AD-753 895

CONSIDERATIONS FOR THE DEVELOPMENT OF A COMPUTER-AIDED ELECTRICAL DESIGN SYSTEM

David A. Luther

Utah University

Prepared for:

Rome Air Development Center
Advanced Research Projects Agency

September 1969

DISTRIBUTED BY:

NTIS
National Technical Information Service
U. S. DEPARTMENT OF COMMERCE
5285 Port Royal Road, Springfield Va. 22151

Best
Available
Copy

Unclassified
Security Classification

DOCUMENT CONTROL DATA - R & D

(Security classification of title, body of abstract and indexing annotation must be entered when the overall report is classified)

1. ORIGINATING ACTIVITY (Corporate author) Information Research Laboratory University of Utah Salt Lake City, Utah 84112	2a. REPORT SECURITY CLASSIFICATION Unclassified 2b. GROUP

3. REPORT TITLE

CONSIDERATIONS FOR THE DEVELOPMENT OF A COMPUTER-AIDED ELECTRICAL DESIGN SYSTEM

4. DESCRIPTIVE NOTES (Type of report and inclusive dates)

technical report

5. AUTHOR(S) (First name, middle initial, last name)

David A. Luther

6. REPORT DATE	7a. TOTAL NO. OF PAGES	7b. NO. OF REFS
September 1969	78	8

8a. CONTRACT OR GRANT NO. AF30(602)-4277	9a. ORIGINATOR'S REPORT NUMBER(S) TR 4-18
b. PROJECT NO. ARPA Order No. 829 c. Program Code Number: 5D30 d.	9b. OTHER REPORT NO(S) (Any other numbers that may be assigned this report) RADC-TR-69-364

10. DISTRIBUTION STATEMENT

This document has been approved for public release and sale; its distribution is unlimited.

11. SUPPLEMENTARY NOTES	12. SPONSORING MILITARY ACTIVITY
Monitored by Rome Air Development Center (EMIIO), Griffiss Air Force Base, New York 13440	Advanced Research Projects Agency, Washington, D. C. 20301

13. ABSTRACT

The specifications of a computer-aided design system are proposed based on the implementation of a prototype subsystem. The design system would aid a designer who wished to select and specify electrical fixtures, controls and circuits, and place them in a building whose description is stored in a computer. The proposed Electrical Design System (EDS) is meant to be a subsystem of the Architectural Design System (ADS).

The proposed Electrical Design System is partitioned into four functional areas or modes of operation. They are the lighting mode, the power mode, the service mode, and the circuiting mode. In the first three modes a user of the design system selects a predefined electrical fixture, or specifies a new fixture, appropriate to the mode chosen. He can then add a graphic symbol representing the fixture he selected to a building plan displayed on a computer-graphics console. The fourth mode, circuiting, would provide computer generated circuits containing the fixtures added to the building in the other modes.

A subsystem prototype Electrical Design System was implemented and experiences related to the implementation are explained. Also, since this prototype was integrated as a subsystem of the Architectural Design System, substantial time and effort was expended on the interface to the two. Other topics covered are man-machine interaction techniques, techniques for storing the positional data and attributes of electrical fixtures, and the use of special purpose languages.

DD FORM 1 NOV 65 1473

Unclassified
Security Classification

14 KEY WORDS	LINK A		LINK B		LINK C	
	ROLE	WT	ROLE	WT	ROLE	WT
computer graphics electrical computer-aided design electrical specification						

CONSIDERATIONS FOR THE DEVELOPMENT OF A
COMPUTER-AIDED ELECTRICAL DESIGN SYSTEM

by

David Alan Luther

This research was supported by the
Advanced Research Projects Agency
of the Department of Defense and
was monitored by David A. Luther,
RADC, GAFB, N.Y. 13440 under
Contract AF30(602)-4277.

FOREWARD

This interim report describes research accomplished by Computer Science of the University of Utah, Salt Lake City, Utah, for the Advanced Research Projects Agency, administered by Rome Air Development Center, Griffiss Air Force Base, New York under Contract AF30(602)-4277. Secondary report number is TR 4-18. Mr. David A. Luther (EMIIO) is the RADC Project Engineer.

This technical report has been reviewed and is approved.

David A. Luther

Approved: DAVID A. LUTHER
Project Engineer

ACKNOWLEDGEMENTS

I would like to thank Dr. Robert Wehrli for his assistance in helping develop many of the techniques and concepts described herein. I would like to express gratitude to Dr. David C. Evans in his role as encouraging advisor, educator and critic.

I would like to particularly thank Dr. C. Stephen Carr and Robert Geertsen for their many hours of patient assistance, both in and out of the Graphics Laboratory.

TABLE OF CONTENTS

I. PREFACE

This document is submitted as a description of a thesis project performed in partial fulfillment for the degree of Master of Science. The major field of study was computer science and the minor field was architecture.

It is the intent of the author that the following thesis, in part, serve to help those who wish to design programs which would interface to the Architectural Design System.

II. INTRODUCTION

Two and one-half years ago a group was formed at the University of Utah for the purpose of designing and developing a three-dimensional computer-aided design system. This group's particular attention was directed toward providing a computer-aided architectural design system although the underlying strategy of the complete design system was to provide the foundation for a general purpose design system. The foundations included geometric modeling techniques, data structuring techniques, and basic graphics routines. A user interface would be added to this core to accommodate a special class of users. The goal, then, of the aforementioned group was to design the core design system and, as well, to design an architectural interface to this system.

As a result of analyzing the architectural design process, certain decisions were made concerning the breakdown of such a process relative to computer implementation. So called "object systems" and "attribute systems" were defined. Examples of object systems are the enclosure system, dealing with the process of defining interior spaces, and the structural system, dealing with the structural components needed to support a building. The attribute systems deal with the attributes of the components used in the object systems. Examples of these are the cost, material and color systems.

Many of the object systems relate to components that, for the most part, are added to a building after such things as the enclosure and structural components have been described. The electrical system is assumed to, primarily, be such a system. Thus, the proposed electrical system would provide for the addition of lighting, power and service fixtures and electrical wiring to a very basic spatial description of a building.

From a computer programming standpoint, the electrical system has to communicate with the core design system and the architectural design system through which a basic building design has been constructed and stored. Inherent at this programming interface are problems of what data should be passed between the electrical and architectural systems, what routings should be provided by which system for display and data storage, and even more basic questions of program and data memory requirements and compatibility. Basic decisions were also necessary as to how and where electrical fixture data was to be arranged on the tree structure model of a building.

A special purpose language was used to simplify the job of formatting textual information for display, handling user interactions, and addressing data in the storage area.

III. OBJECTIVES AND GOALS

What is proposed is an electrical design system. Its objective is to provide a computer-aided system to a designer who wishes to completely describe the characteristics of the electrical system of a building. It is intended that the electrical design system be a well integrated part of a larger and broader architectural design system. Other design systems dealing with such aspects of architectural design as heating, plumbing, structures, esthetics, and so on, would make up a complete architectural design system.

At best, a stand along electrical design system would probably speed up the process of placing, moving, and deleting electrical fixtures on a building plan. In addition an automatic circuiting procedure could be very helpful. However, the motivation to build an electrical design system comes more from its usefulness when used in a complete architectural design system, as mentioned above. In this context the more crucial problems of the relationships between other elements of the building design can be better understood and, hopefully, eliminated. Indeed, components could be arranged more easily to affect, what might be called, "constructive interference". The electrical wiring could be integrated into the structure, for example.

Very simply, then, the goal of an electrical design system, in the context of an architectural design system, is to furnish a system to a designer that will enable him to completely specify the electrical system of a building. This includes the specification of a set of attributes to desotibe each component. Procedures for arranging components in the computer storage of the building would be available. Automatic circuiting and lighting simualtion features would also be desirable.

IV. DESIGN SYSTEM SPECIFICATIONS

The electrical design system is, most basically, intended to be a computer-graphics system. The primary output will be pictures displayed on a display console. The information that causes the console to produce a picture will be generated in a computer based on inputs supplied by the user. The user can communicate with the computer program by means of interaction devices located at the display console. These devices allow him to issue commands to change the program sequences, enter data, and manipulate picture parts displayed on the screen. For now the kinds of devices needed and the way in which they are used will not be expanded. Rather, these items will be discussed as they occur in the development of the concepts of the system. The concepts mentioned in this section describe a proposed electrical design system and are not totally represented by a computer implementation.

The electrical system has been divided into three parts according to three classes of fixtures. They are lighting, power, and service. The lighting fixtures supply electrically generated light. Examples of these are the surface incandescent lamp and the recessed florescent lamp. The power fixtures are used to dispense electrical power. An explosion proof outlet and a two-wire convenience outlet are examples of power fixtures. The third class of fixtures, service, are used to distribute electrical power throughout a building. The control panel and the generator are power fixtures.

Traditionally, whenever an electrical fixture is to be entered into the design of a building, a symbolic reference to that fixture is indicated on the plan view drawing of the building. A fixture is traditionally represented by a small graphic symbol. These symbols are, for the most part, either a circle, a rectangle, a triangle, or a combination of two of the above. The symbols are frequently coded by adding alpha-numerics to the symbol. On a specification sheet the special coding is explained and, as well, the fixtures that are actually represented by the symbols are specified. In addition, notes and other cryptic information are associated with a symbol to indicate its physical location in the z-direction and for other special circumstances.

In the electrical design system the use of graphic symbols to represent electrical fixtures will also be used. A unique symbol will be used to represent a class of fixtures, such as the class of single pole switches. A further symbology will be used to identify a class. It is a 4-character name, such as SNPL for single pole switches. The name will be used by the user for reference to a class and will also be used internally for storage purposes.

An additional 2 characters will be added to the 4-character class identifier to reference particular fixtures within a class, such as SNPL15.

For display purposes all fixtures of a class will be represented by a common graphic symbol. However, each fixture will be referenced by a unique name and will be described by a unique set of specifications. The specifications will be the attributes of the fixture. Each type of fixture will have associated to it sufficient and necessary attributes to specify its type. These will vary from class to class and even within a class. The following is an example of the attributes used to describe SNPL15:

ATTRIBUTES FOR

SNPL15

VOLTAGE	:	120 VOLTS
CURRENT	:	30 AMPS
MANUFACTURER	:	GENERAL ELECTRIC
TYPE	:	FLUSH
MODEL	:	#GE7031
MOUNTING HEIGHT	:	36
COLOR	:	RED
COST	:	$2.35

The attributes are self-explanatory with the possible exception of two, mounting height and cost. Mounting height is sometimes referred to as a "use constraint" or, in other words, a constraint affecting the use of the item in point. In this case the mounting height of the fixture must be specified since it is not apparent when the symbol appears on a 2-dimensional plan view. Here it is 36 inches above the finished floor. The attribute "cost" is referring to the unit cost.

The set of eight attribute types shown would probably be sufficient to describe most of the fixtures that would be used by a designer. However, there do exist a number of fixtures or components which require a special set of attributes, such as a generator or control panel. Those attributes that are common to most designs can be provided for by pre-setting a format into which the values of attributes can be read. However, there are several other problems in the area of specifications.

Some way must be provided to increase the number of fixture types available to a user of the system. That includes creating a new name and a new set of specifications. These operations should preferably be performed by the user. It is assumed here that the new fixture is being added to an already existing class, therefore, a suitable set of attribute types already exist. It is then just a matter of selecting a name and inputting values for the attributes.

A different sort of thing exists when a user finds it necessary to have a new <u>class</u> of fixtures. Again it appears that the <u>user</u> should have the ability to create one. In this case, however, significantly more information is needed. A symbol is needed to represent the

the class. A name is needed for each picture type included in the class and a set of attribute types is needed to specify all fixtures in the class. The last requirement refers to not just the values of the attributes but also the kinds of attributes necessary.

At this point a description of the general operations of the design system will tie together many of the topics mentioned above.

When the electrical design system program is entered, the console will display the mode name lighting, power and service. At this point and at several others, the user will be required to determine his course by entering the command selected from those listed on the screen. It appears that by displaying his alternatives the uneducated user will be explicity made aware of them. The educated user, who will probably be aware of them anyway, can simply overlook their presence. As well, interactive computer programs often leave the user confused as to where he is in the program sequence and some feedback or verification of one's progress appears helpful.

The question of how the user will interact with the scope is a difficult one. In consideration of the interaction devices currently available, the "mouse" appears to be the best for pointing at things displayed on the console. The mouse was developed by Engiebart (6). The fact that the mouse fits so well in the hand, that the mouse holding hand can move comfortably on any flat surface, and that the movement of the mouse seems most free and natural make it a good choice. There is a very direct movement of a tracking cross on the screen for a similar movement of the mouse and the tracking cross can be pinpointed very easily. The use of a function button on the upper surface of the mouse seems very acceptable for use as an "accept" command or switching parameter.

While on the subject of the SRI mouse and interaction devices, it seems that the keyboard and five-finger keyboard used at SRI and illustrated in (6) would also be useful to the electrical design system. The five-finger keyboard would be useful for inputting commands of 2 or 3 characters in length. But its use to input characters in any large quantity is not warranted and a regular typewriter-like keyboard would be necessary. The important point concerning these devices and the mouse, as used at SRI, is their efficient arrangement directly in front of a user. Their use in a system such as the electrical design system involves going from one form of device to another. This condition might be objectionable if the user were required to change his physical position or his seating position to use one device and then another. However, when the devices are arranged as they are at SRI, the transition from one to another does not seem clumsy or confusing.

It is important to establish a consistent use of each of the three devices in order to avoid the confusing use of a multiplicity of devices to execute a single command. For this reason the electrical design system will adopt the use of the teletype to input textual data. The five-fingered keyboard will be used to input commands to the system. The commands will be two or three letters long. The teletype can be used, however, for commands by users inexperienced on the five-fingered keyboard. The mouse will be used exclusively for pointing at picture points on the screen. No command words will be displayed on the screen. The mouse will be used only incidentally for command sequencing.

With this background, assume, again, that a user is presented with the initial display in the electrical design system. The message on the screen asks him to choose one of the three modes by inputting its code name--PO for POWER, LI for LIGHTING, or SV for SERVICE.

The selection of either of the three modes will cause a list of graphic symbols to be displayed. The list represents, in each of the modes, the classes of fixtures currently available in that particular mode. Each class will also have a decimal number and a textual name displayed with it on the screen. If the user wishes to use one of the fixture types provided on the screen, he selects that type by inputting its decimal number. If the user wishes to use a type that is not at that point provided, he may define a new class by inputting the code NC for "new class". Leaving this for the moment, assume the user selects a class already provided.

At this point the user is provided with a displayed message to the effect that N number of fixtures are available in that class, from, say SNPL15 to SNPL45. The user now has several options. He can specify his choice of fixture by inputting the decimal number representing its position on the list of those available, say "2" for SNPL20. This will cause the values of the attributes of SNPL20 to appear, as illustrated on page 8. If the user is not satisfied with this choice, the code word "UP" allows him to view the next fixture in the list, SNPL25, and "DN" will allow him to go back down the list to SNPL15.

Of course, if the user is not satisfied with any of the available choices, he can create a new fixture. The code "NF" will generate a new fixture name, SNPL50 in the example, and display the attribute types without values. The user can then type in data, proceeding from the top attribute down until he has reached the last attribute or signals that he is done by a series of three carriage returns. Up to 60 alpha-numeric characters can be entered into one attribute category.

A user can edit the information he has entered for a new fixture by using the code ED followed by the line number of the attribute he wishes to edit. The line in error must be completely retyped. Of course, three carriage returns will advance the user from this section of the system or if he wishes to input more attributes, RT followed by an attribute line number will return him to the line specified. From this line, the sequence will proceed down the list as described above.

Once the user has selected a fixture, either by scanning those already established and choosing one or by specifying a new one, he can begin placing copies of the fixture chosen on the building plan. The sequence of 3 carriage returns will display the building plan. A fixture is placed by moving the tracking cross with the mouse to the desired position on the plan and pushing the execute button on the mouse. A question now arises as to just where the actual fixture is to be placed in the model of the building.

The architectural design system uses a hierarchial structure to model a building. The building is the top node, followed by room nodes, each of which has wall nodes, and this breakdown continues to some arbitrary level. The question of where electrical fixtures should be placed, mentioned in the last paragraph, is actually the question of where they should be associated in the overall tree structure model of the building. It appears that the choice is really between the room level and the wall level. A comprise solution is probably necessary in which fixtures will normally be associated to walls unless a serious ambiguity exists or such an association makes no sense. The ambiguity refers to whether a fixture is on one side of a wall or the other. It appears that the best approach to the placement of fixtures is as follows. When a user wants to place a fixture on a wall that is common to two rooms, he should place the tracking cross close to the wall but within one of the rooms, or, in other words, not directly on the wall. The computer program will then determine the proximity of the cross to a displayed wall. If it finds a wall within some pre-set tolerance or proximity, a symbol will replace the cross and a line will be drawn from the symbol, normal to the wall. The line is a verification to the user. If he does not feel that the wall selected by program represents his choice, he can use the move or delete command to alter the position of the symbol. These commands will be outlined shortly.

If the program did not find a displayed wall within a sufficient proximity to the tracking cross, the tracking cross will nonetheless be replaced by a symbol. However, in this instance no line will be drawn from the symbol and in the model of the building that fixture will not be associated with a wall, but will be associated with the room that encompasses the position of the symbol. This procedure will allow for the placement of such fixtures as outlets or motors that are not normally associated with walls. Of course, this default procedure will also be invoked when a user means to place a fixture on a wall but fails to place the tracking cross within sufficient proximity to the wall. In this case, again, the move or delete commands can be used to re-position the fixture.

Once these few fundamentals are understood, the user can place fixtures rapidly. The tracking cross-mouse combination can be though of as analogous to the rubber stamp-ink pad, insomuch as the mouse is "inked up" with copies of the current symbol and a symbol will be displayed as many times as the mouse-execute button is pushed. When the user wants to change symbols, "CS" will return him to the list of symbols of the current mode or "CM" will get him back to where he can select another node.

Two editing functions are available for changing the positions or the number of symbols already placed.

The "move" feature enables the user to re-position a symbol. The sequence is "MS" for "move symbol", followed by moving the tracking cross over the symbol to be moved and pushing the execute button, and finally, moving the tracking cross to a new position for the symbol and pushing the execute button. If for the middle step-- identifying the symbol to be moved - the program does not find a symbol, a message will appear on the screen requesting the user to try again. To avoid a loop condition, the command "DC" will delete the move command entirely.

The delete feature allows a user to remove a symbol from the screen. The correct sequence is the code word "DS" for "delete symbol" followed by an identification of the symbol to be deleted using the mouse and execute button. Again, if the program fails to find a symbol matching the coordinates of the mouse, a message will ask the user to try the identification part again. The "delete command" code word can also be used here as described above.

An additional feature may be incorporated into the delete sequence to aid in the identification of the actual fixtures represented by the symbols on the screen. It must be remembered that a symbol represents a whole class of fixtures and that several members of a class may be represented by a single symbol on any given plan. The code word "RN" will retrieve the names of the fixtures currently represented on the screen and display the name below the appropriate symbols. (The addition of the names to the display will probably cause degradation of the picture through flicker or other means and should be used sparingly to preserve good picture quality.) The code name "RS" followed by the identification of a symbol using the mouse will retrieve and display the attributes of the symbol selected this procedure can be only used for viewing the attributes, not for making changes to them.

For viewing convenience another feature of the electrical design system is scaling. The scales of 1/4, 1/8, 1/16 are available and can be selected by the code words "SF", "SE", or "SS". These scales make one inch on the screen equal to four feet, eight feet, and sixteen feet respectively.

To make scaling more meaningful, a light, straight line grid is displayed concurrently with the building plan and symbols. The actual spacing between grid lines in scope units remains constant with changes in scale. Therefore, the value of the grid separation must change. Depending upon the number of raster units per inch on any particular scope, the grid spacing would then be arrayed to produce an actual unscaled value of 16 inches, 32 inches and 64 inches, respectively, at the three scales. Using this criteria, the grid not only provides a reference but also a meaningful guide for designers who are conscious of the traditional multiple of 16 inches building modulus.

Additional Features

The usefulness of the electrical design system could be extended by several additional features.

At present the various lighting manufacturers publish two dimensional graphs which show light intensity patterns for most lamps. The light intensity contours are usually regular curves and probably could be easily duplicated on a computer and displayed on a graphics console. A light simulation program could reproduce the sum of multiple sources of light with infringing light patterns. The psychological effects of lighting and the heating effects of lighting could be integrated into a complete design solution for a building.

It is clear that the electrical design system can provide some kind of aid in determining electrical circuits as well as in establishing wire and conduit runs. The requirement here is to be able to express all the factors (or at least the most critical) in the form of a mathematical relationship or procedure which can be expressed in a computer program. There are, of course, many factors that are expressed in mathematical terms in the electrical code provisions. These provisions deal with such things as the number of fixtures of a particular type in a circuit, the number and kinds of branch circuits, circuit-power relationships, and many others. Electrical properties such as voltage, current, and power relationships and voltage drop could be expressed. Other less precise relationships would require some experimentation to determine what approach is best. An example is the question of how the elements of a circuit should relate according to their physical location in a building. Should an attempt be made to minimize the sum of the distances between a group of similar fixtures? In two dimensions or three? Generally, it appears that automatic circuiting and wire routing could best be accomplished by a man-machine interaction. This could amount to either merely a human override of a computer solution or a more elaborate piecemeal scheme of distributing the tasks between user and computer. This might involve the use of a tracking procedure for the mouse which would then allow the user to trace out a circuit or conduit run.

V. SUBSYSTEM PROTOTYPE

The proposed electrical design system outlined in the last section developed partly from the experience of implementing a prototype electrical design subsystem on the Univac 1108 computer and associated PDP-8 computer and Information Displays Incorporated display console. All equipments are located at the University of Utah. Several aspects of this prototype will be discussed in this section with respect to the computer system involved rather than from the user's viewpoint.

The subsystem was composed of a large main computer program and approximately thirty subroutines. The main program was responsible for responding to user interactions. It provided a skeleton for control and called on the subroutines to perform specific, isolated functions. This approach to program structure made the main program easy to compose and, especially, easier to debug because the flow of control through the program was relatively easy to follow. As well, each subroutine could be independently written and debugged. Approximately two-thirds of the subroutines were written in Fortran V while the remaining subroutines and the main program were written in a language called Graphics Fortran. Graphics Fortran was developed as a system programming language for the architectural design system.

Relationship to the Architectural Design System

The prototype electrical design system was intended for use by a designer after he had established at least a two-dimensional description of a building plan. The designer was required to explicitly label those geometric parts of the plan which he meant to be walls. He was then free to call the electrical design system.

When the label of wall was attached to parts of the plan, an indication of this fact was made in the tree structure modeling scheme of the building. The modeling scheme was stored by a hashing scheme which operated on associative triples similar to those of Lincoln Lab's "Leap" (7). In the case of the wall designation the associative triple was: MAKE 'TYPE' OF 'AAA4' = 'WALL', which was hashed according to the values of the literal TYPE, the literal name of the geometric part AAA4, and the literal WALL.

When the electrical design system was entered, it was necessary to retrieve the information concerning walls since in the prototype fixtures were associated to walls as sub-nodes. Thus, the tact was taken that the electrical design system would provide the means for retrieving that particular data from the model of the building that it needed as a subsystem. Therefore, an electrical system routine searched the data base for all occurrences of the type wall. For all occurrences, an entry was set up in a local array which held the name of the wall element and its centerline coordinate in the building coordinate system.

It is important in cases of this kind to determine just what part of the data concerning the building design is needed by any particular subsystem. Provisions for identifying this data can be made in the general design phase, but it is the responsibility of the subsystem to provide the facility for retrieving that data and only that data it specifically needs. This provides for a less confusing separation of tasks and an economy of computing.

As well as the background data needed by the electrical subsystem prototype, certain information was needed from the architectural design system for display purposes. Since the model of the building may extend down to some arbitrary level of detail, it is clear that for any given application a parameter must be set to the lowest level of detail that it is necessary to display for that application. For the electrical system prototype, the wall level was the lowest level displayed. The architectural design system actually took account of the level display parameter and created the display file for the building. The display file was communicated to the electrical prototype by putting it in a storage array which was common to both the electrical and architectural systems. As well, a parameter which held the value of an index to the end of the display file was passed in common storage.

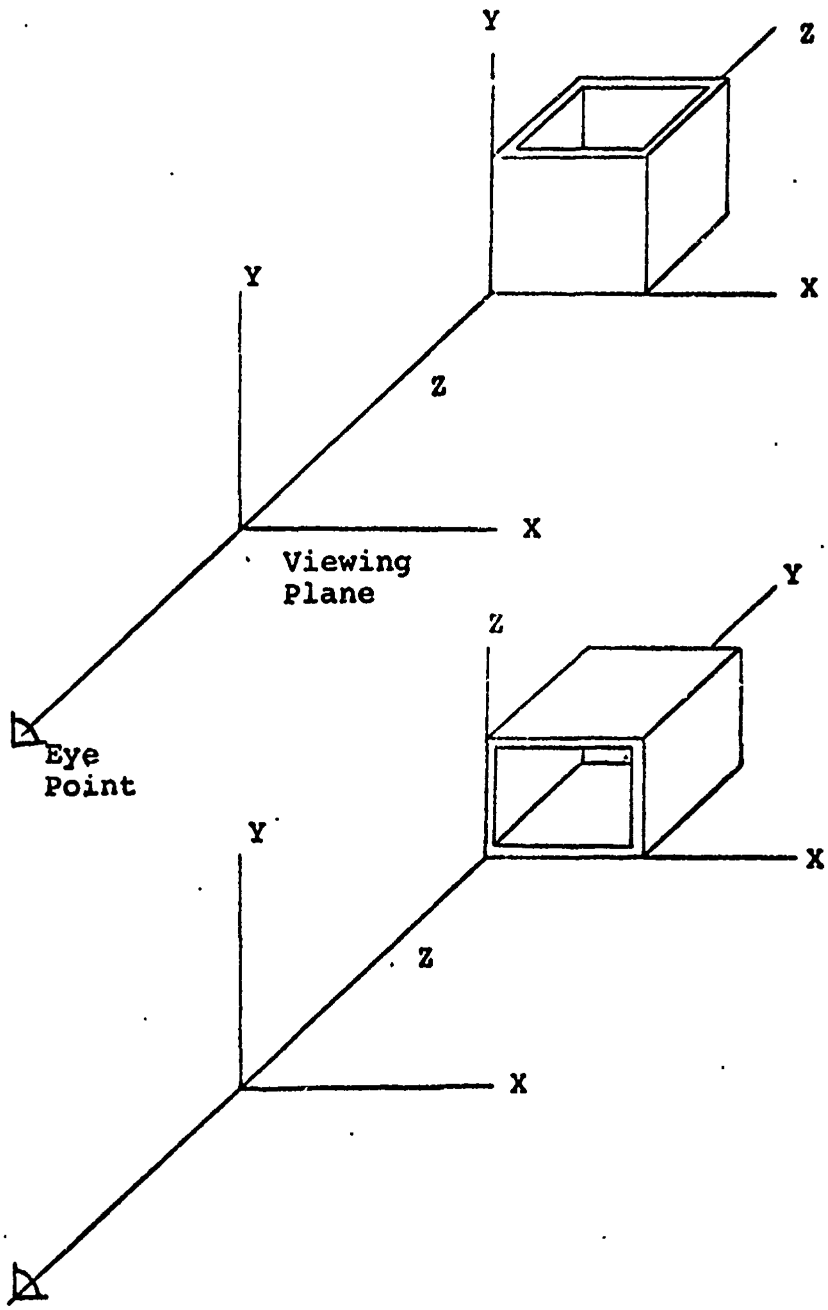
The command to the architectural design system to create the building display file is sent from the electrical system. In addition to the level display parameter, the architectural system display routine is made aware of several other display parameters.

There are nine display parameters used by the architecture modeling scheme as shown by Carr in (1). They are translation in X, Y, and Z, rotation in X, Y, and Z, and scaling in X, Y, and Z. Two of these parameters are set in the electrical subsystem to cause the necessary plan view of the building to appear at a particular scale. The plan view was actually produced by rotating a building without a roof around its own X-axis. This was done by setting the parameter VRX to -90 degrees.

A thorough discussion of the way coordinates are represented is necessary before the electrical scaling requirements can be explained.

The architectural design system records dimensions relative to each node in the tree structure model of a building. That means that, for example, each wall is dimensioned relative to its own coordinate system. If the wall is made a part of a room, then the wall coordinate system is related to the room coordinate system by

This operation is shown in the following diagram.



the nine parameters -- translation, rotation, and scaling in X, Y, and Z. In the same way the room is related to the building coordinate system. The values of building coordinates are determined by the following formula: B scope units = A feet x K where K = 12 $\frac{\text{inches}}{\text{foot}}$ x 100 $\frac{\text{raster points}}{\text{inch}}$ = 1200 $\frac{\text{scope units}}{\text{foot}}$. The value of "A" above is expressed in feet in the building coordinate system. Then, considering only two dimensions and X_R and Y_R stored with values determined by the above, the following equations give screen coordinate values (all values in the positive X and Y quandrant):

$$X_S = \left[\left[(X_R - 256)/\left(\frac{VTZ \times 1200}{2000} + 1\right) \right] + 256 \quad 2 \right]$$
$$Y_S = \left[\left[(Z_R - 256)/\left(\frac{VTZ \times 1200}{2000} + 1\right) \right] + 256 \quad 2 \right]$$

Basically the equation is simply a building coordinate system value multiplied by a scale factor to give a screen coordinate value. The use of "256" and the factor "2" are necessary to provide bias for a hidden line and perspective algorithm and are dealt with in (1).

The scale factor is related to the distance of the viewer from the viewing plane (EYPDIS) and the parameter VTZ which indicates a translation along the Z-axis of the building coordinate system. The formula is

$S.F. = \frac{VTZ \times 1200}{EYPDIS}$. For purposes of this discussion the eye-point distance is constant at 2000 scope units. In the general architecture design system the user may zoom in and out arbitrarily. However, in the electrical design prototype, it was adequate to cause objects to appear at the traditional scales of 1/4, 1/8, and 1/16. At these scales one inch on the screen equals four, eight, and sixteen feet, respectively, in the building coordinate system. To produce these scales, VTZ must be 80, 160, and 320 feet. The value of VTZ is the second parameter supplied to the architecture display routine for the initial display (160 or 1/8 scale) and for every scale change. At the scales of 1/4, 1/8, and 1/16, the screen can accommodate objects of 40, 80, and 160 feet.

There are two types of displays presented by the electrical design system. One type involves the display of the building plan and electrical system symbols. The other type involves no reference at all to the building plan but is one of several local displays. To conserve storage area when the electrical and architectural prototypes were in computer core memory, a common display file storage area was used. This resulted in the following strategy concerning display file creation and execution. When the electrical prototype had to create a local display, such as the display of symbols available, the common storage area for the display file was used in the normal fashion. In this case the display file would be declared, initiated, created, and executed entirely within electrical. However, when a display was to be composed of the building plan, in part, a different approach was taken. The display file was declared in the architecture prototype, initialized, and filled with the commands necessary to display the building plan. The display file index is also in common, thereby enabling the electrical system to record the amount of display file used for the building plan. This much of the display file is executed when electrical is first entered to display only the plan. After this, the electrical symbols are added to the display file beyond that storage used to display the plan. As symbols are added, moved, or deleted, only that part of the display file referring to them need be re-created. Using this strategy, the display could be changed very rapidly and without any annoying flashing or flickering. A very annoying flash is apparent if, instead, the whole display is re-created. In this case, of course, the not-so-small job of re-creating the building plan from the tree structure model is necessary.

Strictly for economy of storage, it was necessary to consolidate common storage areas within the electrical prototype and place them in storage already reserved by the architectural prototype. It turned out that much of the storage reserved by architectural was available to electrical because the function it was provided for was not needed. The most noteworthy case was the hidden line-half tone function. Not using this function provided a large amount of otherwise unused space to the electrical system.

Data Structuring and Storage

The electrical prototype used both local, temporary storage areas and also the more permanent type provided by the architecture system. The information concerning the attributes of the fixtures was stored as a block in the permanent hashed storage space. Information describing the position, class, and type of fixtures was stored in local storage and, as well, was attached to the tree structure model of the building to which they were added.

The attributes of a fixture required 48 words of storage to hold the information related to the eight attribute categories. The storage arrangement is depicted in the following diagram. The single letters are single storage words and the others are arrays as indicated.

ATTRIBUTES - STORAGE

VOLTAGE	V
CURRENT	U
MANUFACTURER	M(10)
TYPE	T(10)
MODEL	D(10)
MOUNTING HEIGHT	H
COLOR	C(10)
COST	A

50-WORD BLOCK

to DISPLAY

For display purposes, the eight storage areas are separately passed to display routines by the use of the special purpose language, Graphics Fortran. They are also arranged in 48 consecutive array locations and are stored in the hash storage space as a 50-word block. The block is stored according to a hash address which is specified by the following expression. The expression is similar to one from the Leap Language (7) and is processed by Graphics Fortran. The expression used is "MAKE 'ATTRIB'* SYMNAM = [50] BLOCK". The hash address generator uses the literal ATTRIB, the value of the fixture or symbol name, and the value of the first word of BLOCK determine an address.

When a symbol is added to the building plan, a new five item entry is made to an array in the electrical system. The entry is depicted in the following illustration:

SYMBOL TABLE 'SYMBOL'

NEW X
NEW Y
SYMBOL TYPE
SYMBOL NAME
BRANCH NAME

NEWX and NEWY are the coordinates of the fixture in the building coordinate system. The next entry SYMBOL TYPE describes the class of fixture used. This is used to determine which symbol should be displayed. The SYMBOL NAME refers to the class and, more specifically, the actual fixture within the class used.

As well as this entry per fixture, another is made into the hashed storage area. When the program determined which wall of the building the fixture was to be attached to, it stored the name of that wall in WALNAM. This value, together with the value of the name of the fixture SYMNAM and the literal SYMBOL, are combined in the association: MAKE BN [BRN]'SYMBOL'*WALNAM = SYMNAM. The effect of this statement is to attach a new fixture given by SYMNAM to the wall node WALNAM in the building tree structure. The number identifying this branch on the wall node is returned into BRN. The value of BRN is stored, right justified, into the variable BRANAM. The literal 'SYSP' is already there left justified. The composite might then be 'SYSP03'. The coordinates NEWX and NEWY are transferred into the two position array POST. The association "MAKE BRANAM *WALNAM = [2] POST is then used to store the coordinates away in the hashed storage space. The use of the branch number provides for uniquely identifying multiple instances of the same fixture on one wall. As shown in a previous diagram, BRANAM is also stored as the fifth entity in a symbol table entry.

Here it is available should a symbol be moved or deleted. When the program determines which symbol is meant to be moved or deleted, which wall it is attached to (the same routine is used to determine the wall when the symbol is first placed), this, together with identifying the branch number from the symbol table is used to make the appropriate correction in the hash area. For example, the association concerning the fixture coordinates can be broken using the expression - "BREAK SYMBOL(I,5) *WALNAM = ?". This means "destroy the association with the branch name contained in SYMBOL(I,5) and wall name contained in WALNAM, regardless of the value of the triple, represented by the ?".

To delete a fixture, the BRANAM must be referred to again. The procedure for removing the coordinates is exactly the same as described for MOVE above. However, the reference to the fixture attached at the wall node must also be removed. Here only the part of BRANAM containing the branch number is used. The 'SYSP' parts is discarded. When the branch number has been stored into BRN, this expression is used: "BREAK BN [BRN] 'SYMBOL' * WALNAM = ?". Again, there may be multiple instances of this fixture on this wall but the only reference destroyed will be the one whose branch number is given by BRN.

It might be well to reflect briefly on the procedure used to determine which wall a fixture should be attached to. The actual scope coordinates of the point where the user is indicating he wishes to place a fixture are converted to building system coordinates. They are then used by a function which determines their proximity to the walls of the building plan. The name and centerline end point coordinates of each wall are already available in a table described earlier. The proximity function works as follows: an imaginary ellipse is created around each wall, in turn, where the major axis of the ellipse is coincident with the center line of the wall and the minor axis is some fraction of the major. The proximity function returns a true value when the point being tested lies within the area of the ellipse. The ellipse was used rather than a rectangle, for example, to reduce the ambiguity where two walls interact at one end of each. The rectangles would overlap each other to some extent where the walls intersect and thus provide an ambiguous situation with respect to which wall the user intended. On the other hand, two ellipses have very little in common at the point where the walls intersect.

The Use of Special Purpose Languages

The electrical design system prototype relied almost entirely upon the graphics software interface described in references (2) and (3). These sources describe a set of Fortran callable subroutines and functions useful for producing graphics on the Utah graphics console. Also outlined are procedures for handling interactions from the teletype and Sylvania tablet and procedures for integrating variables from the teletype.

In addition to these a specific special purpose language was used. The language is entitled, "Graphics Fortran." It has several applications in relation to the electrical system.

The first use of Graphics Fortran occurred within the process of displaying alpha-numeric S after they were typed in through the teletype. This data represented the values of the fixture attributes. The names of the attributes were already displayed on the screen and the values were then added as they were typed. Certain language constructs were provided to pass the data to the display screen in a pre-set format. The data could be of integer, real, or alphabetic type. Display format is easily arranged and changed.

Graphics Fortran also processed the Leap like associated data statements used in the electrical design system. With these associative triple statements, the electrical system structured and stored information into a hashed storage area. The amount of storage used depended upon the ways in which the associative triple was to be retrieved. The user could declare which or all of the combinations of the attribute, object, and value were used to arrive at hash addresses. Another useful feature enabled the storage and retrieval of arrays of storage.

It is important to note that Graphics Fortran was very evolutionary in that changes could readily be made to its compiler to reflect user demands. The compiler was one automatically generated using the Tree Meta complier building system (8).

BIBLIOGRAPHY

1. Carr, C. Stephen. Geometric Modeling. Technical Report 4-13. Salt Lake City, Utah: University of Utah, 1969.

2. Copeland, L. & Carr, C. S. Graphics System. Technical Report 4-1. Salt Lake City, Utah: University of Utah, 1967.

3. Reed, A. C., et al. A Fortran V Interactive Graphical System. Technical Report 4-4. Salt Lake City, Utah: University of Utah, 1968.

4. Smith, Max J., et al. Space-Form. Technical Report 1-2. Salt Lake City, Utah: University of Utah, 1969.

5. Wehrli, Robert. Open-Ended Problem Solving in Design. PhD Thesis University of Utah, 1968.

6. Englebart, D. C., English, W. K. A Research Center For Augmenting Human Intellect. AFIPS Conference Proceedings, FJCC, 1968, p. 395.

7. Rovner, P., Feldman, J. The Leap Language and Data Structure. Technical Report. Lexington, Massachusetts: Lincoln Laboratory. October, 1967.

8. Carr, C. S., Luther, D. A., Erdman, S. The Tree-Meta Compiler-Compiler System. RADC-TR-69-83. Salt Lake City, Utah: University of Utah. March 1969.

```
      EDS2
      LIST CODE
      INCLUDE DSPRMS,LIST
      INCLUDE EDSPRM, LIST
      IMPLICIT INTEGER(A-Z)
      INTEGER POST(3)
       COMMON/EDSPOS/POST
      COMMON/EDSSYM/SYMNUM,SYMBOL(NUMSYM,5),TEMP(NUMSYM,3),NEWX,
C     SYMNAM,WALLST,WTP1,NEWY
      COMMON/EDSDAT/WALNAM,KKK,BRN,BRANAM
      COMMON/EDSSCL/K
      REAL VTX,VTY,VTZ,VRX,VRY,VRZ,VSX,VSY,VSZ
      COMMON /VP/VTX,VTY,VTZ,VRX,VRY,VRZ,VSX,VSY,VSZ
      INTEGER WTP1,WALLST(5,WTPL)
      REAL RWALLT(5,WTPL)
      EQUIVALENCE (WALLST,RWALLT)
      COMMON/EDSATB/V,U,M(10),T(10),D(10),H,C(10),A,ICHAR,ATT
      COMMON/EDSNME/TYPE(50,3),MAP(50),PICK
      COMMON/EDSLBT/LFLAG
      INTEGER DFV(DFVL),DFVP,DFVF
      INTEGER DFB(DFBL,WPN)
      COMMON /DF/DFV,DFB,DFVP,DFVF
      NAMELIST/AGAIN/CM,CD,CS,RETURN,DC,SYMNAM,BRANAM,NEWX,NEWY
      NAMELIST/VIEW/F,E,SIX,WALNAM,WALLST,RWALLT,VTX,VTZ,VTY,WTP1
      NAMELIST/BACK/R
      NAMELIST/RESET/RS
      NAMELIST/ABORT/STOP
      SEGMENT ENTRY 9
       ALLOW  0
      CALL INOUTM
      CALL MARGN(0)
      CALL ORG(0,0)
      CALL SWPCHR('E')
      CALL TABABL
      CALL TABTOL(10)
      CALL INTEN(2)
      CALL NAMELD
      CALL GZZ20
      CALL LCHAR
      CALL WALCRD
      LFLAG = 0
      CT = 0
      VTZ = 160.0
      VTY = 0.0
      VTX = 0.0
      SYMNUM = 0
      XSCALE = 0
      ATTR = 0
      MODE = 0
      YSCALE = 0
```

```
      K = 8
C*************************************************************************
C       ------ THE BEGGINNING OF THE PROGRAM ---------------
C*****  DISPLAY MODE NAMES *********************
 2      CALL SETDF(DFV,DFVP)
        CALL GRIDLN
        CALL DRGRID
        CALL WRITAT(280,900,'THE ELECTRICAL DESIGN SYSTEMΔ')
        CALL WRITAT(648,676,'LIGHTINGΔ')
        CALL WRITAT(648,644,'POWERΔ')
        CALL WRITAT(648,612,'SERVICEΔ')
        CALL WRITAT(648,580,'CIRCUITINGΔ')
        CALL SENDF
        CALL TABINT(1,$10)
        CALL CHRINT(1,$8)
        CALL SETLST
        READ(5,ABORT)
        CALL JUMPS('ABORT',$800)
 6      CALL IDLE
        CALL SWAP
        GO TO 6
 8      CALL TTY
        GO TO 6
 10     CALL GETTAB(IX,IY,IZ)
        IF(IZ) 6,6,15
C*E***  CHECK WHICH MODE WAS SELECTED BY STYLUS *****
 15     CALL LITEBT(600,668,775,700,IX,IY)
        IF(LFLAG .EQ. 1) GO TO 20
        CALL LITEBT(600,636,775,668,IX,IY)
        IF(LFLAG .EQ. 1) GO TO 21
        CALL LITEBT(600,604,775,636,IX,IY)
        IF(LFLAG .EQ. 1) GO TO 22
        CALL LITEBT(600,572,775,604,IX,IY)
        IF(LFLAG .EQ. 1) GO TO 23
        GO TO 900
C****** TRANSFER TO APPROIATE MODE *********
 20     MODE = 1
        GO TO 1505
 21     MODE = 2
        GO TO 85
 22     MODE = 3
        GO TO 1280
 23     MODE = 4
        GO TO 110
C***************************************************************************
C********* THE POWER MODE *********************
 85     ATTR = 0
C*****  DISPLAY POWER SYMBOLS  ***********************
        CALL SETDF(DFV,DFVP)
 86     CONTINUE
```

```
      PART = 2
      CALL POWER
      CALL POWSYM
      CALL LCHAR
      CALL WRITAT(50,50,'ATTRIBUTESA')
      CALL SENDF
1402  CALL TABINT(1,$1410)
      CALL CHRINT(1,$1405)
      CALL SETLST
      READ(5,AGAIN)
      CALL JUMPS('AGAIN',$2,$990,$950,$800,$1150)
      CALL SETLST
      READ(5,RESET)
      CALL JUMPS('RESET',$1447)
1400  CALL IDLE
      CALL SWAP
      GO TO 1400
1405  CALL TTY
      GO TO 1400
1410  CALL GETTAB(XX,YY,ZZ)
      IF(ZZ) 1400,1400,1420
C********  DETERMINE WHICH SYMBOL WAS SELECTED *************
1420  CHECK = 1
      DO 1430  LB = 870,646,-32
      LY = LB - 16
      UPY = LB + 16
      CALL LITEBT(625,LY,655,UPY,XX,YY)
      IF(LFLAG .EQ. 1) GO TO 1445
      CHECK = CHECK + 1
1430  CONTINUE
      DO 1440  LB = 559,175,-32
      LY = LB - 16
      UPY = LB + 15
      CALL LITEBT(620,LY,655,UPY,XX,YY)
      IF(LFLAG .EQ. 1) GO TO 1445
      CHECK = CHECK + 1
1440  CONTINUE
      GO TO 1495
1445  PICK = CHECK
      CALL ATTNAM
      CALL GETVAL
1447  CALL SETUF(UFV,UFVP)
C***********  ESTABLISH SYMBOL NAME AND ATTRIBUTES    *********
1448  CALL ATTRIB
      CALL SENDF
      CALL INPUT($1490,$1448)
1490  IF(CT .EQ. 1) GO TO 260
      GO TO 100
1495  CALL SETUF(UFV,DFVP)
      CALL WRITAT(225,30,'NO SYMBOL WAS FOUNDA')
```

```
      CALL WRITAT(225,5,'PLEASE TRY AGAINΔ')
      GO TO 86
C*******************************************************************
C********* THE SERVICE MODE ************************
 1280 ATTR = 0
C****** DISPLAY SERVICE SYMBOLS   ***************
      CALL SETDF(DFV,DFVP)
 1290 CONTINUE
      PART = 3
      CALL SERVCE
      CALL SERSYM
      CALL WRITAT(50,50,'ATTRIBUTESΔ')
      CALL SENDF
      CALL TABINT(1,$1310)
      CALL CHRINT(1,$1305)
      CALL SETLST
      READ(5,AGAIN)
      CALL JUMPS('AGAIN',$2,$990,$950,$600,$1150)
 1300 CALL IDLE
      CALL SWAP
      GO TO 1300
 1305 CALL TTY
      GO TO 1300
 1310 CALL GETTAB(XX,YY,ZZ)
      IF(ZZ) 1300,1300,1320
C****** DETERMINE WHICH SYMBOL WAS SELECTED   *************
 1320 CHECK = 22
      DO 1330  LB = 702,440,-32
      LY = LB - 16
      UPY = LB + 16
      CALL LITEST(690,LY,725,UPY,XX,YY)
      IF(LFLAG .EQ. 1) GO TO 1335
      CHECK = CHECK + 1
 1330 CONTINUE
      GO TO 1395
 1335 PICK = CHECK
C***********   ESTABLISH SYMBOL NAME AND ATTRIBUTES  ********
      CALL ATTNAM
      CALL GETVAL
 1347 CALL SETDF(DFV,DFVP)
 1348 CALL ATTRIB
      CALL SENDF
      CALL INPUT($1390,$1348)
 1390 IF(CT .EQ. 1) GO TO 260
      GO TO 100
 1395 CALL SETDF(DFV,DFVP)
      CALL WRITAT(225,30,'NO SYMBOL WAS FOUNDΔ')
      CALL WRITAT(225,5,'PLEASE TRY AGAINΔ')
      GO TO 1290
C*******************************************************************
```

```
C      ------ CIRCUITING SELECTED -------------------
 110      CALL SETDF(DFV,DFVP)
          CALL WRITAT(280,900,'THE ELECTRICAL DESIGN SYSTEMΔ')
          CALL WRITAT(648,676,'LIGHTINGΔ')
          CALL WRITAT(648,644,'POWERΔ')
          CALL WRITAT(648,612,'SERVICEΔ')
          CALL INTEN(0)
          CALL POS(500,588)
          CALL VEC(600,588)
          CALL INTEN(1)
          CALL VEC(575,594)
          CALL POS(600,588)
          CALL VEC(575,580)
          CALL INTEN(2)
          CALL WRITAT(648,580,'CIRCUITINGΔ')
          CALL SENDF
          CALL SETLST
          READ(5,AGAIN)
          CALL JUMPS('AGAIN',$2,$990,$950,$800,$1150)
          CALL CHRINT(1,$120)
 115      CALL IDLE
          CALL SWAP
          GO TO 115
 120      CALL TTY
          GO TO 115
C*****************************************************************
C**********************************************
C********* THE LIGHT MODE *********************
 1505     ATTR = 0
          PART = 1
C*******     DISPLAY LIGHTING SYMBOLS     **********
          CALL SETDF(DFV,DFVP)
 1507     CONTINUE
          CALL LIGHT
          CALL LITSYM
          CALL WRITAT(50,50,'ATTRIBUTESΔ')
          CALL LCHAR
          CALL SENDF
          CALL TABINT(1,$1510)
          CALL CHRINT(1,$1502)
          CALL SETLST
          READ(5,AGAIN)
          CALL JUMPS('AGAIN',$2,$990,$950,$800,$1150)
 1500     CALL IDLE
          CALL SWAP
          GO TO 1500
 1502     CALL TTY
          GO TO 1500
 1510     CALL GETTAB(XX,YY,ZZ)
          IF(ZZ) 1500,1500,1520
```

```
 1520     CHECK = 31
          DO 1525  LB = 764,636,-32
          LY = LB - 16
          UPY = LB + 16
          CALL LITEBT(555,LY,605,UPY,XX,YY)
          IF(LFLAG .EQ. 1) GO TO 1540
          CHECK = CHECK + 1
 1525     CONTINUE
          DO 1530  LB = 700,636,-32
          LY = LB - 16
          UPY = LB + 16
          CALL LITEBT(515,LY,545,UPY,XX,YY)
          IF(LFLAG .EQ. 1) GO TO 1540
          CHECK = CHECK + 1
 1530     CONTINUE
          CHECK = 9
          DO 1535  LB = 559,175,-32
          LY = LB - 16
          UPY = LB + 16
          CALL LITEBT(550,LY,590,UPY,XX,YY)
          IF(LFLAG .EQ. 1) GO TO 1540
          CHECK = CHECK + 1
 1535     CONTINUE
          GO TO 1595
 1540     PICK = CHECK
C***********   ESTABLISH SYMBOL NAME AND ATTRIBUTES  *********
          CALL ATTNAM
          CALL GETVAL
 1547     CALL SETDF(DFV,DFVP)
 1548     CALL ATTRIB
          CALL SENDF
          CALL INPUT($1590,$1548)
 1595     CALL SETDF(DFV,DFVP)
          CALL WRITAT(225,30,'NO SYMBOL WAS FOUNDΔ')
          CALL WRITAT(225,5,'PLEASE TRY AGAINΔ')
          GO TO 1507
 1590     IF(CT .EQ. 1) GO TO 260
C**** BUILDING THE GRID *************************
100       CONTINUE
          MARK = 1
105       CONTINUE
C*******  DISPLAY THE BUILDING PLAN FROM TREE  ***********
          CALL HOUSE
          CALL GRIDLN
          CALL DRGRID
          CALL PLACE
          CALL REGEN
          CALL MESAGE
          CALL SENDF
          CT = 1
```

```
C**** PLACING THE SYMBOLS ***********************
 500     CALL TABINT(1,$510)
       CALL CHRINT(1,$507)
         CALL SETLST
         READ(5,AGAIN)
         CALL JUMPS('AGAIN',$2,$990,$950,$800,$1150)
         CALL SETLST
         READ(5,VIEW)
         CALL JUMPS('VIEW',$1000,$1005,$1010)
 505     CALL IDLE
         CALL SWAP
         GO TO 505
 507     CALL TTY
         GO TO 505
 510     CALL GETTAB(IX,IY,IZ)
         IF(IZ) 505,505,520
 520     CALL LITEBT(0,100,1024,1024,IX,IY)
         IF(LFLAG .EQ. 1) GO TO 245
         CALL LITEBT(0,41,150,75,IX,IY)
         IF(LFLAG .EQ. 1) GO TO 530
         CALL LITEBT(0,15,150,40,IX,IY)
         IF(LFLAG .EQ. 1) GO TO 700
         GO TO 975
 245     CALL GRIDPT(IX,IY)
C****** ATTACH SYMBOL NAME WITH ATTRIBUTE OF 'SYMBOL' AND OBJECT OF WALL NAME
        MAKE BN[BRN] 'SYMBOL'*WALNAM=SYMNAM
         CALL BRANH
         SYMNUM = SYMNUM + 1
C***** PUT SYMBOL COORDINATES IN ARRAY 'SYMBOL' ***
 255     SYMBOL(SYMNUM,1) = NEWX
         SYMBOL(SYMNUM,2) = NEWY
         IND = MAP(PICK)
         NEWPIC = TYPE(IND,3)
C********* PUT SYMBOL TYPE ,INSTANCE NAME,AND BRANCH NAME IN 'SYMBOL' **
         SYMBOL(SYMNUM,3) = NEWPIC
         SYMBOL(SYMNUM,4) = SYMNAM
         SYMBOL(SYMNUM,5) = BRANAM
         POST(1) = NEWX
         POST(2) = NEWY
         POST(3) = NEWPIC
C****** STORE COORDINATES WITH ATTRIBUTE OF BRANCH NAME AND OBJECT OF WALL NAME
         MAKE BRANAM*WALNAM=[2]POST
260      CONTINUE
 265     CALL HOUSE
 266     CALL DRGRID
         CALL MESAGE
         CALL PLACE
         CALL REGEN
         CALL WRITAT(50,50,'MOVE∆')
         CALL WRITAT(50,20,'DELETE∆')
```

```
      CALL SENDF
      MARK = 2
      GO TO 500
C**** THE MOVE PART  ***********************
530   CONTINUE
      CALL HOUSE
532   CALL WRITAT(50,50,'MOVEΔ')
      CALL WRITAT(50,20,'DELETEΔ')
      CALL INTEN(0)
      CALL POS(175,56)
      CALL VEC(125,56)
      CALL VEC(137,61)
      CALL POS(125,56)
      CALL VEC(137,51)
      CALL INTEN(2)
      CALL DRGRID
      CALL MESAGE
      CALL REGEN
      CALL SENDF
      IF(FORK .EQ. 1) GO TO 560
      IF(BRANCH .EQ. 1) GO TO 560
C*** DETERMINE WHICH SYMBOL WAS SELECTED *****
      FORK = 0
      MARK = 3
      CALL TABINT(1,$550)
      CALL CHRINT(1,$545)
      CALL SETLST
      READ(5,AGAIN)
      CALL JUMPS('AGAIN',$2,$990,$950,$800,$1150)
540   CALL IDLE
      CALL SWAP
      GO TO 540
545   CALL TTY
      GO TO 540
550   CALL GETTAB(PX,PY,PZ)
      IF(PZ) 540,540,560
C***** DETERMINE WHERE THE SYMBOL IS TO BE MOVED TO *********
560   BRANCH = 1
      CALL TABINT(1,$600)
590   CALL IDLE
      CALL SWAP
      GO TO 590
600   CALL GETTAB(IX,IY,IZ)
      IF(IZ) 590,590,610
610   IF(FORK .EQ. 1) GO TO 612
      CALL GRIDPT(PX,PY)
      WALNOD = WALNAM
      PY = NEWY
      PX = NEWX
      CALL WHICH(PX,PY,$630,NUM)
```

```
 612     CALL LITEBT(0,0,1024,75,IX,IY)
         IF(LFLAG .EQ. 1) GO TO 1100
         CALL GRIDPT(IX,IY)
         SYMBOL(NUM,1) = NEWX
         SYMBOL(NUM,2) = NEWY
C*****   DESTROY OLD COORDINATES IN DATA BASE ****
         BREAK SYMBOL(NUM,5)*WALNOD=?
         POST(1) = NEWX
         POST(2) = NEWY
C*****   STORE NEW COORDINATES ********
         MAKE SYMBOL(NUM,5)*WALNOD=[2]POST
615      CONTINUE
         CALL HOUSE
         CALL DRGRID
         CALL MESAGE
         CALL WRITAT(50,50,'MOVEΔ')
         CALL WRITAT(50,20,'DELETEΔ')
         CALL PLACE
         CALL REGEN
         CALL SENDF
         MARK = 2
         BRANCH = 0
         GO TO 500
C***** SUBROUTINE WHICH DID NOT FIND A SYMBOL ******************
630      CALL HOUSE
         CALL WRITAT(225,30,'NO SYMBOL WAS FOUNDΔ')
         CALL WRITAT(225,5,'PLEASE TRY AGAINΔ')
         BRANCH = 0
         GO TO 532
C******* THE DELETE COMMAND *********************
700      CONTINUE
         CALL HOUSE
702      CALL DRGRID
         CALL MESAGE
         CALL WRITAT(50,50,'MOVEΔ')
         CALL WRITAT(50,20,'DELETEΔ')
         CALL INTEN(0)
         CALL POS(190,26)
         CALL VEC(140,26)
         CALL VEC(152,31)
         CALL POS(140,26)
         CALL VEC(152,21)
         CALL INTEN(2)
         CALL REGEN
         CALL SENDF
C*******    DETERMINE WHICH SYMBOL IS TO BE DELETED*********
         MARK = 4
         COUNT = 0
 705     CALL TABINT(1,$720)
         CALL CHRINT(1,$715)
```

```
      CALL SETLST
      READ(5,AGAIN)
      CALL JUMPS('AGAIN',$2,$990,$950,$800,$1150)
      CALL SETLST
      READ(5,VIEW)
      CALL JUMPS('VIEW',$1000,$1005,$1010)
 710  CALL IDLE
      CALL SWAP
      GO TO 710
 715  CALL TTY
      GO TO 710
 720  CALL GETTAB(DX,DY,DZ)
      IF(DZ) 710,710,730
 730  CALL GRIDPT(DX,DY)
      DX= NEWX
      DY = NEWY
      CALL WHICH(DX,DY,$760,BOXNUM)
      XBOX = BOXNUM
C*******  RETRIEVE BRANCH NUMBER  ******
      BRN = SYMBOL(5,XBOX)
C******   DESTROY ASSOCIATIONS TO SYMBOL BEING DELETED ***
     BREAK BN[BRN] 'SYMBOL'*WALNAM=?
      COUNT = 1
      IF(SYMNUM .EQ. 1) GO TO 750
      IF(XBOX .NE. SYMNUM) GO TO 735
      SYMNUM = SYMNUM - 1
      GO TO 745
 735  SYMNUM = SYMNUM - 1
C******  DELETE THE SYMBOL AND RE-ORDER 'SYMBOL' ****
      DO 740 SORT =XBOX,SYMNUM
      SYMBOL(SORT,1) = SYMBOL(SORT+1,1)
      SYMBOL(SORT,2) = SYMBOL(SORT+1,2)
      SYMBOL(SORT,3) = SYMBOL(SORT+1,3)
      SYMBOL(SORT,4) = SYMBOL(SORT+1,4)
 740  SYMBOL(SORT,5) = SYMBOL(SORT+1,5)
745   CONTINUE
      CALL HOUSE
      CALL DRGRID
      CALL MESAGE
      CALL WRITAT(50,50,'MOVEΔ')
      CALL WRITAT(50,20,'DELETEΔ')
      CALL PLACE
      CALL REGEN
      CALL SENDF
      MARK = 5
      GO TO 705
 750  SYMNUM = 0
      MARK = 2
      GO TO 500
C******* STYLUS WAS NOT POINTING AT A SYMBOL TO BE DELETED *****
```

```
 760      IF(COUNT .EQ. 0) GO TO 765
          IX = DX
          IY = DY
          GO TO 520
765       CALL HOUSE
          CALL INTEN(0)
          CALL WRITAT(225,30,'NO SYMBOL WAS FOUNDΔ')
          CALL WRITAT(225,5,'PLEASE TRY AGAINΔ')
          GO TO 702
C******************************************************************
 900      CALL SETLST
          READ(5,BACK)
          CALL JUMPS('BACK',$2)
          CALL SETDF(DFV,DFVP)
          CALL WRITAT(272,500,'TRACKING CROSS OUT OF BOUNDSΔ')
          CALL WRITAT(272,480,'TYPE AN ¤R¤ TO RETURNΔ')
          CALL SENDF
          CALL CHRINT(1,$905)
 910      CALL IDLE
          CALL SWAP
          GO TO 910
 905      CALL TTY
          GO TO 910
C******************************************************************
 950      GO TO (1505,85,1260),PART
 975      CALL SETDF(DFV,DFVP)
          CALL HOUSE
          CALL WRITAT(225,30,'THIS IS A RESTRICTED AREAΔ')
          CALL WRITAT(225,5,'PLEASE TRY AGAINΔ')
          GO TO 266
 990      CT = 0
          DO 995  IGA = 1,SYMNUM
          SYMBOL(IGA,1) = 0
          SYMBOL(IGA,2) = 0
          SYMBOL(IGA,3) = 0
 995      CONTINUE
          SYMNUM = 0
          GO TO 2
C**************************************************************
C         ---  K IS THE SCALE PARAMATER  -----------
 1000     K = 4
          VTZ = 80.0
          VTX = 0.0
          VTY = 0.0
          GO TO (100,260,530,700,745),MARK
 1005     K = 8
          VTZ = 160.0
          VTX = 0.0
          VTY = 0.0
          GO TO (100,260,530,700,745),MARK
```

```
 1010    K = 16
         VTZ = 320.0
        VTX = 0.0
         VTY = 0.0
         GO TO (100,260,530,700,745),MARK
1100     CALL HOUSE
         CALL WRITAT(225,30,'SYMBOL CANNOT BE MOVED TO A RESTRICTED Δ')
         CALL WRITAT(855,30,'AREAΔ')
         CALL WRITAT(225,5,'PLEASE TRY ANOTHER LOCATIONΔ')
         FORK = 1
         GO TO 532
 1150    CALL RESTOR
         MARK = 2
         BRANCH = 0
         GO TO 500
 800     CONTINUE
         CALL SETDF(DFV,DFVP)
         CALL SENDF
         SEGMENT RETURN
         END
C**************************************************************************
C**************************************************************************
         ATTRIB
C*****) FORMATS FIXED AND VARIABLE INFORMATION ****
C             FOR FIXTURE ATTRIBUTES --- PAX,PAY ARE COORDINATES OF
C           UPRER LEFT CORNER OF BLOCK DISPLAY INFO ***
         SUBROUTINE ATTRIB
        LIST OFF
         INCLUDE EDSPRM,LIST
         IMPLICIT INTEGER(B-Z)
         COMMON/EDSSYM/SYMNUM,SYMBOL(NUMSYM,5),TEMP(NUMSYM,3),NEWX,
     *        NEWY,SYMNAM,WALLST,WTP1
         COMMON/EDSPTB/V,U,M(10),T(10),D(10),H,C(10),A,ICHAR,ATT
         PAX = 50
         PAY = 516
         CALL WRITAT(50,200,'CONTINUEΔ')
         CALL WRITAT(25,700,'THESE ARE THE ATTRIBUTES OFΔ')
         PICTURE FORM AT 75,675
#SYMNAM#
         PICTURE FORM END
         PICTURE FORM AT PAX,PAY
VOLTAGE          :?V? VOLTS

CURRENT          :?U? AMPS

MANUFACTUER      :#M(1)##M(2)##M(3)##M(4)##M(5)#

TYPE             :#T(1)##T(2)##T(3)##T(4)##T(5)#

MODEL            :#D(1)##D(2)##D(3)##D(4)##D(5)#
```

```
MOUNTING HEIGHT:?H?

COLOR          :#C(1)#C(2)##C(3)##C(4)##C(5)#

COST           :$&2,A&
        PICTURE FORM END
        RETURN
        END
C*******************************************************************************
        SERVCE
C*****  FORMAT FOR NAMES OF SERVICE SYMBOLS **
        SUBROUTINE SERVCE
        LIST OFF
        INCLUDE EDSPRM, LIST
        INTEGER PSEX,PSEY
        COMMON/EDSSYM/SYMNUM,SYMBOL(NUMSYM,5),TEMP(NUMSYM,3),NEWX,
     *       NEWY,SYMNAM,WALLST,WTP1
        PSEX = 766
        PSEY = 700
        CALL WRITAT(328,925,'-- THE SERVICE SYMBOLS --Δ')
        CALL WRITAT(745,750,'SPECIAL DEVICESΔ')
        PICTURE FORM AT PSEX,PSEY
GENERATOR

MOTOR

METER

THERMOSTAT

ANTICIPATOR

PANEL

CONTROL PANEL

DISCONNECT

TRANSFORMER
        PICTURE FORM END
        RETURN
        END
C*******************************************************************************
        POWER
C****** FORMATS NAMES OF POWER SYMBOLS ******
        SUBROUTINE POWER
        LIST OFF
        INCLUDE EDSPRM, LIST
        INTEGER PSWX,PSWY,PPX,PPY
```

```
      COMMON/EDSSYM/SYMNUM,SYMBOL(NUMSYM,5),TEMP(NUMSYM,3),NEWX,
     *     NEWY,SYMNAM,WALLST,WTP1
      PSWX = 700
      PSWY = 550
      PPX = 700
      PPY = 868
      CALL WRITAT(325,950,'-- THE POWER SYMBOLS --Δ')
      CALL WRITAT(625,918,'CONVENIENCE OUTLETSΔ')
      PICTURE FORM AT PPX,PPY
TWO WIRE

WEATHERPROOF

EXPLOSION PROOF

GROUNDED

CLOCK

THREE WIRE

SPECIAL PURPOSE

FLOOR
      PICTURE FORM END
      CALL WRITAT(660,600,'SWITCHESΔ')
      PICTURE FORM AT PSWX,PSWY
SINGLE POLE

DOUBLE POLE

THREE-WAY

FOUR-WAY

WEATHERPROOF

PILOT LIGHT

TWO-SPEED

THREE-SPEED

AUTOMATIC

W/CONV. OUTLET

EXPLOSION PROOF

KEY OPERATED
```

```
PHOTOCELL

          PICTURE FORM END
          RETURN
          END
C*****************************************************************************
          LIGHT
C**********   FORMATS THE TEXTUAL NAMES OF LIGHTING SYMBOLS ***
          SUBROUTINE LIGHT
          LIST OFF
          INCLUDE EOSPRM, LIST
          INTEGER PFX,PFY,PSX,PSY
          COMMON/EDSSYM/SYMNUM,SYMBOL(NUMSYM,5),TEMP(NUMSYM,3),NEWX,
     *          NEWY,SYMNAM,WALLST,WTP1
          PFX = 638
          PFY = 757
          PSX = 638
          PSY = 550
          CALL WRITAT(275,925,'-- THE LIGHTING SYMBOLS --Δ')
          CALL WRITAT(590,807,'FIXTURESΔ')
          PICTURE FORM AT PFX,PFY
SURFACE FLOURESCENT

RECESSED FLOURESCENT

SURFACE INCANDESCENT

RECESSED INCANDESCENT

PULL SWITCH
          PICTURE FORM END
          CALL WRITAT(590,600,'SWITCHESΔ')
          PICTURE FORM AT PSX,PSY
SINGLE POLE

DOUBLE POLE

THREE-WAY

FOUR-WAY

WEATHERPROOF

PILOT LIGHT

TWO-SPEED

THREE-SPEED
```

AUTOMATIC

W/CONV. OUTLET

EXPLOSION PROOF

KEY OPERATED

PHOTOCELL

```
         PICTURE FORM END
         RETURN
         END
C*****************************************************************************
         TYPATT
         SUBROUTINE TYPATT
         LIST OFF
C****** STORES 50 WORD BLOCK OF ATTRIBUTE VALUES ***
         INCLUDE EDSPRM, LIST
         INTEGER V,U,M,T,D,H,C,ATT
         COMMON/EDSSYM/SYMNUM,SYMBOL(NUMSYM,5),TEMP(NUMSYM,3),NEWX,NEWY,
     C   SYMNAM,WALLST,WTP1
         COMMON/EDSATB/V,U,M(10),T(10),D(10),H,C(10),A,ICHAR,ATT
         INTEGER DATABL(50)
         EQUIVALENCE(DATABL,DAAAT)
         REAL DAAAT(50)
         ALLOW  0
         DATABL(1) = V
         DATABL(2) = U
         DO 10  I = 1,10
         DATABL(I+2) = M(I)
 10      CONTINUE
         DO 20  I = 1,10
 20      CONTINUE
         DATABL(I+12) = T(I)
         DO 30  I = 1,10
         DATABL(I+22) = D(1)
 30      CONTINUE
         DATABL(33) = H
         DO 40  I = 1,10
         DATABL(I+33) = C(1)
 40      CONTINUE
         DAAAT(44) = A
         BREAK 'ATTRIB'*SYMNAM=?
         MAKE  'ATTRIB'*SYMNAM=[50]DATABL
         RETURN
         END
C*****************************************************************************
         TABW
         SUBROUTINE TABW
```

```
         LIST SOURCE CODE
C        TABLE A WALL SPACEFORM
         INCLUDE EOSPRM,LIST
C
         INTEGER CNODE,CLEVEL
         COMMON /APYTRD/CNODE,CLEVEL
C
         REAL U(3,4)
         COMMON /TNSMRX/U
         INTEGER WALLST(5,WTPL)
         REAL RWALLT(5,WTPL)
         EQUIVALENCE (WALLST,RWALLT)
         INTEGER WTP1
         COMMON/EDSSYM/SYMNUM,SYMBOL(NUMSYM,5),TEMP(NUMSYM,3),NEWX,NEWY,
     C   SYMNAM,WALLST,WTP1
C
         COMMON /ERR/IERR
         INTEGER IERR
C
         REAL PA(60)
         INTEGER IPA(60)
         EQUIVALENCE (PA,IPA)
         COMMON /PA/PA
         REAL STX,STY,STZ,SRX,SRY,SRZ,SSX,SSY,SSZ
         COMMON /STR/STX,STY,STZ,SRX,SRY,SRZ,SSX,SSY,SSZ
         REAL DISPT1,DISPT3(3),DISPT4(3)
         CALL TYPOUT('(6HCNODE=,A6,2H¤Δ)',CNODE)
         ALLOW  0
         IF('$TYPE'*CNODE='$WALL')GO TO 50
         RETURN
50       WTP1=WTP1+1
         IF(WTP1.LE.WTPL)GO TO 200
C
C        NO MORE ROOM.
         WTP1=WTPL
         IERR=5
         RETURN
C
200      IF('$NOTYP'*CNODE='$TRM')GO TO 205
         CALL TYPOUT('WALL IS NOT A PRIMATIVE¤Δ')
         RETURN
  205    [60]PA<='$PRIM'*CNODE=?
         I1 = IPA(1)+4
         DISPT3(1) = ABS(PA(I1))*SSX
         DISPT3(2)=0.0
         DISPT3(3)=ABS(PA(I1+2))*SSX
         DISPT1=AMAX1(ABS(PA(I1  ))*SSX,ABS(PA(I1+2))*SSZ)
         IF(DISPT3(1).NE.DISPT1)DISPT3(1)=0.0
         IF(DISPT3(3).NE.DISPT1)DISPT3(3)=0.0
         CALL APPYRM(DISPT3,DISPT4)
```

```
      RWALLT(2,WTP1)=DISPT4(1)
      RWALLT(3,WTP1)=DISPT4(3)
      DISPT3(1)=-DISPT3(1)
      DISPT3(3)=-DISPT3(3)
      CALL APPYRM(DISPT3,DISPT4)
      RWALLT(4,WTP1)=DISPT4(1)
      RWALLT(5,WTP1)=DISPT4(3)
      WALLST(1,WTP1)=CNODE
      PRINT 2000,(WALLST(I,WTP1),I=1,5)
2000  FORMAT(1X,A6,4(1X,F10.5))
      RETURN
      END
C*************************************************************************
      APYTRE
      SUBROUTINE APYTRE(TOPNOD,FNDOWN,FNUP,STPCOD,ATTR,VAL)
      LIST SOURCE CODE
C     APPLY A FUNCTION TO A TREE
C
C     TOPNOD CONTAINS THE NAME OF THE TOP NODE OF THE TREE
C     FN CONTAINS THE NAME OF THE FUNCTIONS TO BE CALLED AT EACH NODE
C     STPCOD CONTAINS THE STSTOP CODE (1 OR 2 PRESENTLY)
C
      INTEGER TOPNOD,STPCOD,ATTR,VAL
      EQUIVALENCE (ATTR,LEVEL)
C
C
     INCLUDE DSPRMS, LIST
C
C
C
      INTEGER IERR
      COMMON /ERR/IERR
C
      REAL U(3,4)
      COMMON /TNSMRX/U
C
      INTEGER NAME1(WPN)
C
      REAL TX,TY,TZ,RX,RY,RZ,SX,SY,SZ
      COMMON /TR/TX,TY,TZ,RX,RY,RZ,SX,SY,SZ
C
      REAL VTX,VTY,VTZ,VRX,VRY,VRZ,VSX,VSY,VSZ
      COMMON /VP/VTX,VTY,VTZ,VRX,VRY,VRZ,VSX,VSY,VSZ
      REAL STX,STY,STZ,SRX,SRY,SRZ,SSX,SSY,SSZ
      COMMON /STR/STX,STY,STZ,SRX,SRY,SRZ,SSX,SSY,SSZ
C
      INTEGER TNME
C
C
C
```

```
      REAL TRS(9)
      INTEGER TRSNME
      COMMON /TRS/TRSNME,TRS
C
C
C     RELEVANT DATA WHEN 'FN' IS CALLED
      INTEGER CLEVEL          @CURRENT LEVEL DURING TREE PROCESSING
      COMMON /APYTRD/TNME,CLEVEL
C
C
      INTEGER FNDOWN,FNUP
C
C
C     CHECK STOP CODE VALIDITY
      ALLOW  0
      IF((STPCOD.EQ.1).OR.(STPCOD.EQ.2))GO TO 100
C     STOP CODE OUT OF RANGE
      IERR=6
      RETURN
C
C     INITIALIZE THE TRANSLATION, ROTATION AND SCALING PARAMETERS
100   U(1,4)=0
      U(2,4)=0
      U(3,4)=0
      DO 10 I=1,3
      DO 10 J=1,3
      U(I,J)=0.0
      IF(I.EQ.J)U(I,J)=1.0
10    CONTINUE
      IERR=0
      TX = 0.0
      TY = 0.0
      TZ = 0.0
      RX=0.0
      RY=0.0
      RZ=0.0
      SSX=1
      SSY=1
      SSZ=1
      SX=1.0
      SY=1.0
      SZ=1.0
      CALL UPDRM
C
C     INITIALIZE THE TREE FOLLOWING STACK
      NEXT NODE CURRENT TOPNOD,'$MBMR'
      NEXT NODE DOWN BN[IBN]=TNME
      IF(IBN.EQ.0)GO TO 2040
      NAME1(1)=TOPNOD
      CLEVEL=1
```

```
1200    FLD(0,24,TRSNME)=6H$TRS
        FLD(24,12,TRSNME)=IBN
        [9]TX<=TRSNME*NAME1(1)=?
        PRINT 3002,TX,TY,TZ,RX,RY,RZ,SX,SY,SZ
3002    FORMAT(1X,5H3002 ,9F10.5)
        CALL UPDRM
C
C
C       EXECUTE USER SPECIFIED FN AFTER GOING DOWN TO THIS NODE
        IF(FNDOWN.NE.0)CALL FNDOWN
C
C       TEST IF PRESENT NODE MEETS STOP CODE CONDITION
        IF((STPCOD.EQ.1).AND.(LEVEL.EQ.CLEVEL))GO TO 1699
        IF(STPCOD.NE.2)GO TO 1629
        IF(ATTR*TNME=VAL)GO TO 1699
        GO TO 1629
C
C
1699    CONTINUE
        TNME=NAME1(1)
C       ZERO BYTE ENDS PRIMATIVE SPECIFICATION
1627    TX=-TX
        TY=-TY
        TZ=-TZ
        RX=-RX
        RY=-RY
        RZ=-RZ
        CALL UPDRMB
1626    NEXT NODE UP=NAME1(1)
        CLEVEL=CLEVEL-1
        PRINT 5000,NAME1(1)
5000    FORMAT(1X,6H1626  ,A6)
        IF(NAME1(1).EQ.0)GO TO 2040
1628    NEXT NODE RIGHT BN[IBN]
        PRINT 5001,IBN
5001    FORMAT(1X,6H1628  ,I5)
        IF(IBN.EQ.0)GO TO 2100
C
1629    NAME1(1)=TNME
        NEXT NODE DOWN BN[IBN]=TNME
        CLEVEL=CLEVEL+1
        IF(IBN.NE.0)GO TO 1200
1634    NEXT NODE UP BN[IBN]=NAME1(1)
        CLEVEL=CLEVEL-1
        GO TO 1627
C
C
C       FINISHED.
2040    CONTINUE
        RETURN
```

```
C
C       CURRENT NODE PROCESSED POP THE STACK
2100    NEXT NODE UP BNCIBN] = NAME1(1)
        CLEVEL=CLEVEL-1
        PRINT 5003,IBN,NAME1(1)
5003    FORMAT(1X,6H2100  ,I4 A6)
        IF(NAME1(1).EQ.0)GO TO 2040
        FLD(0,24,TRSNME)=6H$TRS
        FLD(24,12,TRSNME)=IBN
        L9JTX<=TRSNME*NAME1(1)=?
        PRINT 3001,TX,TY,TZ,RX,RY,RZ,SX,SY,SZ
3001    FORMAT(1X,5H3001 ,9F10.5)
        TX=-TX
        TY=-TY
        TZ=-TZ
        RX=-RX
        RY=-RY
        RZ=-RZ
        CALL UPDRMG
        TNME=NAME1(1)
C
C       EXECUTE USER SPECIFIED FN AFTER MOVING UP TO THIS NODE
        IF(FNUP.NE.0)CALL FNUP
C
C
        GO TO 1628
        END
C*****************************************************************************
        GETVAL
        SUBROUTINE GETVAL
        LIST OFF
C****** CHECK FOR ATTRIBUTE VALUES ALREADY STORED ***
C****** IF OLD VALUES ARE PRESENT THEY ARE STORED IN 'BLOCK'
        INTEGER V,U,M,T,D,H,C
        INTEGER BLOCK(50)
        REAL DAAAT(50)
        COMMON/EDSATB/V,U,M(10),T(10),D(10),H,C(10),A,ICHAR,ATT
         ALLOW  0
        EQUIVALENCE (BLOCK,DAAAT)
        IF('ATTRIB'*SYMNAM=?) GO TO 10
        DO 7717 INIT1=1,10
        M(INIT1)=6H
        T(INIT1)=6H
        D(INIT1)=6H
        C(INIT1)=6H
 7717   CONTINUE
        V=5H
        U=5H
        H=5H
        A=5H
```

```
      GO TO 100
10    CONTINUE
      C50 JBLOCK<='ATTRIB'+SYMNAME?
      V = BLOCK(1)
      U = BLOCK(2)
      DO 20  I = 1,10
      M(I) = BLOCK(I+2)
20    CONTINUE
      DO 30  I= 1,10
      T(I) = BLOCK(I+12)
30    CONTINUE
      DO 40  I = 1,10
      D(I) = BLOCK(I+22)
40    CONTINUE
      H = BLOCK(33)
      DO 50  I = 1,10
      C(I) = BLOCK(I+33)
      A = DAAAT(44)
50    CONTINUE
100   CONTINUE
      RETURN
      END
C*****************************************************************************
C*****************************************************************
C***** DISPLAYS BUILDING PLAN FROM TREE ***
      SUBROUTINE HOUSE
      IMPLICIT INTEGER(A-Z)
      INCLUDE EDSPRM, LIST
      INCLUDE DSPRMS, LIST
      INTEGER DFV(DFVL),DFVP,DFVF
      INTEGER DFB(DFBL,WPN)
      COMMON /DF/DFV,DFB,DFVP,DFVF
      COMMON /CROOT/O1,O2,PLNODE
      COMMON/EDSSYM/SYMNUM,SYMBOL(NUMSYM,5),TEMP(NUMSYM,3),NEWX,NEWY,
     C SYMNAM,WALLST,WTP1
      INTEGER DSPTRE(WPN)
      COMMON /DSPTRE/DSPTRE
      COMMON /VP/VTX,VTY,VTZ,VRX,VRY,VRZ,VSX,VSY,VSZ
      REAL VTX,VTY,VTZ,VRX,VRY,VRZ,VSX,VSY,VSZ
      LOGICAL DNSFLG,FBALL,FPEDAL,FMAKHT,FRESND,FSETVP
      COMMON /FLAGS/DNSFLG,FBALL,FPEDAL,FMAKHT,FRESND,FSETVP
      FSETVP=.FALSE.
      VRX = -90.0
      DSPTRE(1) = PLNODE
      DFVP=0
      CALL DISP
      CALL GRG(' ,0)
      RETURN
      END
C*****************************************************************************
```

```
      SUBROUTINE WALGRD
C     SET UP IN TABLE FORM THE COORDINATES OF ALL 'WALLS' IN AN
C     OBJECT.
C
      INCLUDE EDSPRM,LIST
      INTEGER WALLST(5,WTPL)
      INTEGER WTP1
      COMMON/EDSSYM/SYMNUM,SYMBOL(NUMSYM,5),TEMP(NUMSYM,3),NEWX,NEWY,
     C     SYMNAM,WALLST,WTP1
      EXTERNAL TABW
C
      INTEGER FLRPLN
      COMMON /CROOT/DUM(2),FLRPLN
      WTP1=0
C
      CALL APYTRE(FLRPLN,TABW,0,2,'$TYPE','$WALL')
      RETURN
      END
C****************************************************************************
C**** SUBROUTINE TO CREATE GRID POINTS *****
      SUBROUTINE GRIDLN
      IMPLICIT INTEGER(A-Z)
      COMMON/EDSGRD/GRID(50,2)
      GRID(1,1) = 0
      GRID(1,2) = 0
      VALUE = 0
      DELTA = 33
      DO 100 J = 2,31
      VALUE = VALUE + DELTA
      GRID(J,1) = VALUE
 100  GRID(J,2) = VALUE
      RETURN
      END
C****************************************************************************
C**** SUBROUTINE TO DRAW THE GRID *******
      SUBROUTINE DRGRID
      IMPLICIT INTEGER(A-Z)
      INCLUDE EDSPRM, LIST
      COMMON/EDSGRD/GRID(50,2)
      COMMON/EDSSYM/SYMNUM,SYMBOL(NUMSYM,5),TEMP(NUMSYM,3),NEWX,NEWY,
     C     SYMNAM,WALLST,WTP1
      LOWY = 100
      LEFTX = 0
      RIGHTX = 1023
      UPY = 1023
      CALL INTEN(1)
      DO 10 LOG = 1,31
      KX = GRID(LOG,1)
      KY = GRID(LOG,2)
      CALL POS(KX,LOWY)
```

```
      CALL VEC(KX,UPY)
      CALL POS(LEFTX,KY)
 10   CALL VEC(RIGHTX,KY)
      CALL INTEN(2)
      CALL POS(0,100)
      CALL VEC(1023,100)
      RETURN
      END
C*********************************************************************
C*****************************************************
C***** DRAWS TRIANGLE FOR SYMBOLS TX,TY ARE CENTER OF TRIANGLE**
      SUBROUTINE TRI(TX,TY)
      INCLUDE EDSPRM, LIST
      IMPLICIT INTEGER(A-Z)
      COMMON/EDSSYM/SYMNUM,SYMBOL(NUMSYM,5),TEMP(NUMSYM,3),NEWX,NEWY,
     C SYMNAM,WALLST,WTP1
      CALL INTEN(0)
      VX = TX - 8
      VY = TY - 8
      CALL POS(VX,VY)
      CALL VEC(TX,VY + 14)
      CALL VEC(VX +16,VY)
      CALL VEC(VX,VY)
      CALL INTEN(2)
      RETURN
      END
C*********************************************************************
C*****************************************************
C***** SUBROUTINE TO DRAW A CIRCLE ***********
C****  CIX AND CIY ARE THE CENTER COORDINATES ******
C****  R IS THE RADIUS AND N IS THE NUMBER OF SEGMENTS ***
      SUBROUTINE CIRCLE(CIX,CIY)
      INCLUDE EDSPRM, LIST
      REAL THETA,ANGLE,N
      INTEGER R,CIX,CIY,AA,X,Y,DF
      COMMON/EDSSYM/SYMNUM,SYMBOL(NUMSYM,5),TEMP(NUMSYM,3),NEWX,NEWY,
     C SYMNAM,WALLST,WTP1
      CALL INTEN(0)
      R = 10
      N = 10.0
      THETA = (360/N)/6.28
      ANGLE = THETA
      CALL POS(CIX,CIY + R)
      DO 10  AA = 1,12
      X = R*SIN(ANGLE)
      Y = R*COS(ANGLE)
        CALL VEC(X + CIX,Y + CIY)
 10   ANGLE = ANGLE + THETA
      CALL INTEN(2)
      RETURN
```

```
      END
C***********************************************************************
C**** SUBROUTINE TO DRAW THE SMALL TEE FOR SYMBOLS ***
C****  SMX AND SMY ARE THE COORDINATES OF THE INTEGSECTION ***
      SUBROUTINE SMTEE(SMX,SMY)
      INCLUDE EDSPRM, LIST
      INTEGER SMX,SMY
      COMMON/EDSSYM/SYMNUM,SYMBOL(NUMSYM,5),TEMP(NUMSYM,3),NEWX,NEWY,
     C    SYMNAM,WALLST,WTP1
      CALL INTEN(0)
      CALL POS(SMX,SMY)
      CALL VEC(SMX-8,SMY)
      CALL POS(SMX-8,SMY-5)
      CALL VEC(SMX-8,SMY+5)
      CALL INTEN(2)
      RETURN
      END
C***********************************************************************
C****************************************************
C**** SUBROUTINE TO DRAW  A BOX ***
C***  W IS WIDTH,H IS HIGHT,BX AND BY ARE LOWER LEFT CORNER ***
      SUBROUTINE BOX(W,H,BX,BY)
      INCLUDE EDSPRM, LIST
      IMPLICIT INTEGER(A-Z)
      COMMON/EDSSYM/SYMNUM,SYMBOL(NUMSYM,5),TEMP(NUMSYM,3),NEWX,NEWY,
     C    SYMNAM,WALLST,WTP1
      CALL INTEN(0)
      CALL POS(BX,BY)
      CALL VEC(BX+W,BY)
      CALL VEC(BX+W,BY+H)
      CALL VEC(BX,BY+H)
      CALL VEC(BX,BY)
      CALL INTEN(2)
      RETURN
      END
C***********************************************************************
C*** SUBROUTINE TO DRAW A TEE ***
C*** TX,TY ARE THE COORDINATES OF INTERSECTION ****
      SUBROUTINE TEE(TX,TY)
      INCLUDE EDSPRM, LIST
      INTEGER TX,TY
      COMMON/EDSSYM/SYMNUM,SYMBOL(NUMSYM,5),TEMP(NUMSYM,3),NEWX,NEWY,
     C    SYMNAM,WALLST,WTP1
      CALL POS(TX,TY-5)
      CALL VEC(TX,TY+5)
      CALL POS(TX,TY)
      CALL VEC(TX+25,TY)
      RETURN
      END
C***********************************************************************
```

```
C**** SUBROUTINE TO DRAW A DOUBLE TEE ****
C**** TTX,TTY ARE THE COORDINATES OF INTERSECTION ***
      SUBROUTINE TOTEE(TTX,TTY)
      INTEGER TTX,TTY
      CALL POS(TTX,TTY-7)
      CALL VEC(TTX,TTY+7)
      CALL POS(TTX,TTY-2)
      CALL VEC(TTX+30,TTY-2)
      CALL POS(TTX,TTY+2)
      CALL VEC(TTX+30,TTY+2)
      RETURN
      END
C***************   ****************************************************************
C************************************************************
C**** SUBROUTINE TO CAPTURE THE BRANCH NUMBER OF CURRENT NODE
      SUBROUTINE BRANH
      IMPLICIT INTEGER(A-Z)
      COMMON/EDSDAT/WALNAM,KKK,BRN,BRANAM
      TEMP10 = 'SYSP'
      CALL ENCODE(NNN)
      WRITE(23,100) BRN
 100  FORMAT(I2)
      FLD(24,12,TEMP10) = FLD(0,12,NNN)
      BRANAM = TEMP10
      RETURN
      END
C********************************************************************************
C************************************************************
      SUBROUTINE PLACE
C          SCALES THE SYMBOL COORDINATES FROM 'SYMBOL' TO 'TEMP'
C********* SYMBOL IS LOCAL STORAGE ,TEMP IS USED FOR DISPLAY**
      INCLUDE EDSPRM, LIST
      IMPLICIT INTEGER(A-Z)
      REAL VTX,VTY,VTZ,VRX,VRY,VRZ,VSX,VSY,VSZ
      COMMON /VP/VTX,VTY,VTZ,VRX,VRY,VRZ,VSX,VSY,VSZ
      COMMON/EDSSCL/K
      COMMON/EDSSYM/SYMNUM,SYMBOL(NUMSYM,5),TEMP(NUMSYM,3),NEWX,NEWY,
     C    SYMNAM,WALLST,ATP1
      DO 10 I = 1,SYMNUM
      TEMP(I,1) =(((SYMBOL(I,1)-256)/(VTZ*1200/2000.0+1))+256 )*2
      TEMP(I,2) =(((SYMBOL(I,2)-256)/(VTZ*1200/2000.0+1))+256)*2
 10   TEMP(I,3) = SYMBOL(I,3)
      RETURN
      END
C********************************************************************************
C*****  SPECIFIES THE FIRST 4 CHARACTERS OF SYMBOL NAMES ***
      SUBROUTINE NAMELD
      INTEGER TYPE
      COMMON/EDSNME/TYPE(50,3),MAP(50),PICK
      TYPE(1,1)='PTWW'
```

```
      TYPE(2,1)='PWHP'
      TYPE(3,1)='PEXP'
      TYPE(4,1)='PGRD'
      TYPE(5,1)='PCLK'
      TYPE(6,1)='PTHW'
      TYPE(7,1)='PSPP'
      TYPE(8,1)='PFLR'
      TYPE(9,1)='SNPL'
      TYPE(10,1)='DBPL'
      TYPE(11,1)='THWY'
      TYPE(12,1)='FRWY'
      TYPE(13,1)='WHPF'
      TYPE(14,1)='PLLT'
      TYPE(15,1)='TWSP'
      TYPE(16,1)='THSP'
      TYPE(17,1)='ADDR'
      TYPE(18,1)='WCOT'
      TYPE(19,1)='EXPF'
      TYPE(20,1)='KYOP'
      TYPE(21,1)='PHCL'
      TYPE(22,1)='SGTR'
      TYPE(23,1)='SMTR'
      TYPE(24,1)='SMER'
      TYPE(25,1)='STHT'
      TYPE(26,1)='SATR'
      TYPE(27,1)='SNPL'
      TYPE(28,1)='SCPL'
      TYPE(29,1)='SDCT'
      TYPE(30,1)='STRF'
      TYPE(31,1)='LSFL'
      TYPE(32,1)='LRFL'
      TYPE(33,1)='LRIN'
      TYPE(34,1)='LREN'
      TYPE(35,1)='LPSW'
      TYPE(36,1)='LSIN'
      TYPE(37,1)='LREI'
      TYPE(38,1)='LPS2'
      DO 10  I = 1,38
      MAP(I) = I
      TYPE(I,3) = I
      TYPE(I,2) = 15
 10   CONTINUE
      RETURN
      END
C***********************************************************************:
C**** INITIALIZES  ATTRIBUTE VALUE STORAGE **
      BLOCK DATA
      REAL A
      INTEGER V,U,M,T,D,H,C
      COMMON/EDSATB/V,U,M(10),T(10),D(10),H,C(10),A,ICHAR
```

```
      DATA V,U,M(1),M(2),M(3),M(4),M(5),M(6),M(7),M(8),M(9),M(10),
     C          T(1),T(2),T(3),T(4),T(5),T(6),T(7),T(8),T(9),T(10),
     C          D(1),D(2),D(3),D(4),D(5),D(6),D(7),D(8),D(9),D(10),
     C        H,C(1),C(2),C(3),C(4),C(5),C(6),C(7),C(8),C(9),C(10),A/
     *           44*0050505050505/
      END
C*************************************************************************
C****** THE ROUTINE THAT CREATES DISPLAY OF ALL SERVICE SYMBOLS ***
      SUBROUTINE SERSYM
C**** GENERATOR
      CALL CIRCLE(706,705)
      CALL WRITAT(706,701,'GΔ')
C*** MOTOR   *******
      CALL CIRCLE(706,673)
      CALL WRITAT(706,667,'MΔ')
C**** METER *****
      CALL CIRCLE(706,641)
      CALL BOX(20,20,696,631)
      CALL WRITAT(701,632,'MΔ')
C**** THERMOSTAT ****
      CALL BOX(16,16,690,601)
      CALL WRITAT(695,600,'TΔ')
      CALL SMTEE(690,609)
C*** ANTICIPATOR ****
      CALL BOX(16,16,690,569)
      CALL WRITAT(693,572,'AΔ')
      CALL SMTEE(690,577)
C*** PANEL ****
      CALL BOX(20,10,696,540)
C**** CONTROL PANEL ****
      CALL BOX(20,10,696,508)
      CALL INTEN(0)
      CALL POS(696,508)
      CALL VEC(716,518)
      CALL POS(716,508)
      CALL VEC(696,518)
     CALL INTEN(2)
C**** DISCONNECT ****
      CALL BOX(15,8,696,477)
      CALL INTEN(0)
      CALL POS(711,481)
      CALL VEC(716,481)
      CALL VEC(716,489)
      CALL INTEN(2)
C***  TRANSFORMER ****
      CALL BOX(20,10,6[illegible],453)
      RETURN
      END
C**** DISPLAYS ALL LIGHTING SYMBOLS ***
      SUBROUTINE LITSYM
```

```
      INCLUDE EDSPRM, LIST
      COMMON/EDSSYM/SYMNUM,SYMBOL(NUMSYM,5),TEMP(NUMSYM,3),NEWX,NEWY,
     C   SYMNAM,WALLST,WTP1
C*** SURFACE FLOURESCENT ****
      CALL BOX(40,20,563,754)
      CALL CIRCLE(573,764)
C**** RECESSED FLOURESENT ****
      CALL BOX(40,20,563,722)
      CALL CIRCLE(573,732)
C*** SURFACE INCANDESCENT ****
      CALL CIRCLE(530,700)
      CALL CIRCLE(583,700)
C*** RECESSED INCANDESCENT ***
      CALL CIRCLE(530,668)
      CALL CIRCLE(583,668)
C*** PULL SWITCH ***
      CALL CIRCLE(530,636)
      CALL CIRCLE(583,636)
      CALL SUB
      CALL WRITAT(596,666,'NΔ')
      CALL WRITAT(596,624,'PSΔ')
      CALL WRITAT(583,722,'FΔ')
      CALL WRITAT(596,688,'MΔ')
      CALL WRITAT(583,754,'GΔ')
      CALL LCHAR
C*****  SWITCHES  *****  ** LIGHT ****
      CALL LCHAR
C**** SINGLE POLE *******                       7
      CALL TEE(566,559)
      CALL WRITAT(576,553,'SΔ')
C**** DOUBLE POLE  *****
      CALL TEE(566,527)
      CALL WRITAT(576,521,'SΔ')
C***   THREE WAY  *****
      CALL TEE(566,495)
      CALL WRITAT(576,489,'SΔ')
C***** FOUR WAY  ******
      CALL TEE(566,463)
      CALL WRITAT(576,457,'SΔ')
C**** WEATHERPROOF  ******
      CALL TEE(566,431)
      CALL WRITAT(576,425,'SΔ')
C**** PILOT LIGHT  *****
      CALL TEE(566,399)
      CALL WRITAT(576,393,'SΔ')
C**** TWO-SPEED *****
      CALL TEE(566,367)
      CALL WRITAT(576,361,'SΔ')
C**** THREE-SPEED *****
      CALL TEE(566,335)
```

```
         CALL WRITAT(576,329,'SΔ')
C**** AUTO DOOR ****
         CALL TEE(566,303)
         CALL WRITAT(576,297,'SΔ'
C***** W/CONV. OUTLET ****
         CALL WRITAT(576,261,'SΔ')
C**** EXPLOSION PROOF ****
         CALL TEE(566,239)
         CALL WRITAT(576,231,'SΔ')
C*** KEY OPERATED *****
         CALL TEE(566,207)
         CALL WRITAT(576,201,'SΔ')
C*** PHOTO CELL****
         CALL TEE(566,175)
         CALL WRITAT(576,169,'SΔ')
C**** SUBSCRIPTS FOR LIGHT SWITCHES ***
         CALL SUB
         CALL WRITAT(589,511,'2Δ')
         CALL WRITAT(589,480,'3Δ')
         CALL WRITAT(589,447,'4Δ')
         CALL WRITAT(589,415,'WPΔ')
         CALL WRITAT(589,383,'PΔ')
         CALL WRITAT(589,351,'2SPΔ')
         CALL WRITAT(589,319,'3SPΔ')
         CALL WRITAT(589,287,'DΔ')
         CALL WRITAT(589,223,'EXΔ')
         CALL WRITAT(589,191,'KΔ')
         CALL WRITAT(589,165,'OΔ')
         RETURN
         END
C*************************************************************************************
C***** DISPLAYS POWER SYMBOLS ****
         SUBROUTINE POWSYM
         INCLUDE EDSPRM, LIST
         COMMON/EDSSYM/SYMNUM,SYMBOL(NUMSYM,5),TEMP(NUMSYM,3),NEWX,NEWY,
     C   SYMNAM,WALLST,WTP1
C******TWO WIRE
         CALL CIRCLE(640,870)
         CALL TOTEE(628,870)
C    ***WEATHERPROOF ****
         CALL CIRCLE(640,840)
         CALL TOTEE(628,840)
         CALL SUB
         CALL WRITAT(653,828,'WPΔ')
C****EXPLOSION PROOF ****
         CALL CIRCLE(640,806)
         CALL TOTEE(628,806)
         CALL WRITAT(653,794,'EXΔ')
C*** GROUNDED ****
         CALL CIRCLE(640,774
```

```
          CALL TOTEE(628,774)
          CALL WRITAT(653,762,'GΔ')
C****** CLOCK  ****
          CALL CIRCLE(640,744)
          CALL TOTEE(628,744)
          CALL WRITAT(653,732,'CΔ')
C**** THREE WIRE ***
          CALL CIRCLE(640,712)
C***** SPECIAL PURPOSE ****
          CALL CIRCLE(640,680)
          CALL TRI(640,683)
C**** FLOOR*******
          CALL CIRCLE(640,648)
          CALL DOT(640,653)
C*****  SWITCHES  *****   ** POWER ****
          CALL LCHAR
C**** SINGLE POLE ******
          CALL TEE(628,559)
          CALL WRITAT(639,553,'SΔ')
C**** DOUBLE POLE  *****
          CALL TEE(628,527)
          CALL WRITAT(639,521,'SΔ')
C***   THREE WAY  *****
          CALL TEE(628,495)
          CALL WRITAT(639,489,'SΔ')
C***** FOUR WAY  ******
          CALL TEE(628,463)
          CALL WRITAT(639,457,'SΔ')
C**** WEATHERPROOF  ******
          CALL TEE(628,431)
          CALL WRITAT(639,425,'SΔ')
C**** PILOT LIGHT  *****
          CALL TEE(628,399)
          CALL WRITAT(639,393,'SΔ')
C**** TWO-SPEED *****
          CALL TEE(628,367)
          CALL WRITAT(639,361,'SΔ')
C**** THREE-SPEED *****
          CALL TEE(628,335)
          CALL WRITAT(639,329,'SΔ')
C**** AUTO DOOR ****
          CALL TEE(628,303)
          CALL WRITAT(639,297,'SΔ')
C***** W/CONV. OUTLET ****
          CALL WRITAT(639,261,'SΔ')
C**** EXPLOSION PROOF ****
          CALL TEE(628,239)
          CALL WRITAT(639,231,'SΔ')
C*** KEY OPERATED *****
          CALL TEE(628,2?7)
```

```
         CALL WRITAT(639,201,'SΔ')
C*** PHOTO CELL****
         CALL TEE(626,175)
         CALL WRITAT(639,169,'SΔ')
C**** SUBSCRIPTS FOR POWER SWITCHES ***
         CALL SUB
         CALL WRITAT(651,511,'2Δ')
         CALL WRITAT(651,479,'3Δ')
         CALL WRITAT(651,447,'4Δ')
         CALL WRITAT(651,415,'WPΔ')
         CALL WRITAT(651,383,'PΔ')
         CALL WRITAT(651,351,'2SPΔ')
         CALL WRITAT(651,319,'3SPΔ')
         CALL WRITAT(651,287,'DΔ')
         CALL WRITAT(651,223,'EXΔ')
         CALL WRITAT(651,191,'KΔ')
         CALL WRITAT(651,170,'OΔ')
         RETURN
         END
C*****************************************************************************
C*******************************************************
         SUBROUTINE RESTOR
C             RESTORES THE PROGRAM AND DISPLAY ON 'RETURN' COMMAND
         INCLUDE EDSPRM, LIST
         IMPLICIT INTEGER(A-Z)
         COMMON/EDSSYM/SYMNUM,SYMBOL(NUMSYM,5),TEMP(NUMSYM,3),NEWX,NEWY,
     C   SYMNAM,WALLST,WTP1
         CALL HOUSE
         CALL INTEN(1)
         CALL DRGRID
         CALL INTEN(0)
         CALL REGEN
         CALL WRITAT(50,50,'MOVEΔ')
         CALL WRITAT(50,20,'DELETEΔ')
         CALL MESAGE
         CALL SENDF
         RETURN
         END
C*****************************************************************************
         SUBROUTINE WHICH(WX,WY,$,INDEX)
C***** SUBROUTINE TO DETERMINE WHICH SYMBOL WAS POINTED AT *****
         INCLUDE EDSPRM,LIST
         IMPLICIT INTEGER(A-Z)
         COMMON/EDSSYM/SYMNUM,SYMBOL(NUMSYM,5),TEMP(NUMSYM,3),NEWX,NEWY,
     C   SYMNAM,WALLST,WTP1
         REAL VTX,VTY,VTZ,VRX,VRY,VRZ,VSX,VSY,VSZ
         COMMON /VPS/VTX,VTY,VTZ,VRX,VPY,VRZ,VSX,VSY,VSZ
         WX = WX - 15
         WY = WY - 15
         XCOM1 = (((WX/2)- [illegible]*1200/2000+1))+256
```

```
      YCOM1 = (((WY/2)-256)*(VTZ*1200/2000+1))+256
      WX = WX + 30
      WY = WY + 30
      XCOM2 = (((WX/2)-256)*(VTZ*1200/2000+1))+256
      YCOM2 = (((WY/2)-256)*(VTZ*1200/2000+1))+256
      DO 100  I = 1,SYMNUM
      IF(((SYMBOL(I,1) .GE. XCOM1) .AND. (SYMBOL(I,1) .LE. XCOM2))
     C .AND. ((SYMBOL(I,2) .GE. YCOM1) .AND. (SYMBOL(I,2) .LE. YCOM2
     C ))) GO TO 125
  100 CONTINUE
      RETURN 3
  125 INDEX = I
      RETURN
      END
      SUBROUTINE GRIDPT(IX,IY)
C*** SUBROUTINE  TO SELECT CLOSEST GRID INTERSECTION ***
      INCLUDE EDSPRM,LIST
      REAL EPDIS
      INTEGER XSCOP1,XSCOP2,YSCOP1,YSCOP2
      INTEGER IC1(2),IC2(2),WTP1,WALLST(5,WTPL)
      REAL VTX,VTY,VTZ,VRX,VRY,VRZ,VSX,VSY,VSZ
      COMMON /VP/VTX,VTY,VTZ,VRX,VRY,VRZ,VSX,VSY,VSZ
      REAL RWALLT(5,WTPL)
      COMMON/EDSSCL/K
      COMMON/EDSDAT/WALNAM,KKK,BRN,BRANAM
      INTEGER WALNAM,BRANAM,BRN
      COMMON/EDSSYM/SYMNUM,SYMBOL(NUMSYM,5),TEMP(NUMSYM,3),NEWX,NEWY
     C SYMNAM,WALLST,WTP1
      EQUIVALENCE (WALLST,RWALLT)
      LOGICAL PROXMY
      EPDIS = 2000.0
      DO 100  I = 1,WTP1
      XSCOP1 =(((RWALLT(2,I)-256)/(VTZ*1200.0/2000.0+1))+256)*2
      YSCOP1 =(((RWALLT(3,I)-256)/(VTZ*1200.0/2000.0+1))+256)*2
      XSCOP2 =(((RWALLT(4,I)-256)/(VTZ*1200.0/2000.0+1))+256)*2
      YSCOP2 =(((RWALLT(5,I)-256)/(VTZ*1200.0/2000.0+1))+256)*2
      IC1(1) = XSCOP1
      IC1(2) =YSCOP1
      IC2(1) = XSCOP2
      IC2(2) = YSCOP2
      IF(PROXMY(IX,IY,IC1,IC2)) GO TO 200
  100 CONTINUE
      GO TO 250
  200 KKK = I
      WALNAM = WALLST(1,I)
      IF(RWALLT(2,I).EQ.RWALLT(4,I))GO TO 220
C     TWO Y'S ARE EQUAL
      NEWX = (((IX/2)-256)*(VTZ*1200/2000+1))+256
      NEWY=RWALLT(3,I)
      GO TO 300
```

```
C        TWO X'S ARE EQUAL
220      CONTINUE
         NEWX=RWALLT(2,1)
         NEWY =(((IY/2)-256)*(VTZ*1200/2000+1))+256
         GC TO 300
 250     WALNAM = 'ERROR'
 300     CONTINUE
         RETURN
         END
C****************************************************************************
C**** PROVIDES NAME FOR SYMBOL TYPE --- INCLUDES 4 FIXED CHARACTERS
C             LEFT JUSTIFIED AND 2 NUMERALS RIGHT JUSTIFIED
C             NUMERALS SPECIFY PECULIAR INSTANCE OF SYMBOL TYPE **
         SUBROUTINE ATTNAM
         INCLUDE EDSPRM, LIST
         IMPLICIT INTEGER(A-Z)
         COMMON/EDSSYM/SYMNUM,SYMBOL(NUMSYM,5),TEMP(NUMSYM,3),NEWX,NEWY,
     C   SYMNAM,WALLST,WTP1
         COMMON/EDSNME/TYPE(50,3),MAP(50),PICK
         DEFINE NAME(I) = TYPE(I,1)
         NUMB = MAP(PICK)
         NUMBER = TYPE(NUMB,2)
         TEMP1 = NAME(MAP(PICK))
         CALL ENCODE(KKK)
         WRITE(25,1600),NUMBER
 1600    FORMAT(I2)
         FLD(24,12,TEMP1)= FLD(0,12,KKK)
         SYMNAM = TEMP1
         RETURN
         END
C****************************************************************
         SUBROUTINE LITEBT(LX,LY,RX,UY,CX,CY)
C*******      CX,CY ARE COORDINATES OF TABLET STYLUS
C               CHECKS FOR A HIT BETWEEN :LX-LEFT X,:LY-LOW Y,
C                                        :RX-RIGHT X,:UY-UPPER Y
C               LFLAG SET TO 1  FOR A HIT
         IMPLICIT INTEGER(A-Z)
         COMMON/EDSLBT/LFLAG
         LFLAG = 0
         IF(((CX .GE. LX) .AND. (CX .LE. RX))  .AND.
     C      ((CY .GE. LY) .AND. (CY .LE. UY)))
     C   LFLAG = 1
         RETURN
C****************************************************************************
C  ******  DISPLAYS THE SYMBOLS  FROM 'TEMP'
C               TEMP(KK,3) CONTAINS MASTER SYMBOL TYPE  ***
         SUBROUTINE REGEN
C               REGENERATES THE DISPLAY OF SYMBOLS FROM TEMP
         INCLUDE EDSPRM, LIST
         IMPLICIT INTEGER(A-Z)
```

```
      COMMON/EDSSYM/SYMNUM,SYMBOL(NUMSYM,5),TEMP(NUMSYM,3),NEWX,NEWY,
     C   SYMNAM,WALLST,WTP1
      IF(SYMNUM.EQ. 0) GO TO 110
      KK = 0
 50   KK = KK + 1
      IF(KK .GT. SYMNUM) GO TO 110
      CALL LCHAR
      DO 55  II = 1,38
      IF(TEMP(KK,3) .EQ. II) GO TO (1,2,3,4,5,6,7,8,9,10,11,12,13,
     C              14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,29,
     C              30,31,32,33,34,35,36,37,38),II
 55   CONTINUE
C *******  TWO WIRE          ******
 1    CALL CIRCLE(TEMP(KK,1),TEMP(KK,2))
      CALL TOTEE(TEMP(KK,1)-12,TEMP(KK,2))
      GO TO 100
C *******  WEATHERPROOF      ******
 2    CALL CIRCLE(TEMP(KK,1),TEMP(KK,2))
      CALL TOTEE(TEMP(KK,1)-12,TEMP(KK,2))
      CALL SUB
      CALL WRITAT(TEMP(KK,1)+13,TEMP(KK,2)-12,'WPA')
      GO TO 100
C *******  EXPLOSION PROOF   ******
 3    CALL CIRCLE(TEMP(KK,1),TEMP(KK,2))
      CALL TOTEE(TEMP(KK,1)-12,TEMP(KK,2))
      CALL SUB
      CALL WRITAT(TEMP(KK,1)+13,TEMP(KK,2)-12,'EXA')
      GO TO 100
C *******  GROUNDED          ******
 4    CALL CIRCLE(TEMP(KK,1),TEMP(KK,2))
      CALL TOTEE(TEMP(KK,1)-12,TEMP(KK,2))
      CALL SUB
      CALL WRITAT(TEMP(KK,1)+13,TEMP(KK,2)-12,'GA')
      GO TO 100
C *******  CLOCK             ******
 5    CALL CIRCLE(TEMP(KK,1),TEMP(KK,2))
      CALL TOTEE(TEMP(KK,1)-12,TEMP(KK,2))
      CALL SUB
      CALL WRITAT(TEMP(KK,1)+13,TEMP(KK,2)-12,'CA')
      GO TO 100
C *******  THREE WIRE        ******
 6    CALL CIRCLE(TEMP(KK,1),TEMP(KK,2))
      GO TO 100
C *******  SPECIAL PURPOSE   ******
 7    CALL CIRCLE(TEMP(KK,1),TEMP(KK,2))
      CALL TRI(TEMP(KK,1),TEMP(KK,2)+5)
      GO TO 110
C *******  FLOOR             ******
 8    CALL CIRCLE(TEMP(KK,1),TEMP(KK,2))
      CALL DOT(TEMP(KK,1),TEMP(KK,2)+4)
```

```
      GO TO 100
C ******* SINGLE POLE        ******
 9      CALL TEE(TEMP(KK,1)-10,TEMP(KK,2))
        CALL WRITAT(TEMP(KK,1)+1,TEMP(KK,2)-6,'SΔ')
        GO TO 100
C ******* DOUBLE POLE        ******
 10     CALL TEE(TEMP(KK,1)-10,TEMP(KK,2))
        CALL WRITAT(TEMP(KK,1)+1,TEMP(KK,2)-6,'SΔ')
        CALL SUB
        CALL WRITAT(TEMP(KK,1)+13,TEMP(KK,2)-13,'2Δ')
        GO TO 100
C ******* THREE WAY          ******
 11     CALL TEE(TEMP(KK,1)-10,TEMP(KK,2))
        CALL WRITAT(TEMP(KK,1)+1,TEMP(KK,2)-6,'SΔ')
        CALL SUB
        CALL WRITAT(TEMP(KK,1)+13,TEMP(KK,2)-13,'3Δ')
        GO TO 100
C ******* FOUR WAY           ******
 12     CALL TEE(TEMP(KK,1)-10,TEMP(KK,2))
        CALL WRITAT(TEMP(KK,1)+1,TEMP(KK,2)-6,'SΔ')
        CALL SUB
        CALL WRITAT(TEMP(KK,1)+13,TEMP(KK,2)-13,'4Δ')
        GO TO 100
C ******* WEATHER PROOF      ******
 13     CALL TEE(TEMP(KK,1)-10,TEMP(KK,2))
        CALL WRITAT(TEMP(KK,1)+1,TEMP(KK,2)-6,'SΔ')
        CALL SUB
        CALL WRITAT(TEMP(KK,1)+13,TEMP(KK,2)-13,'WPΔ')
        GO TO 100
C ******* PILOT LIGHT        ******
 14     CALL TEE(TEMP(KK,1)-10,TEMP(KK,2))
        CALL WRITAT(TEMP(KK,1)+1,TEMP(KK,2)-6,'SΔ')
        CALL SUB
        CALL WRITAT(TEMP(KK,1)+13,TEMP(KK,2)-13,'PΔ')
        GO TO 100
C ******* TWO SPEED          ******
 15     CALL TEE(TEMP(KK,1)-10,TEMP(KK,2))
        CALL WRITAT(TEMP(KK,1)+1,TEMP(KK,2)-6,'SΔ')
        CALL SUB
        CALL WRITAT(TEMP(KK,1)+13,TEMP(KK,2)-13,'2SPΔ')
        GO TO 100
C ******* THREE SPEED        ******
 16     CALL TEE(TEMP(KK,1)-10,TEMP(KK,2))
        CALL WRITAT(TEMP(KK,1)+1,TEMP(KK,2)-6,'SΔ')
        CALL SUB
        CALL WRITAT(TEMP(KK,1)+13,TEMP(KK,2)-13,'3SPΔ')
        GO TO 100
C ******* AUTO DOOR          ******
 17     CALL TEE(TEMP(KK,1)-10,TEMP(KK,2))
        CALL WRITAT(TEMP(KK,1)+1,TEMP(KK,2)-6,'SΔ')
```

```
      CALL SUB
      CALL WRITAT(TEMP(KK,1)+13,TEMP(KK,2)-13,'DΔ')
      GO TO 100
C *******  W/CONV. OUTLET      ******
 18   CALL WRITAT(TEMP(KK,1)+1,TEMP(KK,2)-6,'SΔ')
      GO TO 100
C *******  EXPLOSION PROOF    ******
 19   CALL TEE(TEMP(KK,1)-10,TEMP(KK,2))
      CALL WRITAT(TEMP(KK,1)+1,TEMP(KK,2)-6,'SΔ')
      CALL SUB
      CALL WRITAT(TEMP(KK,1)+13,TEMP(KK,2)-13,'EXΔ')
      GO TO 100
C *******  KEY OPERATED       ******
 20   CALL TEE(TEMP(KK,1)-10,TEMP(KK,2))
      CALL WRITAT(TEMP(KK,1)+1,TEMP(KK,2)-6,'SΔ')
      CALL SUB
      CALL WRITAT(TEMP(KK,1)+13,TEMP(KK,2)-13,'KΔ')
      GO TO 100
C *******  PHOTOCELL          ******
 21   CALL TEE(TEMP(KK,1)-10,TEMP(KK,2))
      CALL WRITAT(TEMP(KK,1)+1,TEMP(KK,2)-6,'SΔ')
      CALL SUB
      CALL WRITAT(TEMP(KK,1)+13,TEMP(KK,2)-18,'OΔ')
      GO TO 100
C *******  GENERATOR          ******
 22   CALL CIRCLE(TEMP(KK,1),TEMP(KK,2))
      CALL WRITAT(TEMP(KK,1)-5,TEMP(KK,2)-6,'GΔ')
      GO TO 100
C *******  MOTOR              ******
 23   CALL CIRCLE(TEMP(KK,1),TEMP(KK,2))
      CALL WRITAT(TEMP(KK,1)-5,TEMP(KK,2)-6,'MΔ')
C *******  METER              ******
 24   CALL CIRCLE(TEMP(KK,1),TEMP(KK,2))
      CALL BOX(20,20,TEMP(KK,1)-10,TEMP(KK,2)-10)
      CALL WRITAT(TEMP(KK,1)-5,TEMP(KK,2)-6,'MΔ')
      GO TO 100
C *******  THERMOSTAT         ******
 25   CALL BOX(16,16,TEMP(KK,1)-8,TEMP(KK,2)-8)
      CALL WRITAT(TEMP(KK,1)-5,TEMP(KK,2)-6,'TΔ')
      CALL SMTEE(TEMP(KK,1)-8,TEMP(KK,2))
      GO TO 100
C *******  ANTICIPATOR        ******
 26   CALL BOX(16,16,TEMP(KK,1)-8,TEMP(KK,2)-8)
      CALL WRITAT(TEMP(KK,1)-5,TEMP(KK,2)-6,'TΔ')
      CALL SMTEE(TEMP(KK,1)-8,TEMP(KK,2))
      GO TO 100
C *******  PANEL              ******
 27   CALL BOX(20,10,TEMP(KK,1)-10,TEMP(KK,2)-5)
      GO TO 100
C *******  CONTROL  PANEL     ******
```

```
 28      CALL BOX(20,10,TEMP(KK,1)-10,TEMP(KK,2)-5)
         CALL POS(TEMP(KK,1)-10,TEMP(KK,2)-5)
         CALL VEC(TEMP(KK,1)+10,TEMP(KK,2)+5)
         CALL POS(TEMP(KK,1)+10,TEMP(KK,2)-5)
         CALL VEC(TEMP(KK,1)-10,TEMP(KK,2)+5)
         GO TO 100
C *******  DISCONNECT          ******
 29      CALL BOX(15,8,TEMP(KK,1)-7,TEMP(KK,2)-4)
         GO TO 100
C *******  TRANSFORMER         ******
 30      CALL BOX(20,10,TEMP(KK,1)-10,TEMP(KK,2)-5)
         GO TO 100
C *******  SURFACE FLOURESCENT******
 31      CALL BOX(40,20,TEMP(KK,1)-20,TEMP(KK,2)-10)
         CALL CIRCLE(TEMP(KK,1)-10,TEMP(KK,2))
         GO TO 100
C *******RECESSED FLOURESCENT *********
 32     CALL BOX(40,20,TEMP(KK,1)-20,TEMP(KK,2)-10)
         CALL CIRCLE(TEMP(KK,1)-10,TEMP(KK,2))
         GO TO 100
C *******RESURFACE INCANDESCENT *******
 33      CALL CIRCLE(TEMP(KK,1),TEMP(KK,2))
         CALL SUB
         CALL WRITAT(TEMP(KK,1)+13,TEMP(KK,2)-12,'MA')
         GO TO 100
C *******RERECESSED INCANDESCENT*******
 34      CALL CIRCLE(TEMP(KK,1),TEMP(KK,2))
         CALL SUB
         CALL WRITAT(TEMP(KK,1)+13,TEMP(KK,2)-12,'NA')
         GO TO 100
C *********PULL SWITCH         *************
 35      CALL CIRCLE(TEMP(KK,1),TEMP(KK,2))
         CALL SUB
         CALL WRITAT(TEMP(KK,1)+13,TEMP(KK,2)-12,'PSA')
         GO TO 100
C********  SURFACE INCANDESCENT  **********
 36      CALL CIRCLE(TEMP(KK,1),TEMP(KK,2))
         GO TO 100
C********  RECESSED INCANDESCENT  *********
 37      CALL CIRCLE(TEMP(KK,1),TEMP(KK,2))
         GO TO 100
C********  PULL SWITCH          T  *********
 38      CALL CIRCLE(TEMP(KK,1),TEMP(KK,2))
         CALL SUB
         CALL WRITAT(TEMP(KK,1)+13,TEMP(KK,2)-12,'PSA')
 100     GO TO 50
 110     CONTINUE
         CALL LCHAR
         RETURN
         END
```

```
C*****************************************************************************
C*****************************************************
C***** DISP AYS APPROIATE SCALE MESSAGE ***
         SU ROUTINE MESAGE
         IMPLICIT INTEGER(A-Z)
         INCLUDE EDSPRM, LIST
         COMMON/EDSSCL/K
         COMMON/EDSSYM/SYMNUM,SYMBOL(NUMSYM,5),TEMP(NUMSYM,3),NEWX,NEWY,
     C   SYMNAM,WALLST,WTP1
         CALL INTEN(0)
         IF(K .NE. 4) GO TO 20
         CALL WRITAT(300, 80,'SCALE:1/4 INCH= 1 FOOTΔ')
         CALL WRITAT(300,60,'SPACING BETWEEN GRIDS EQUALS 16 INCHESΔ')
         GO TO 40
 20      IF(K .NE. 8) GO TO 30
         CALL WRITAT(300, 80,'SCALE:1/8 INCH= 1 FOOTΔ')
         CALL WRITAT(300,60,'SPACING BETWEEN GRIDS EQUALS 32 INCHESΔ')
         GO TO 40
 30      CALL WRITAT(300, 80,'SCALE:1/16 INCH = 1 FOOTΔ')
         CALL WRITAT(300,60,'SPACING BETWEEN GRIDS EQUALS 64 INCHESΔ')
 40      CONTINUE
         CALL INTEN(2)
         RETURN
         END
C*****************************************************************************
C***** CHECKS STYLUS PROXIMITY TO A WALL ***
C       ELLIPSE IS USED --- LONG AXIS ALONG WALL, SHORT PERPENDICULAR
         LOGICAL FUNCTION PROXMY(IXP,IYP,IC1,IC2)
C        TEST FOR POINT IN PROXIMITY WITH A LINE
C
         INTEGER IXP,IYP,IC1(2),IC2(2)
C
      DIMENSION C1(2),C2(2)
      V=4.
         EPLISN=10.0
         PROXMY=.FALSE.
C
         XP=IXP
         YP=IYP
         C1(1)=IC1(1)
         C1(2)=IC1(2)
         C2(1)=IC2(1)
         C2(2)=IC2(2)
C
         IF(ABS(C1(1)-C2(1)) .LT.1.) GO TO 200
         ATN=ATAN((C2(2)-C1(2))/(C2(1)-C1(1)))
 50      ASQRD=((C2(1)-C1(1))/2.0)**2 + ((C2(2)-C1(2))/2.0)**2
      IF (ASQRD.LE.0) GO TO 310
         SNATN=SIN(ATN)
         CSATN=COS(ATN)
```

```
      AVGX=(C2(1)+C1(1))/2.0
      AVGY=(C2(2)+C1(2))/2.0
      ELPS=(((XP-AVGX)*CSATN+(YP-AVGY)*SNATN)**2)/ASQRD  +
     *      Y*(((YP-AVGY)*CSATN+(XP-AVGX)*SNATN)**2)/ASQRD
      IF(ELPS.LE.1.0)PROXMY=.TRUE.              @POINT INSIDE ELLIPSE
      RETURN
C
C     LINE IS ZERO LENGTH
C
100   IF(ABS(C1(1)-XP).LE.EPLISN)PROXMY=.TRUE.
      RETURN
C
200   ATN=6.28/4.0
      GO TO 50
      END
C*****************************************************************************
C*****  DETERMINES WHETHER  ATTRIBUTE VALUES ARE TO BE ENTERED **
C***** DETERMINES WHICH ATTRIBUTE CATERGORY WAS SELECTED BY STYLUS**
C*****  TYPIN CAPTURES INPUT AND STORES IT IN LOCATION OR ARRAY**
C******  THESE VALUES ARE PASSED TO 'ATTRIB' THRU COMMON 'EDSATB'
      SUBROUTINE INPUT($,$)
      IMPLICIT INTEGER(B-Z)
      INCLUDE DSPRMS,LIST
      INCLUDE EDSPRM, LIST
      INTEGER AX,AY,AZ
      INTEGER ATT
      REAL A
      COMMON/EDSLBT/LFLAG
      COMMON/EDSNME/TYPE(50,3),MAP(50),PICK
      COMMON/EDSATB/V,U,M(10),T(10),D(10),H,C(10),A,ICHAR,ATT
      COMMON/EDSSYM/SYMNUM, SYMBOL(NUMSYM,5),TEMP(NUMSYM,3),NEWX,NEWY,
     C   SYMNAM,WALLST,WTP1
      INTEGER DFV(DFVL),DFVP,L VF
      INTEGER DFB(DFBL,WPN)
      COMMON /DF/DFV,DFB,DFVP,
      NAMELIST/ANSWER/YES,NO
      DEFINE C1(I) = C(I)
      DEFINE  C2(I) = FLD(ABS(MOD(I-1,6))*6,6,C1(I-1/6+1))
      DEFINE D1(I) = D(I)
      DEFINE  D2(I) = FLD(ABS(MOD(I-1,6))*6,6,D1(I-1/6+1))
      DEFINE M1(I) = M(I)
      DEFINE  M2(I) = FLD(ABS(MOD(I-1,6))*6,6,M1(I-1/6+1))
      DEFINE TT(I) = T(I)
      DEFINE TT   = FLD(ABS(MOD(I-1,6))*6,6,TT(I-1/6+1))
      CALL SETL
      READ(5,ANSWER)
      CALL JUMPS('ANSWER', 500,$1030)
      CALL TYPOUT('DO YOU WISH TO NAME A NEW INSTANCE?@A')
      CALL TYPOUT('YES OR NO@A
      CALL CHRINT(1,$1010)
```

```
1000    CALL IDLE
        CALL SWAP
        GO TO 1000
1010    CALL TTY
        GO TO 1000
1500    NUM = MAP(PICK)
        TYPE(NUM,2) = TYPE(NUM,2) + 5
        CALL ATTNAM
        CALL ATTRIB
        CALL SENDF
1050    CALL TABINT(1,$1460)
        CALL CHRINT(1,$1455)
1060    CONTINUE
       CALL TYPOUT('SELECT ATTRIBUTE CATEGORY¤Δ')
1450    CALL IDLE
        CALL SWAP
        GO TO 1450
1455    CALL TTY
        GO TO 1450
1460    CALL GETTAB(AX,AY,AZ)
        IF(AZ) 1450,1450,1470
1470    CALL LITEBT(25,180,200,230,AX,AY)
        IF(LFLAG .EQ. 1) GO TO 1490
        ATT = 1
        DO 1475  LB = 616,392,-32
        LY = LB - 16
        UPY = LB + 16
        CALL LITEBT(40,LY,450,UPY,AX,AY)
        IF(LFLAG .EQ. 1) GO TO 1480
        ATT = ATT + 1
1475    CONTINUE
        GO TO 1498
1490    CALL TYPATT
        RETURN 1
1480    CALL TYPOUT('PLEASE TYPE IN¤Δ')
        GO TO(100,200,300,400,500,600,700,800),ATT
100     CALL TYPIN('(I)',V)
        GO TO 2000
200     CALL TYPIN('(I)',U)
        GO TO 2000
300     CALL TYPIN(M)
        ICHAR = NTYPIN(IDUM)
        DO 310  J = ICHAR,30
        M2(J) = ' '
310     CONTINUE
        GO TO 2000
400     CALL TYPIN(T)
        ICHAR = NTYPIN(IDUM)
        DO 410  J = ICHAR,30
        TIT(J) = ' '
```

```
410     CONTINUE
        GO TO 2000
500     CALL TYPIN(D)
        ICHAR = NTYPIN(IDUM)
        DO 510  J = ICHAR,30
        D2(J) = ' '
510     CONTINUE
        GO TO 2000
600     CALL TYPIN('(I)',H)
        GO TO 2000
700     CALL TYPIN(C)
        ICHAR = NTYPIN(IDUM)
        DO 710  J = ICHAR,30
        C2(J) = ' '
710     CONTINUE
        GO TO 2000
800     CALL TYPIN('(R)',A)
        A = A/100.0
2000    CALL SETDF(DFV,DFVP)
        CALL ATTRIB
        CALL SENDF
        GO TO 1060
1498    CALL SETDF(DF,I)
        CALL WRITAT(225,30,'NO ATTRIBUTE WAS FOUNDΔ')
        CALL WRITAT(225,5,'PLEASE TRY AGAINΔ')
        RETURN 2
        END
C**************************************************************************
C**************************************************************************
```

DATA CARDS IGNORED - FIRST IS LISTED BELOW