (of each phase, autoSize or simulate)	subscr of local heat ZHX for same terminal if any, else 0; not set for self.
xiArH	–	X	integer number	run start time (of each phase, autoSize or simulate)	was xiHeat. subscr of air heat or air set output ZHX for same terminal, if any, else 0
xiArC	–	X	integer number	run start time (of each phase, autoSize or simulate)	xiCool. subscr of air cool ZHX for same terminal, if any, else 0
nxZhx4z	–	X	integer number	run start time (of each phase, autoSize or simulate)	chain: 0 or subscript of next terminal zhx for this zone; 0?? if vent; head ZNR.zhx1.
nxZhxSt4z	–	X	integer number	hourly	chain: 0 or ss of next SETTMP zhx for this zone; head ZNR.zhx1St; kept sorted on sp/pri at runtime.

Name	Input?	Runtime?	Type	Variability	Description
nxZhx4a	–	X	integer number	run start time (of each phase, autoSize or simulate)	chain: 0 or subscript of next terminal zhx for this air handler; head AH.zhx1.
mda	–	X	integer number	hourly	for SETTMP, mode (mdSeq[] subscr) in which this is active (ctrl'd by its sp) ZHX.

6.53 @znRes[1..].

Name	Input?	Runtime?	Type	Variability	Description
name	–	X	string	constant	–
Y.n	–	X	unrecognized	end of run (of each phase, autoSize or simulate)	–
Y.nHrHeat	–	X	integer number	end of run (of each phase, autoSize or simulate)	–
Y.nHrCool	–	X	integer number	end of run (of each phase, autoSize or simulate)	–
Y.nHrFanv	–	X	integer number	end of run (of each phase, autoSize or simulate)	–
Y.nHrNatv	–	X	integer number	end of run (of each phase, autoSize or simulate)	–
Y.nHrCeilFan	–	X	integer number	end of run (of each phase, autoSize or simulate)	–
Y.nIter	–	X	number	end of run (of each phase, autoSize or simulate)	–
Y.nShUnMetH	–	X	number	end of run (of each phase, autoSize or simulate)	–
Y.nShUnMetC	–	X	number	end of run (of each phase, autoSize or simulate)	–

Name	Input?	Runtime?	Type	Variability	Description
Y.nHrUnMetH	–	X	number	end of run (of each phase, autoSize or simulate)	–
Y.nHrUnMetC	–	X	number	end of run (of each phase, autoSize or simulate)	–
Y.nShVentH	–	X	number	end of run (of each phase, autoSize or simulate)	–
Y.nSubhr	–	X	number	end of run (of each phase, autoSize or simulate)	–
Y.nSubhrLX	–	X	number	end of run (of each phase, autoSize or simulate)	–
Y.tAir	–	X	number	end of run (of each phase, autoSize or simulate)	–
Y.tRad	–	X	number	end of run (of each phase, autoSize or simulate)	–
Y.PMV7730	–	X	number	end of run (of each phase, autoSize or simulate)	–
Y.PPD7730	–	X	number	end of run (of each phase, autoSize or simulate)	–
Y.ivAirX	–	X	number	end of run (of each phase, autoSize or simulate)	–
Y.pz0	–	X	number	end of run (of each phase, autoSize or simulate)	–
Y.wAir	–	X	number	end of run (of each phase, autoSize or simulate)	–
Y.qCond	–	X	number	end of run (of each phase, autoSize or simulate)	–

Name	Input?	Runtime?	Type	Variability	Description
Y.qsInfil	–	X	number	end of run (of each phase, autoSize or simulate)	–
Y.qSlr	–	X	number	end of run (of each phase, autoSize or simulate)	–
Y.qsIg	–	X	number	end of run (of each phase, autoSize or simulate)	–
Y.qMass	–	X	number	end of run (of each phase, autoSize or simulate)	–
Y.qsIz	–	X	number	end of run (of each phase, autoSize or simulate)	–
Y.qsMech	–	X	number	end of run (of each phase, autoSize or simulate)	–
Y.eqfVentHr	–	X	number	end of run (of each phase, autoSize or simulate)	–
Y.qlInfil	–	X	number	end of run (of each phase, autoSize or simulate)	–
Y.qlIg	–	X	number	end of run (of each phase, autoSize or simulate)	–
Y.qlIz	–	X	number	end of run (of each phase, autoSize or simulate)	–
Y.qlAir	–	X	number	end of run (of each phase, autoSize or simulate)	–
Y.qlMech	–	X	number	end of run (of each phase, autoSize or simulate)	–
Y.qsBal	–	X	number	end of run (of each phase, autoSize or simulate)	–

Name	Input?	Runtime?	Type	Variability	Description
Y.qlBal	–	X	number	end of run (of each phase, autoSize or simulate)	–
Y.qlX	–	X	number	end of run (of each phase, autoSize or simulate)	–
Y.qcMech	–	X	number	end of run (of each phase, autoSize or simulate)	–
Y.qhMech	–	X	number	end of run (of each phase, autoSize or simulate)	–
Y.qvMech	–	X	number	end of run (of each phase, autoSize or simulate)	–
Y.litDmd	–	X	number	end of run (of each phase, autoSize or simulate)	–
Y.litEu	–	X	number	end of run (of each phase, autoSize or simulate)	–
M.n	–	X	unrecognized	end of each month	–
M.nHrHeat	–	X	integer number	end of each month	–
M.nHrCool	–	X	integer number	end of each month	–
M.nHrFanv	–	X	integer number	end of each month	–
M.nHrNatv	–	X	integer number	end of each month	–
M.nHrCeilFan	–	X	integer number	end of each month	–
M.nIter	–	X	number	end of each month	–
M.nShUnMetH	–	X	number	end of each month	–
M.nShUnMetC	–	X	number	end of each month	–
M.nHrUnMetH	–	X	number	end of each month	–
M.nHrUnMetC	–	X	number	end of each month	–
M.nShVentH	–	X	number	end of each month	–

Name	Input?	Runtime?	Type	Variability	Description
M.nSubhr	–	X	number	end of each month	–
M.nSubhrLX	–	X	number	end of each month	–
M.tAir	–	X	number	end of each month	–
M.tRad	–	X	number	end of each month	–
M.PMV7730	–	X	number	end of each month	–
M.PPD7730	–	X	number	end of each month	–
M.ivAirX	–	X	number	end of each month	–
M.pz0	–	X	number	end of each month	–
M.wAir	–	X	number	end of each month	–
M.qCond	–	X	number	end of each month	–
M.qsInfil	–	X	number	end of each month	–
M.qSlr	–	X	number	end of each month	–
M.qsIg	–	X	number	end of each month	–
M.qMass	–	X	number	end of each month	–
M.qsIz	–	X	number	end of each month	–
M.qsMech	–	X	number	end of each month	–
M.eqfVentHr	–	X	number	end of each month	–
M.qlInfil	–	X	number	end of each month	–
M.qlIg	–	X	number	end of each month	–
M.qlIz	–	X	number	end of each month	–
M.qlAir	–	X	number	end of each month	–
M.qlMech	–	X	number	end of each month	–
M.qsBal	–	X	number	end of each month	–
M.qlBal	–	X	number	end of each month	–
M.qlX	–	X	number	end of each month	–
M.qcMech	–	X	number	end of each month	–

Name	Input?	Runtime?	Type	Variability	Description
M.qhMech	–	X	number	end of each month	–
M.qvMech	–	X	number	end of each month	–
M.litDmd	–	X	number	end of each month	–
M.litEu	–	X	number	end of each month	–
D.n	–	X	unrecognized	end of each day	–
D.nHrHeat	–	X	integer number	end of each day	–
D.nHrCool	–	X	integer number	end of each day	–
D.nHrFanv	–	X	integer number	end of each day	–
D.nHrNatv	–	X	integer number	end of each day	–
D.nHrCeilFan	–	X	integer number	end of each day	–
D.nIter	–	X	number	end of each day	–
D.nShUnMetH	–	X	number	end of each day	–
D.nShUnMetC	–	X	number	end of each day	–
D.nHrUnMetH	–	X	number	end of each day	–
D.nHrUnMetC	–	X	number	end of each day	–
D.nShVentH	–	X	number	end of each day	–
D.nSubhr	–	X	number	end of each day	–
D.nSubhrLX	–	X	number	end of each day	–
D.tAir	–	X	number	end of each day	–
D.tRad	–	X	number	end of each day	–
D.PMV7730	–	X	number	end of each day	–
D.PPD7730	–	X	number	end of each day	–
D.ivAirX	–	X	number	end of each day	–
D.pz0	–	X	number	end of each day	–
D.wAir	–	X	number	end of each day	–
D.qCond	–	X	number	end of each day	–
D.qsInfil	–	X	number	end of each day	–
D.qSlr	–	X	number	end of each day	–
D.qsIg	–	X	number	end of each day	–
D.qMass	–	X	number	end of each day	–
D.qsIz	–	X	number	end of each day	–
D.qsMech	–	X	number	end of each day	–
D.eqfVentHr	–	X	number	end of each day	–
D.qlInfil	–	X	number	end of each day	–
D.qlIg	–	X	number	end of each day	–
D.qlIz	–	X	number	end of each day	–
D.qlAir	–	X	number	end of each day	–
D.qlMech	–	X	number	end of each day	–
D.qsBal	–	X	number	end of each day	–
D.qlBal	–	X	number	end of each day	–
D.qlX	–	X	number	end of each day	–
D.qcMech	–	X	number	end of each day	–
D.qhMech	–	X	number	end of each day	–
D.qvMech	–	X	number	end of each day	–
D.litDmd	–	X	number	end of each day	–
D.litEu	–	X	number	end of each day	–
H.n	–	X	unrecognized	end of each hour	–

Name	Input?	Runtime?	Type	Variability	Description
H.nHrHeat	–	X	integer number	end of each hour	–
H.nHrCool	–	X	integer number	end of each hour	–
H.nHrFanv	–	X	integer number	end of each hour	–
H.nHrNatv	–	X	integer number	end of each hour	–
H.nHrCeilFan	–	X	integer number	end of each hour	–
H.nIter	–	X	number	end of each hour	–
H.nShUnMetH	–	X	number	end of each hour	–
H.nShUnMetC	–	X	number	end of each hour	–
H.nHrUnMetH	–	X	number	end of each hour	–
H.nHrUnMetC	–	X	number	end of each hour	–
H.nShVentH	–	X	number	end of each hour	–
H.nSubhr	–	X	number	end of each hour	–
H.nSubhrLX	–	X	number	end of each hour	–
H.tAir	–	X	number	end of each hour	–
H.tRad	–	X	number	end of each hour	–
H.PMV7730	–	X	number	end of each hour	–
H.PPD7730	–	X	number	end of each hour	–
H.ivAirX	–	X	number	end of each hour	–
H.pz0	–	X	number	end of each hour	–
H.wAir	–	X	number	end of each hour	–
H.qCond	–	X	number	end of each hour	–
H.qsInfil	–	X	number	end of each hour	–
H.qSlr	–	X	number	end of each hour	–
H.qsIg	–	X	number	end of each hour	–
H.qMass	–	X	number	end of each hour	–
H.qsIz	–	X	number	end of each hour	–

Name	Input?	Runtime?	Type	Variability	Description
H.qsMech	–	X	number	end of each hour	–
H.eqfVentHr	–	X	number	end of each hour	–
H.qlInfil	–	X	number	end of each hour	–
H.qlIg	–	X	number	end of each hour	–
H.qlIz	–	X	number	end of each hour	–
H.qlAir	–	X	number	end of each hour	–
H.qlMech	–	X	number	end of each hour	–
H.qsBal	–	X	number	end of each hour	–
H.qlBal	–	X	number	end of each hour	–
H.qlX	–	X	number	end of each hour	–
H.qcMech	–	X	number	end of each hour	–
H.qhMech	–	X	number	end of each hour	–
H.qvMech	–	X	number	end of each hour	–
H.litDmd	–	X	number	end of each hour	–
H.litEu	–	X	number	end of each hour	–
S.n	–	X	unrecognized	end of each subhour	–
S.nHrHeat	–	X	integer number	end of each subhour	–
S.nHrCool	–	X	integer number	end of each subhour	–
S.nHrFanv	–	X	integer number	end of each subhour	–
S.nHrNatv	–	X	integer number	end of each subhour	–
S.nHrCeilFan	–	X	integer number	end of each subhour	–
S.nIter	–	X	number	end of each subhour	–
S.nShUnMetH	–	X	number	end of each subhour	–
S.nShUnMetC	–	X	number	end of each subhour	–
S.nHrUnMetH	–	X	number	end of each subhour	–
S.nHrUnMetC	–	X	number	end of each subhour	–

Name	Input?	Runtime?	Type	Variability	Description
S.nShVentH	–	X	number	end of each subhour	–
S.nSubhr	–	X	number	end of each subhour	–
S.nSubhrLX	–	X	number	end of each subhour	–
S.tAir	–	X	number	end of each subhour	–
S.tRad	–	X	number	end of each subhour	–
S.PMV7730	–	X	number	end of each subhour	–
S.PPD7730	–	X	number	end of each subhour	–
S.ivAirX	–	X	number	end of each subhour	–
S.pz0	–	X	number	end of each subhour	–
S.wAir	–	X	number	end of each subhour	–
S.qCond	–	X	number	end of each subhour	–
S.qsInfil	–	X	number	end of each subhour	–
S.qSlr	–	X	number	end of each subhour	–
S.qsIg	–	X	number	end of each subhour	–
S.qMass	–	X	number	end of each subhour	–
S.qsIz	–	X	number	end of each subhour	–
S.qsMech	–	X	number	end of each subhour	–
S.eqfVentHr	–	X	number	end of each subhour	–
S.qlInfil	–	X	number	end of each subhour	–
S.qlIg	–	X	number	end of each subhour	–
S.qlIz	–	X	number	end of each subhour	–
S.qlAir	–	X	number	end of each subhour	–
S.qlMech	–	X	number	end of each subhour	–
S.qsBal	–	X	number	end of each subhour	–
S.qlBal	–	X	number	end of each subhour	–
S.qlX	–	X	number	end of each subhour	–

Name	Input?	Runtime?	Type	Variability	Description
S.qcMech	–	X	number	end of each subhour	–
S.qhMech	–	X	number	end of each subhour	–
S.qvMech	–	X	number	end of each subhour	–
S.litDmd	–	X	number	end of each subhour	–
S.litEu	–	X	number	end of each subhour	–
prior.Y.n	–	X	unrecognized	run start time (of each phase, autoSize or simulate)	–
prior.Y.nHrHeat	–	X	integer number	run start time (of each phase, autoSize or simulate)	–
prior.Y.nHrCool	–	X	integer number	run start time (of each phase, autoSize or simulate)	–
prior.Y.nHrFanv	–	X	integer number	run start time (of each phase, autoSize or simulate)	–
prior.Y.nHrNatv	–	X	integer number	run start time (of each phase, autoSize or simulate)	–
prior.Y.nHrCeilFan	–	X	integer number	run start time (of each phase, autoSize or simulate)	–
prior.Y.nIter	–	X	number	run start time (of each phase, autoSize or simulate)	–
prior.Y.nShUnMetH	–	X	number	run start time (of each phase, autoSize or simulate)	–
prior.Y.nShUnMetC	–	X	number	run start time (of each phase, autoSize or simulate)	–
prior.Y.nHrUnMetH	–	X	number	run start time (of each phase, autoSize or simulate)	–

Name	Input?	Runtime?	Type	Variability	Description
prior.Y.nHrUnMetC	–	X	number	run start time (of each phase, autoSize or simulate)	–
prior.Y.nShVentH	–	X	number	run start time (of each phase, autoSize or simulate)	–
prior.Y.nSubhr	–	X	number	run start time (of each phase, autoSize or simulate)	–
prior.Y.nSubhrLX	–	X	number	run start time (of each phase, autoSize or simulate)	–
prior.Y.tAir	–	X	number	run start time (of each phase, autoSize or simulate)	–
prior.Y.tRad	–	X	number	run start time (of each phase, autoSize or simulate)	–
prior.Y.PMV7730	–	X	number	run start time (of each phase, autoSize or simulate)	–
prior.Y.PPD7730	–	X	number	run start time (of each phase, autoSize or simulate)	–
prior.Y.ivAirX	–	X	number	run start time (of each phase, autoSize or simulate)	–
prior.Y.pz0	–	X	number	run start time (of each phase, autoSize or simulate)	–
prior.Y.wAir	–	X	number	run start time (of each phase, autoSize or simulate)	–
prior.Y.qCond	–	X	number	run start time (of each phase, autoSize or simulate)	–
prior.Y.qsInfil	–	X	number	run start time (of each phase, autoSize or simulate)	–

Name	Input?	Runtime?	Type	Variability	Description
prior.Y.qSlr	–	X	number	run start time (of each phase, autoSize or simulate)	–
prior.Y.qsIg	–	X	number	run start time (of each phase, autoSize or simulate)	–
prior.Y.qMass	–	X	number	run start time (of each phase, autoSize or simulate)	–
prior.Y.qsIz	–	X	number	run start time (of each phase, autoSize or simulate)	–
prior.Y.qsMech	–	X	number	run start time (of each phase, autoSize or simulate)	–
prior.Y.eqfVentHr	–	X	number	run start time (of each phase, autoSize or simulate)	–
prior.Y.qlInfil	–	X	number	run start time (of each phase, autoSize or simulate)	–
prior.Y.qlIg	–	X	number	run start time (of each phase, autoSize or simulate)	–
prior.Y.qlIz	–	X	number	run start time (of each phase, autoSize or simulate)	–
prior.Y.qlAir	–	X	number	run start time (of each phase, autoSize or simulate)	–
prior.Y.qlMech	–	X	number	run start time (of each phase, autoSize or simulate)	–
prior.Y.qsBal	–	X	number	run start time (of each phase, autoSize or simulate)	–
prior.Y.qlBal	–	X	number	run start time (of each phase, autoSize or simulate)	–

Name	Input?	Runtime?	Type	Variability	Description
prior.Y.qlX	–	X	number	run start time (of each phase, autoSize or simulate)	–
prior.Y.qcMech	–	X	number	run start time (of each phase, autoSize or simulate)	–
prior.Y.qhMech	–	X	number	run start time (of each phase, autoSize or simulate)	–
prior.Y.qvMech	–	X	number	run start time (of each phase, autoSize or simulate)	–
prior.Y.litDmd	–	X	number	run start time (of each phase, autoSize or simulate)	–
prior.Y.litEu	–	X	number	run start time (of each phase, autoSize or simulate)	–
prior.M.n	–	X	unrecognized	monthly	–
prior.M.nHrHeat	–	X	integer number	monthly	–
prior.M.nHrCool	–	X	integer number	monthly	–
prior.M.nHrFanv	–	X	integer number	monthly	–
prior.M.nHrNatv	–	X	integer number	monthly	–
prior.M.nHrCeilFan	–	X	integer number	monthly	–
prior.M.nIter	–	X	number	monthly	–
prior.M.nShUnMetH	–	X	number	monthly	–
prior.M.nShUnMetC	–	X	number	monthly	–
prior.M.nHrUnMetH	–	X	number	monthly	–
prior.M.nHrUnMetC	–	X	number	monthly	–
prior.M.nShVentH	–	X	number	monthly	–
prior.M.nSubhr	–	X	number	monthly	–
prior.M.nSubhrLX	–	X	number	monthly	–
prior.M.tAir	–	X	number	monthly	–
prior.M.tRad	–	X	number	monthly	–
prior.M.PMV7730	–	X	number	monthly	–
prior.M.PPD7730	–	X	number	monthly	–
prior.M.ivAirX	–	X	number	monthly	–
prior.M.pz0	–	X	number	monthly	–
prior.M.wAir	–	X	number	monthly	–
prior.M.qCond	–	X	number	monthly	–
prior.M.qsInfil	–	X	number	monthly	–
prior.M.qSlr	–	X	number	monthly	–
prior.M.qsIg	–	X	number	monthly	–
prior.M.qMass	–	X	number	monthly	–
prior.M.qsIz	–	X	number	monthly	–
prior.M.qsMech	–	X	number	monthly	–

Name	Input?	Runtime?	Type	Variability	Description
prior.M.eqfVentHr	–	X	number	monthly	–
prior.M.qlInfil	–	X	number	monthly	–
prior.M.qlIg	–	X	number	monthly	–
prior.M.qlIz	–	X	number	monthly	–
prior.M.qlAir	–	X	number	monthly	–
prior.M.qlMech	–	X	number	monthly	–
prior.M.qsBal	–	X	number	monthly	–
prior.M.qlBal	–	X	number	monthly	–
prior.M.qlX	–	X	number	monthly	–
prior.M.qcMech	–	X	number	monthly	–
prior.M.qhMech	–	X	number	monthly	–
prior.M.qvMech	–	X	number	monthly	–
prior.M.litDmd	–	X	number	monthly	–
prior.M.litEu	–	X	number	monthly	–
prior.D.n	–	X	unrecognized	daily	–
prior.D.nHrHeat	–	X	integer number	daily	–
prior.D.nHrCool	–	X	integer number	daily	–
prior.D.nHrFanv	–	X	integer number	daily	–
prior.D.nHrNatv	–	X	integer number	daily	–
prior.D.nHrCeilFan	–	X	integer number	daily	–
prior.D.nIter	–	X	number	daily	–
prior.D.nShUnMetH	–	X	number	daily	–
prior.D.nShUnMetC	–	X	number	daily	–
prior.D.nHrUnMetH	–	X	number	daily	–
prior.D.nHrUnMetC	–	X	number	daily	–
prior.D.nShVentH	–	X	number	daily	–
prior.D.nSubhr	–	X	number	daily	–
prior.D.nSubhrLX	–	X	number	daily	–
prior.D.tAir	–	X	number	daily	–
prior.D.tRad	–	X	number	daily	–
prior.D.PMV7730	–	X	number	daily	–
prior.D.PPD7730	–	X	number	daily	–
prior.D.ivAirX	–	X	number	daily	–
prior.D.pz0	–	X	number	daily	–
prior.D.wAir	–	X	number	daily	–
prior.D.qCond	–	X	number	daily	–
prior.D.qsInfil	–	X	number	daily	–
prior.D.qSlr	–	X	number	daily	–
prior.D.qsIg	–	X	number	daily	–
prior.D.qMass	–	X	number	daily	–
prior.D.qsIz	–	X	number	daily	–
prior.D.qsMech	–	X	number	daily	–
prior.D.eqfVentHr	–	X	number	daily	–
prior.D.qlInfil	–	X	number	daily	–
prior.D.qlIg	–	X	number	daily	–
prior.D.qlIz	–	X	number	daily	–
prior.D.qlAir	–	X	number	daily	–
prior.D.qlMech	–	X	number	daily	–
prior.D.qsBal	–	X	number	daily	–
prior.D.qlBal	–	X	number	daily	–
prior.D.qlX	–	X	number	daily	–
prior.D.qcMech	–	X	number	daily	–

Name	Input?	Runtime?	Type	Variability	Description
prior.D.qhMech	–	X	number	daily	–
prior.D.qvMech	–	X	number	daily	–
prior.D.litDmd	–	X	number	daily	–
prior.D.litEu	–	X	number	daily	–
prior.H.n	–	X	unrecognized	hourly	–
prior.H.nHrHeat	–	X	integer number	hourly	–
prior.H.nHrCool	–	X	integer number	hourly	–
prior.H.nHrFanv	–	X	integer number	hourly	–
prior.H.nHrNatv	–	X	integer number	hourly	–
prior.H.nHrCeilFan	–	X	integer number	hourly	–
prior.H.nIter	–	X	number	hourly	–
prior.H.nShUnMetH	–	X	number	hourly	–
prior.H.nShUnMetC	–	X	number	hourly	–
prior.H.nHrUnMetH	–	X	number	hourly	–
prior.H.nHrUnMetC	–	X	number	hourly	–
prior.H.nShVentH	–	X	number	hourly	–
prior.H.nSubhr	–	X	number	hourly	–
prior.H.nSubhrLX	–	X	number	hourly	–
prior.H.tAir	–	X	number	hourly	–
prior.H.tRad	–	X	number	hourly	–
prior.H.PMV7730	–	X	number	hourly	–
prior.H.PPD7730	–	X	number	hourly	–
prior.H.ivAirX	–	X	number	hourly	–
prior.H.pz0	–	X	number	hourly	–
prior.H.wAir	–	X	number	hourly	–
prior.H.qCond	–	X	number	hourly	–
prior.H.qsInfil	–	X	number	hourly	–
prior.H.qSlr	–	X	number	hourly	–
prior.H.qsIg	–	X	number	hourly	–
prior.H.qMass	–	X	number	hourly	–
prior.H.qsIz	–	X	number	hourly	–
prior.H.qsMech	–	X	number	hourly	–
prior.H.eqfVentHr	–	X	number	hourly	–
prior.H.qlInfil	–	X	number	hourly	–
prior.H.qlIg	–	X	number	hourly	–
prior.H.qlIz	–	X	number	hourly	–
prior.H.qlAir	–	X	number	hourly	–
prior.H.qlMech	–	X	number	hourly	–
prior.H.qsBal	–	X	number	hourly	–
prior.H.qlBal	–	X	number	hourly	–
prior.H.qlX	–	X	number	hourly	–
prior.H.qcMech	–	X	number	hourly	–
prior.H.qhMech	–	X	number	hourly	–
prior.H.qvMech	–	X	number	hourly	–
prior.H.litDmd	–	X	number	hourly	–
prior.H.litEu	–	X	number	hourly	–
prior.S.n	–	X	unrecognized	subhourly	–
prior.S.nHrHeat	–	X	integer number	subhourly	–
prior.S.nHrCool	–	X	integer number	subhourly	–
prior.S.nHrFanv	–	X	integer number	subhourly	–
prior.S.nHrNatv	–	X	integer number	subhourly	–
prior.S.nHrCeilFan	–	X	integer number	subhourly	–

Name	Input?	Runtime?	Type	Variability	Description
prior.S.nIter	–	X	number	subhourly	–
prior.S.nShUnMetH	–	X	number	subhourly	–
prior.S.nShUnMetC	–	X	number	subhourly	–
prior.S.nHrUnMetH	–	X	number	subhourly	–
prior.S.nHrUnMetC	–	X	number	subhourly	–
prior.S.nShVentH	–	X	number	subhourly	–
prior.S.nSubhr	–	X	number	subhourly	–
prior.S.nSubhrLX	–	X	number	subhourly	–
prior.S.tAir	–	X	number	subhourly	–
prior.S.tRad	–	X	number	subhourly	–
prior.S.PMV7730	–	X	number	subhourly	–
prior.S.PPD7730	–	X	number	subhourly	–
prior.S.ivAirX	–	X	number	subhourly	–
prior.S.pz0	–	X	number	subhourly	–
prior.S.wAir	–	X	number	subhourly	–
prior.S.qCond	–	X	number	subhourly	–
prior.S.qsInfil	–	X	number	subhourly	–
prior.S.qSlr	–	X	number	subhourly	–
prior.S.qsIg	–	X	number	subhourly	–
prior.S.qMass	–	X	number	subhourly	–
prior.S.qsIz	–	X	number	subhourly	–
prior.S.qsMech	–	X	number	subhourly	–
prior.S.eqfVentHr	–	X	number	subhourly	–
prior.S.qlInfil	–	X	number	subhourly	–
prior.S.qlIg	–	X	number	subhourly	–
prior.S.qlIz	–	X	number	subhourly	–
prior.S.qlAir	–	X	number	subhourly	–
prior.S.qlMech	–	X	number	subhourly	–
prior.S.qsBal	–	X	number	subhourly	–
prior.S.qlBal	–	X	number	subhourly	–
prior.S.qlX	–	X	number	subhourly	–
prior.S.qcMech	–	X	number	subhourly	–
prior.S.qhMech	–	X	number	subhourly	–
prior.S.qvMech	–	X	number	subhourly	–
prior.S.litDmd	–	X	number	subhourly	–
prior.S.litEu	–	X	number	subhourly	–

6.54 @zone[1..].

Name	Input?	Runtime?	Type	Variability	Description
name	X	X	string	constant	–
znModel	X	X	integer number	input time	–
znArea	X	X	number	input time	–
znVol	X	X	number	input time	–
floorZ	X	X	number	input time	–
ceilingHt	X	X	number	run start time (of each phase, autoSize or simulate)	–
znCAir	X	X	number	input time	–

Name	Input?	Runtime?	Type	Variability	Description
HIRatio	X	X	number	run start time (of each phase, autoSize or simulate)	–
znAzm	X	X	number	input time	–
plenumRet	X	X	integer number	input time	–
znSC	X	X	number	hourly	–
znTH	X	X	number	hourly	–
znTD	X	X	number	hourly	–
znTC	X	X	number	hourly	–
znQMxH	X	X	number	hourly	–
znQMxHRated	X	X	number	run start time (of each phase, autoSize or simulate)	–
znQMxC	X	X	number	hourly	–
znQMxCRated	X	X	number	run start time (of each phase, autoSize or simulate)	–
rsi	X	X	integer number	run start time (of each phase, autoSize or simulate)	–
hcFrcF	X	X	number	hourly	–
hcAirX	X	X	number	end of each subhour	–
hcAirXIsSet	X	X	unrecognized	run start time (of each phase, autoSize or simulate)	–
znComfClo	X	X	number	subhourly	–
znComfMet	X	X	number	subhourly	–
znComfAirV	X	X	number	subhourly	–
znComfRh	X	X	number	subhourly	–
znComfUseZoneRH	X	X	unrecognized	run start time (of each phase, autoSize or simulate)	–
xfanFOn	X	X	number	hourly	–
xfan.fanTy	X	X	unrecognized	autosize and simulate phase start time	–
xfan.vfDs	X	X	number	end of each subhour	–
xfan.vfDs_As	X	X	number	autosize and simulate phase start time	–
xfan.vfDs_AsNov	X	X	number	autosize and simulate phase start time	–

Name	Input?	Runtime?	Type	Variability	Description
xfan.vfMxF	X	X	number	autosize and simulate phase start time	–
xfan.press	X	X	number	run start time (of each phase, autoSize or simulate)	–
xfan.eff	X	X	number	run start time (of each phase, autoSize or simulate)	–
xfan.shaftPwr	X	X	number	run start time (of each phase, autoSize or simulate)	–
xfan.elecPwr	X	X	number	run start time (of each phase, autoSize or simulate)	–
xfan.motTy	X	X	unrecognized	run start time (of each phase, autoSize or simulate)	–
xfan.motEff	X	X	number	autosize and simulate phase start time	–
xfan.motPos	X	X	unrecognized	autosize and simulate phase start time	–
xfan.curvePy.k[0]	X	X	number	autosize and simulate phase start time	–
xfan.curvePy.k[1]	X	X	number	autosize and simulate phase start time	–
xfan.curvePy.k[2]	X	X	number	autosize and simulate phase start time	–
xfan.curvePy.k[3]	X	X	number	autosize and simulate phase start time	–
xfan.curvePy.k[4]	X	X	number	autosize and simulate phase start time	–
xfan.curvePy.k[5]	X	X	number	autosize and simulate phase start time	–
xfan.mtri	X	X	integer number	input time	–
xfan.endUse	X	X	integer number	autosize and simulate phase start time	–

Name	Input?	Runtime?	Type	Variability	Description
xfan.ausz	X	X	integer number	run start time (of each phase, autoSize or simulate)	–
xfan.outPower	X	X	number	subhourly	–
xfan.airPower	X	X	number	subhourly	–
xfan.cMx	X	X	number	end of each subhour	–
xfan.c	X	X	number	end of each subhour	–
xfan.t	X	X	number	end of each subhour	–
xfan.frOn	X	X	number	end of each subhour	–
xfan.p	X	X	number	end of each subhour	–
xfan.q	X	X	number	end of each subhour	–
xfan.dT	X	X	number	end of each subhour	–
xfan.qAround	X	X	number	end of each subhour	–
infAC	X	X	number	hourly	–
infELA	X	X	number	hourly	–
infShld	X	X	integer number	input time	–
infStories	X	X	integer number	input time	–
eaveZ	X	X	number	run start time (of each phase, autoSize or simulate)	–
windFLkg	X	X	number	subhourly	–
vrZdd	X	X	unrecognized	run start time (of each phase, autoSize or simulate)	–
xsurf1	–	X	integer number	run start time (of each phase, autoSize or simulate)	–
xsSpecT1	–	X	integer number	run start time (of each phase, autoSize or simulate)	–
tu1	–	X	integer number	run start time (of each phase, autoSize or simulate)	–
zhx1	–	X	integer number	run start time (of each phase, autoSize or simulate)	–

Name	Input?	Runtime?	Type	Variability	Description
zhx1St	–	X	integer number	run start time (of each phase, autoSize or simulate)	–
znSCF	–	X	integer number	run start time (of each phase, autoSize or simulate)	–
stackc	–	X	number	run start time (of each phase, autoSize or simulate)	–
windc	–	X	number	run start time (of each phase, autoSize or simulate)	–
rIgDistNAl	–	X	integer number	run start time (of each phase, autoSize or simulate)	–
rIgDistN	–	X	integer number	run start time (of each phase, autoSize or simulate)	–
rIgDist	–	X	unrecognized	run start time (of each phase, autoSize or simulate)	–
surfA	–	X	number	run start time (of each phase, autoSize or simulate)	–
surfASlr	–	X	number	run start time (of each phase, autoSize or simulate)	–
ductA	–	X	number	run start time (of each phase, autoSize or simulate)	–
surfEpsLWAvg	–	X	number	run start time (of each phase, autoSize or simulate)	–
airRadXC1	–	X	number	run start time (of each phase, autoSize or simulate)	–
airRadXC2	–	X	number	run start time (of each phase, autoSize or simulate)	–

Name	Input?	Runtime?	Type	Variability	Description
airRadXArea	–	X	number	run start time (of each phase, autoSize or simulate)	–
FAir	–	X	number	run start time (of each phase, autoSize or simulate)	–
airCxF	–	X	number	end of each hour	–
airCx	–	X	number	end of each subhour	–
rmTrans[0]	–	X	number	end of each hour on 1st day of month/run	–
rmTrans[1]	–	X	number	end of each hour on 1st day of month/run	–
rmAbs	–	X	number	end of each hour on 1st day of month/run	–
adjRmAbs[0]	–	X	number	end of each hour on 1st day of month/run	–
adjRmAbs[1]	–	X	number	end of each hour on 1st day of month/run	–
rmAbsCAir	–	X	number	end of each hour on 1st day of month/run	–
cavAbsCAir[0]	–	X	number	end of each hour on 1st day of month/run	–
cavAbsCAir[1]	–	X	number	end of each hour on 1st day of month/run	–
sgfCavBm[0]	–	X	number	end of each hour on 1st day of month/run	–
sgfCavBm[1]	–	X	number	end of each hour on 1st day of month/run	–
sgfCavDf[0]	–	X	number	end of each hour on 1st day of month/run	–
sgfCavDf[1]	–	X	number	end of each hour on 1st day of month/run	–
sgSaBm[0]	–	X	number	end of each hour on 1st day of month/run	–

Name	Input?	Runtime?	Type	Variability	Description
sgSaBm[1]	–	X	number	end of each hour on 1st day of month/run	–
sgSaDf[0]	–	X	number	end of each hour on 1st day of month/run	–
sgSaDf[1]	–	X	number	end of each hour on 1st day of month/run	–
sgfCAirBm[0]	–	X	number	end of each hour on 1st day of month/run	–
sgfCAirBm[1]	–	X	number	end of each hour on 1st day of month/run	–
sgfCAirDf[0]	–	X	number	end of each hour on 1st day of month/run	–
sgfCAirDf[1]	–	X	number	end of each hour on 1st day of month/run	–
uaSpecT	–	X	number	run start time (of each phase, autoSize or simulate)	–
ua	–	X	number	run start time (of each phase, autoSize or simulate)	–
UANom	–	X	number	run start time (of each phase, autoSize or simulate)	–
ductCondUANom	–	X	number	run start time (of each phase, autoSize or simulate)	–
haMass	–	X	number	run start time (of each phase, autoSize or simulate)	–
BGWallPerim	–	X	number	run start time (of each phase, autoSize or simulate)	–
BGWallPA4	–	X	number	run start time (of each phase, autoSize or simulate)	–

Name	Input?	Runtime?	Type	Variability	Description
BGWallPA5	–	X	number	run start time (of each phase, autoSize or simulate)	–
qSgTot	–	X	number	end of each hour	–
sgTotTarg.bm	–	X	number	end of each subhour	–
sgTotTarg.df	–	X	number	end of each subhour	–
sgTotTarg.tot	–	X	number	end of each subhour	–
qrIgTot	–	X	unrecognized	end of each hour	–
qrIgTotO	–	X	unrecognized	end of each hour	–
qrIgTotIz	–	X	unrecognized	end of each hour	–
qrIgAir	–	X	unrecognized	end of each hour	–
qrIgMs	–	X	number	end of each hour	–
znSGain	–	X	number	end of each hour	–
znLGain	–	X	number	end of each hour	–
znLitDmd	–	X	number	end of each hour	–
znLitEu	–	X	number	end of each hour	–
znXLGain	–	X	number	end of each subhour	–
znXLGainLs	–	X	number	end of each subhour	–
bcon	–	X	number	run start time (of each phase, autoSize or simulate)	–
qMsSg	–	X	number	end of each subhour	–
qSgAir	–	X	number	end of each subhour	–
sgAirTarg.bm	–	X	number	end of each subhour	–
sgAirTarg.df	–	X	number	end of each subhour	–
sgAirTarg.tot	–	X	number	end of each subhour	–
qSgTotSh	–	X	number	end of each subhour	–
sgTotShTarg.bm	–	X	number	end of each subhour	–
sgTotShTarg.df	–	X	number	end of each subhour	–
sgTotShTarg.tot	–	X	number	end of each subhour	–
qIzXAnSh	–	X	number	end of each subhour	–
qIzSh	–	X	number	end of each subhour	–
pz0W[0]	–	X	number	end of each subhour	–

Name	Input?	Runtime?	Type	Variability	Description
pz0W[1]	–	X	number	end of each subhour	–
pz0	–	X	number	end of each subhour	–
ventUt	–	X	unrecognized	end of each subhour	–
qDuctCondAir	–	X	number	end of each subhour	–
qDuctCondRad	–	X	number	end of each subhour	–
qDuctCond	–	X	number	end of each subhour	–
qDHWLossAir	–	X	number	end of each subhour	–
qDHWLossRad	–	X	number	end of each subhour	–
qDHWLoss	–	X	number	end of each subhour	–
qHPWH	–	X	number	end of each subhour	–
hpwhAirX	–	X	number	end of each subhour	–
airNetI[0].tdb	–	X	number	end of each subhour	–
airNetI[0].w	–	X	number	end of each subhour	–
airNetI[0].amf	–	X	number	end of each subhour	–
airNetI[1].tdb	–	X	number	end of each subhour	–
airNetI[1].w	–	X	number	end of each subhour	–
airNetI[1].amf	–	X	number	end of each subhour	–
fVent	–	X	number	end of each subhour	–
tzVent	–	X	number	end of each subhour	–
anAmfCpVent	–	X	number	end of each subhour	–
anAmfCpTVent	–	X	number	end of each subhour	–
ductLkI.tdb	–	X	number	end of each subhour	–
ductLkI.w	–	X	number	end of each subhour	–
ductLkI.amf	–	X	number	end of each subhour	–
ductLkO.tdb	–	X	number	end of each subhour	–
ductLkO.w	–	X	number	end of each subhour	–

Name	Input?	Runtime?	Type	Variability	Description
ductLkO.amf	–	X	number	end of each subhour	–
sysAirI.tdb	–	X	number	end of each subhour	–
sysAirI.w	–	X	number	end of each subhour	–
sysAirI.amf	–	X	number	end of each subhour	–
sysAirO.tdb	–	X	number	end of each subhour	–
sysAirO.w	–	X	number	end of each subhour	–
sysAirO.amf	–	X	number	end of each subhour	–
OAVRlfO.tdb	–	X	number	end of each subhour	–
OAVRlfO.w	–	X	number	end of each subhour	–
OAVRlfO.amf	–	X	number	end of each subhour	–
sysDepAirIls.tdb	–	X	number	end of each subhour	–
sysDepAirIls.w	–	X	number	end of each subhour	–
sysDepAirIls.amf	–	X	number	end of each subhour	–
qCondQS	–	X	number	end of each subhour	–
qCondMS	–	X	number	end of each subhour	–
rsAmfSysReq[0]	–	X	number	end of each subhour	–
rsAmfSysReq[1]	–	X	number	end of each subhour	–
rsFSize	–	X	number	end of each subhour	–
rsAmfSup	–	X	number	end of each subhour	–
rsAmfRet	–	X	number	end of each subhour	–
rsAmfRetLs	–	X	number	subhourly	–
tzsp	–	X	number	end of each subhour	–
hcMode	–	X	integer number	end of each subhour	–
unMetH	–	X	unrecognized	end of each subhour	–
unMetC	–	X	unrecognized	end of each subhour	–
fConvH	–	X	number	subhourly	–
fConvC	–	X	number	subhourly	–
fConv	–	X	number	subhourly	–

Name	Input?	Runtime?	Type	Variability	Description
comfPMV7730	–	X	number	end of each subhour	–
comfPPD7730	–	X	number	end of each subhour	–
qsHvac	–	X	number	end of each subhour	–
qlHvac	–	X	number	end of each subhour	–
qlIz	–	X	number	end of each subhour	–
wCase	–	X	number	end of each subhour	–
airMode	–	X	number	end of each subhour	–
rho	–	X	number	end of each subhour	–
rho0	–	X	number	end of each subhour	–
rho0ls	–	X	number	subhourly	–
dryAirMass	–	X	number	end of each subhour	–
dryAirMassEff	–	X	number	end of each subhour	–
ivAirX	–	X	number	end of each subhour	–
airX	–	X	number	end of each subhour	–
hcAirXls	–	X	number	subhourly	–
hcFrc	–	X	number	subhourly	–
windPresV	–	X	number	subhourly	–
tz	–	X	number	end of each subhour	–
aTz	–	X	number	end of each subhour	–
wz	–	X	number	end of each subhour	–
relHum	–	X	number	end of each subhour	–
twb	–	X	number	end of each subhour	–
aWz	–	X	number	end of each subhour	–
tzls	–	X	number	subhourly	–
wzls	–	X	number	subhourly	–
tzlh	–	X	number	hourly	–
tzlsDelta	–	X	number	constant	–
wzlsDelta	–	X	number	constant	–
tr	–	X	number	end of each subhour	–
trls	–	X	number	end of each subhour	–
trlh	–	X	number	hourly	–

Name	Input?	Runtime?	Type	Variability	Description
md	–	X	integer number	end of each subhour	–