BUILDING DESCRIPTION LANGUAGE (BDL)

BDL Instructions

In this manual, the acronym BDL is used both as the name of the DOE-2 input language and as the name of the DOE-2 subprogram that translates DOE-2 input instructions into a machine readable format. For a list of *basic* BDL input commands and keywords, see Appendix F of this manual. A complete list of commands and keywords can be found in the *BDL Summary (2.1E).*

To aid in understanding the following sections, we recommend that you refer to the "Sample Input", starting on p.7.

The primary element of BDL is the *instruction*. An instruction corresponds roughly to an English sentence and always includes a *command* that specifies the subject matter, and a *terminator* that ends the command. An example is the BDL instruction that defines the run period for the simulation (see "Sample Input"):

```
RUN-PERIOD   JAN 1 1974   THRU   DEC 31 1974 ..
```

Each instruction is assumed by BDL to stay in effect until it sees a terminator. If there is more data, it assumes a new instruction is coming and seeks out the controlling command word of that instruction. This process continues until BDL reads the STOP command. Note that no reference is made here to coming to the end of a line. An instruction may stretch over many lines without your needing to indicate that the second and following lines are continuations of the first. BDL assumes that they are continuations as long as it has not discovered a terminator, and thus allows you to arrange the input any way you wish.

Terminator

The symbol for the *terminator* is .. (two periods with no space between them and one or more blank spaces preceding them. *Don't forget to end each instruction with a terminator*; otherwise, pages of error messages may result.

Comments

Any line of input may contain a comment; comment lines start and end with a dollar sign ($). Be sure to put at least one space between the terminator and the start of the comment line (.. $comment).

INPUT Command

Before inputting data for a subprogram, you have to use the INPUT command to tell BDL which program data are being presented next; for example:

```
INPUT LOADS   ..
```

END Command

END tells BDL that the program data started by INPUT command is complete. The lines between INPUT and END must contain all data needed to perform the simulation for the subprogram indicated.

COMPUTE Command

COMPUTE requests that a simulation be performed using input data between the previous INPUT and END commands. In the sample, COMPUTE LOADS requests a LOADS simulation and that the LOADS data between INPUT LOADS and END be used. The sequence continues by telling BDL to accept SYSTEMS data input (INPUT SYSTEMS).

Keywords*

The description of a building is entered between INPUT and END with a series of commands, each of which accepts additional pieces of information that describe the content of the command. The identification of the specific content is through *keywords.*

Keywords always appear in the form keyword=value, or keyword=(a list of values), where you specify a "value" or "list of values". As an example, the BUILDING-LOCATION command tells BDL that the data to follow give the building's location and time zone.

```
     BUILDING-LOCATION   LATITUDE=42     LONGITUDE=88
                         ALTITUDE=610    TIME-ZONE=6
                         AZIMUTH=0       HOLIDAY=NO ..
```

In this example there are six keyword=value pairs (LATITUDE=42, LONGITUDE=88, ALTITUDE=610, TIME-ZONE=6, AZIMUTH=0, and HOLIDAY=NO).

Spacing between lines, commands, keywords, etc., is arbitrary except that a blank indicates the end of a keyword=value pair. For this reason blank spaces may not be embedded within a single keyword. For example, the keyword TIME-ZONE is recognized as one word; if the dash is omitted (TIME ZONE), then two words are produced and BDL won't recognize either one as a keyword. Because spaces, commas, and equal signs may be used interchangeably as separators, BDL interprets

```
LATITUDE 42
LATITUDE,42
LATITUDE=42
```

all in the same way. However, equal signs between keywords and their values do make the input more readable. Note that for the keyword, HOLIDAY, the value assigned (NO) is a code-word rather than a number.

U-Names and Referenced Commands

Some keywords take values that are user-defined names, called "u-names". With u-names, previously-defined commands can be referenced, allowing data from one instruction to be used in one or more subsequent instructions.

To illustrate the use of referenced commands and u-names, we specify the construction of a wall. The first step is to indicate the different layers of the wall starting from the outside surface. This is given by an instruction whose command is LAYERS, which must be given a u-name, in this case WA-1-2.

```
    WA-1-2 =LAYERS    MATERIAL=(WD01,PW03,IH02,GP01) ..
```

* Please refer to Appendix E for a list of *"basic"* commands, keywords, and abbreviations.

Note that, in general, a command must be the first word in an instruction unless it is preceded by a u-name (preferably with an intervening equal sign).* There are two other points to note in this example. First: when a list of values is assigned to a keyword, the list must be enclosed in parentheses ( ). Second: the values of some keywords are code-words; in this example they were taken from the list of materials described in Appendix D, the Materials Library.

Since the LAYERS command has a u-name, it can be referenced in a subsequent instruction that describes the unique construction of a wall. Its value is the u-name you have given to the set of materials above. Thus

```
WALL-1 =CONSTRUCTION    LAYERS=WA-1-2  ..
```

Here, the CONSTRUCTION command has been given the u-name WALL-1 to remind you that it describes the construction type of the exterior wall, and so that it can be referenced later.

To complete the chain of referenced commands, the south-facing wall of the building has an exterior wall with construction WALL-1:

```
FRONT-1  =EXTERIOR-WALL    HEIGHT=8
                           WIDTH=100
                           AZIMUTH=180
                           CONSTRUCTION=WALL-1  ..
```

Choosing u-names
In the example above, the command EXTERIOR-WALL has also been given a u-name, FRONT-1. This is optional and is *not* required. However, there are reasons for u-naming specific walls, windows and the like. The first is that several of the optional reports in the LOADS subprogram are verifications of input organized in an informative manner. Unless the various components are u-named, it is difficult to tell which wall, for example, is being described. Another reason is to make use of the labor-saving keyword LIKE, which is described below. The rule for choosing u-names is this: a u-name is any alpha-numeric string of 16 or fewer symbols which have no embedded spaces and that are different from all commands, keywords and code-words or their corresponding abbreviations or reserved words. "Reserved" means that the word is recognized by the program as a command or keyword or value.

LIKE Keyword
Many commands allow the LIKE keyword. When used, LIKE must be the first keyword following the command and its symbolic value must be the u-name of a previously defined command of the same type. This keyword instructs BDL to assign to this command the same values of all the keywords in the referenced command. For example,

```
WF-1=WINDOW    HEIGHT=4
               WIDTH=45
               GLASS-TYPE=WINDOW-1  ..
```

allows us to reference this window and change its width, creating a new window, WR-1:

```
WR-1=WINDOW  LIKE=WF-1  WIDTH=25  ..
```

* There are some commands, like BUILDING-LOCATION, which cannot have u-names. The complete list of commands, keywords, their abbreviations, and summary rules is given in the *BDL Summary (2.1E)*.

Subcommands

A *subcommand* is similar to a command except that it can be referred to by a subsequent command through the use of a u-name. Unlike a command, however, the keywords of a subcommand can be included within its associated command. In the example, SPACE–CONDITIONS keywords are referenced in the SPACE command by specifying SPACE–CONDITIONS=OFFICE–ENV. In a multi-space building, this would allow the same SPACE–CONDITIONS to be assigned to several spaces, thus saving input effort.

Building Description

One of the first steps in energy analysis is to obtain the architectural and mechanical drawings for the building to be simulated. Keep in mind that the goal is to create a model of the building in order to analyze thermal energy flows and not to describe in minute detail what the building looks like architecturally. You can save input time and computer time by describing the building from an energy perspective rather than from an architectural perspective.

To understand this more fully, it is necessary to know how DOE-2 treats the boundaries of spaces. DOE-2 does *not* attempt to reconstruct the space geometrically from the your description of the bounding surfaces. Rather, the program calculates the flow of energy *only through the surfaces you describe.* It does not test whether walls meet or even whether the surfaces describe an enclosed three-dimensional space. It is possible, for example, to define a space with a floor area and a volume and then to describe only one exterior south-facing wall. DOE-2 accepts your word that all you wanted is to examine the energy flow in the space through that one surface. Or you may have decided that the energy flow through the other surfaces (perhaps interior walls) was negligible.

Internal Zoning

The first decision to make is how to divide the inside of the building into discrete spaces or zones. In LOADS these regions are referred to as "spaces" and in SYSTEMS these identical regions are called "zones". When considering the structure and use of a building the word "space" seems appropriate; however, when designing an HVAC system the central concern relates to spaces that are under the same thermostatic control, i.e., zones.

In practical terms this means that you need not be constrained by the details of the architectural plan. Contiguous rooms, that can be expected to behave similarly from a thermodynamic perspective, can be described as a single SPACE in the LOADS input and as a single ZONE (with the same u-name) in the SYSTEMS input. The objective from the perspective of reducing input preparation time and computer run time is to have as few zones as possible consistent with making an adequate model of the thermodynamic behavior of the building.

It is not even necessary to have zones separated by real partitions or interior walls. However, it is common in building energy analysis to create one internal zone and four external zones (one for each exposure — see Appendix B).

In the Example, we have chosen to input the building as one SPACE in LOADS and one ZONE in SYSTEMS. Note that if the dynamics of a building system require transfer of heat from one zone to another, such as a water source heat pump system, it is imperative to input multiple zones so that the transfer can be simulated.

Use of Comments

BDL allows you to introduce comments into the body of the input without affecting the translation of the data. Comments help when you return to DOE-2 input after an absence and may have difficulty reconstructing the original intent. Comments are also helpful if someone else wants to use the input.

BDL recognizes the dollar sign, $, as the beginning and end of a comment. Any string of characters between dollar signs is ignored by BDL in translation but is echoed back in the output. For example, $BUILT-UP ROOF$ is a comment in the following instruction:

```
ROOF-1 =CONSTRUCTION    LAYERS=RB-1-1   $BUILT-UP ROOF$ ..
```

Note that a comment that spans several lines must have $ at the beginning of each line of comment.

Alternative Runs

DOE-2 allows you to make a series of alternative runs in a single input file.* Any number or series of alternative runs is allowed in DOE-2 and in any combination. For example, the input of a base LOADS run with two SYSTEMS alternatives, followed by a single PLANT, is as follows:

```
INPUT LOADS   ..
   .
   .
   .
END           ..
COMPUTE LOADS ..
INPUT SYSTEMS  ..      $ FIRST SYSTEM $
   .
   .
   .
END           ..
COMPUTE SYSTEMS ..
INPUT PLANT    ..
   .
   .
   .
END           ..
COMPUTE PLANT  ..
INPUT SYSTEMS  ..      $ SECOND SYSTEM $
   .
   .
   .
END           ..
COMPUTE SYSTEMS ..
COMPUTE PLANT  ..
STOP          ..
```

It is not necessary to re-input the LOADS input after the first SYSTEM, nor is it necessary to re-input the PLANT input after the last SYSTEM; the command, COMPUTE PLANT, is sufficient to re-call the previous PLANT input and thus recompute the effect of the second SYSTEM on it.

* DOE-2 also allows you to make parametric runs, i.e., those that involve changing one parameter (or a number of related parameters) to analyze the effect on energy use. You are referred to the *Reference Manual (2.1A)* for a description of how to prepare the input for parametric runs.

Schedules

Hourly profiles of such quantities as lighting power, occupancy, and thermostat setpoint are known as "schedules" in DOE-2. There are three basic types of schedules used in BDL for the entire program.

1) **DAY-SCHEDULE**
   Defines the day's hourly profile; therefore, a separate DAY-SCHEDULE is required for each type of day that needs to be defined.

2) **WEEK-SCHEDULE**
   Defines each type of day in the week [week-day, holidays, half-workdays, etc.].

3) **SCHEDULE**
   Defines the type of week in the year, thereby allowing for the definition of calendar periods, such as summer vacations, etc.

DAY-SCHEDULE

In its simplest form, the input for DAY-SCHEDULEs is:

```
U-NAME = DAY-SCHEDULE (all 24 hours covered) (values for each hour) ..
```

For example:

```
LTG-1 = DAY-SCHEDULE (1,24) (0,0,0,0,0,0,0,0,0.3,0.6,0.8,1,1,1,1,1,1,1,0,0,0,0,0,0) ..
```

Optionally, this can be shortened by writing

```
LTG-1 = DAY-SCHEDULE (1,8)(0) (9,11) (0.3,0.6,0.8) (12,18) (1) (19,24) (0) ..
```

which is representative of a week-day daily profile. Note that hour 1 is midnight to 1am, hour 2 is 1am to 2am, etc. For example, (12,18)(1)(19,24) (0), above, means that the lights are fully on from 11am to 6pm and fully off from 6pm to midnight.

For week-ends and holidays, let's assume that:

```
LTG-2 = DAY-SCHEDULE (1,24)(0) ..
```

WEEK-SCHEDULE

The purpose of the WEEK-SCHEDULE should now be apparent;
we have two day types — LTG-1 represents week-days, and LTG-2 represents week-ends and holidays. The form of the WEEK-SCHEDULE is:

```
U-NAME = WEEK-SCHEDULE (†) (U-NAME of DAY-SCHEDULE referenced) ..
† days of week covered
```

Using the previously defined DAY-SCHEDULEs, the example can be carried forward with:

```
NORMAL = WEEK-SCHEDULE        (MON,FRI) LTG-1
                              (SAT,HOL) LTG-2 ..
```

where (MON,FRI) includes MON,TUE,WED,THU,FRI and (SAT,HOL) includes SAT,SUN,HOL.

Optionally, this can be shortened to:

```
NORMAL = WEEK-SCHEDULE (WD) LTG-1 (WEH) LTG-2 ..
```

where (WD) stands for week-days and (WEH) for week-ends and holidays. If Saturday is considered part of the normal week, you have to write (MON,SAT) LTG-1 and (SUN,HOL) LTG-2.

SCHEDULE
To illustrate the purpose of SCHEDULE, assume we have a school that is closed in the summer and on week-ends and holidays. Therefore, we need another week type:

```
VACATION = WEEK-SCHEDULE (ALL) LTG-2 ..
```

where (ALL) stands for all days of the week, including holidays, and LTG-2 was the DAY-SCHEDULE representing lights as being "off" for 24 hours.

In its simplest form, SCHEDULE takes the form of:

```
U-NAME =SCHEDULE(THRU †)(U-NAME of WEEK-SCHEDULE referenced) ..
† calendar period covered
```

To finalize the example:

```
     LIGHTS = SCHEDULE                THRU JUN 10 NORMAL
                                      THRU SEP 5  VACATION
                                      THRU DEC 31 NORMAL ..
```

Another option, "nesting of schedules", can be very useful in lessening the chore of preparing schedules. In the above example we could have bypassed the WEEK-SCHEDULEs by "nesting" the DAY-SCHEDULEs in the SCHEDULE itself. For example:

```
 LIGHTS = SCHEDULE       THRU JUN 10        (WD) LTG-1 (WEH) LTG-2
                         THRU SEP  5        (ALL) LTG-2
                         THRU DEC 31        (WD) LTG-1 (WEH) LTG-2 ..
```

Further, if there had been no vacation period, the DAY-SCHEDULE as well as the WEEK-SCHEDULE could have been bypassed by "nesting" as follows:

```
  LIGHTS = SCHEDULE THRU DEC 31           (WD) (1,8)(0) (9,11)(0.3,0.8,0.8)
                                          (12,18)(1) (19,24)(0)
                                          (WEH) (1,24)(0) ..
```

In the BDL for SYSTEMS, there are special requirements for DAY-RESET schedules, in PLANT there are DAY-ASSIGN schedules, but they all follow the same pattern described above.

Flexibility of Input Format

Most users develop a format that suits them and after a few inputs have been prepared they will use their editor to patch sections of old inputs into new ones, and thus reduce time of preparation. For example, starting on p.1.9, the entire sample input down to the first SPACE is probably worth saving as representative of office buildings. And as you will see in the SYSTEMS section of this manual, we have prepared alternative system inputs, any one of which could be merged into the file to replace the one in this sample.

The following input lines are identical to the PLANT and ECONOMICS input starting on p.1.12. The purpose of the example is to display the freestyle formatting available to you and to display the use of abbreviations and lower case lettering. This also shows how confusing a jumble of input styles can be and the importance of annotating your inputs as you go along.

If you organize the input so that the most important items appear in the left column and indent the less pertinent information, your original intent will be more understandable if you have to review the input in the future. To prove the point, compare the input in the sample run to the input below, which was prepared with no recognizable format.

```
        INPUT PLANT ..

SHW = P-E          TYPE = DHW-HEATER  size =-999 ..
HWG = PLANT-EQUIPMENT  TYPE = HW-BOILER
SIZE =-999 ..
CHR = PLANT-EQUIPMENT  TYPE = HERM-REC-CHLR  SIZE =-999 ..
P-P  BOILER-FUEL = NATURAL-GAS   HERM-REC-COND-TYPE = AIR ..
PLANT-REPORT  S = (BEPU) ..  END ..  COMPUTE PLANT ..
   INPUT ECONOMICS ..

          ELECT-RATE = UTILITY-RATE
         R = ELECTRICITY  B-C = BLC ..
         BLC = B-C   BLOCK-TYPE = ENERGY
                BLOCK-DATA = (800,0.75,1200,.090,1,.10) ..

         GAS-RATE = UTILITY-RATE   R = NATURAL-GAS
         ENERGY-CHGS = (.62) ..
     ECONOMICS-REPORT   S = (ES-D) ..
END ..
COMPUTE ECONOMICS ..
STOP ..
```