ENERPAC: AN INTERACTIVE PROGRAM FOR ESTIMATING BUILDING ENERGY LOADS

by

Harry Frank Moate, III

Thesis submitted to the Graduate Faculty of the
Virginia Polytechnic Institute and State University
in partial fulfillment of the requirements for the degree of

MASTER OF ARCHITECTURE

in

Environmental Systems Studies

APPROVED:

Robert N. S. Chiang, Chairman

Gar Day Ding

Jack R. Warner

March, 1972

Blacksburg, Virginia

ACKNOWLEDGEMENTS

I would first like to thank my family for the support and encouragement which they have given during my return to academic life.

In addition, I would like to thank the members of my graduate committee, G. Day Ding, Jack R. Warner, and Robert N. S. Chiang, for both their help with this thesis and for the assistance provided on previous projects.

A final thanks must go to Miss Carol Kirk for her help in typing the manuscript.

TABLE OF CONTENTS

	Page
Acknowledgements	ii
Table of Contents	iii
List of Illustrations	iv
CHAPTER I - Introduction	1
CHAPTER II - A Review of Existing Energy Load Programs	9
CHAPTER III - Requirements for an Energy Load Calculation Program for Architects	18
CHAPTER IV - Methodology for ENERPAC	28
CHAPTER V - Analysis of the Results	48
CHAPTER VI - Conclusions	54
Listed References	62
Selected Bibliography	64
Appendices	
I - User's Manual	68
II - Program Listing for ENERPAC	88
Vita	107

LIST OF ILLUSTRATIONS

Figure		Page
1	Energy Consumption in the United States, Past, Present and Future	2
2	Average Cost per Year for Various Fuels	6
3	ENERPAC Flow Diagram	33
4	Flow Diagram for Subroutine INPUT/QUEST 2	35
5	Typical INPUT Summary	49
6	Typical Results from Program ENERPAC	50

CHAPTER I

INTRODUCTION

> Let us use the letter Q to stand for the energy derived from burning some 38 billion tons of coal. In the eighteen and one half centuries after Christ, the total energy consumed averaged less than one half Q per century. This means, roughly speaking, that half of all the energy consumed by man in the past 2000 years has been consumed in the last one hundred.[1]

This quotation from Dr. Homi Bhabha, Chairman of the First Conference on the Peaceful Uses of Atomic Energy, indicates, quite dramatically, the tremendous increase in energy consumption which has occurred during the past several years. The future is even more ominous:

> Extrapolating from current trends, the need for energy in the next century will be between 100 and 400 Q.[2]

In the United States, the increase in energy usage is even more accelerated. John W. Simpson, President of the Power Systems Division of Westinghouse Electric Corporation, gave the following facts in a recent speech: "Between 1960 and 1969, the average household consumption of electric power rose from 3851 to 6550 kilowatt hours, up 70%. By 1985, it has been estimated that average annual consumption will be 17,240 kilowatt hours, about 4 1/2 times that of 1960."[3] "Between 1960 and 1970, industries increased their use of power by 70%. Yet in the decade from last year to 1980, industries will double their use of power."[4] Figure 1, from a different source, shows a similar prediction of the magnitude of this acceleration of energy consumption in the United States.

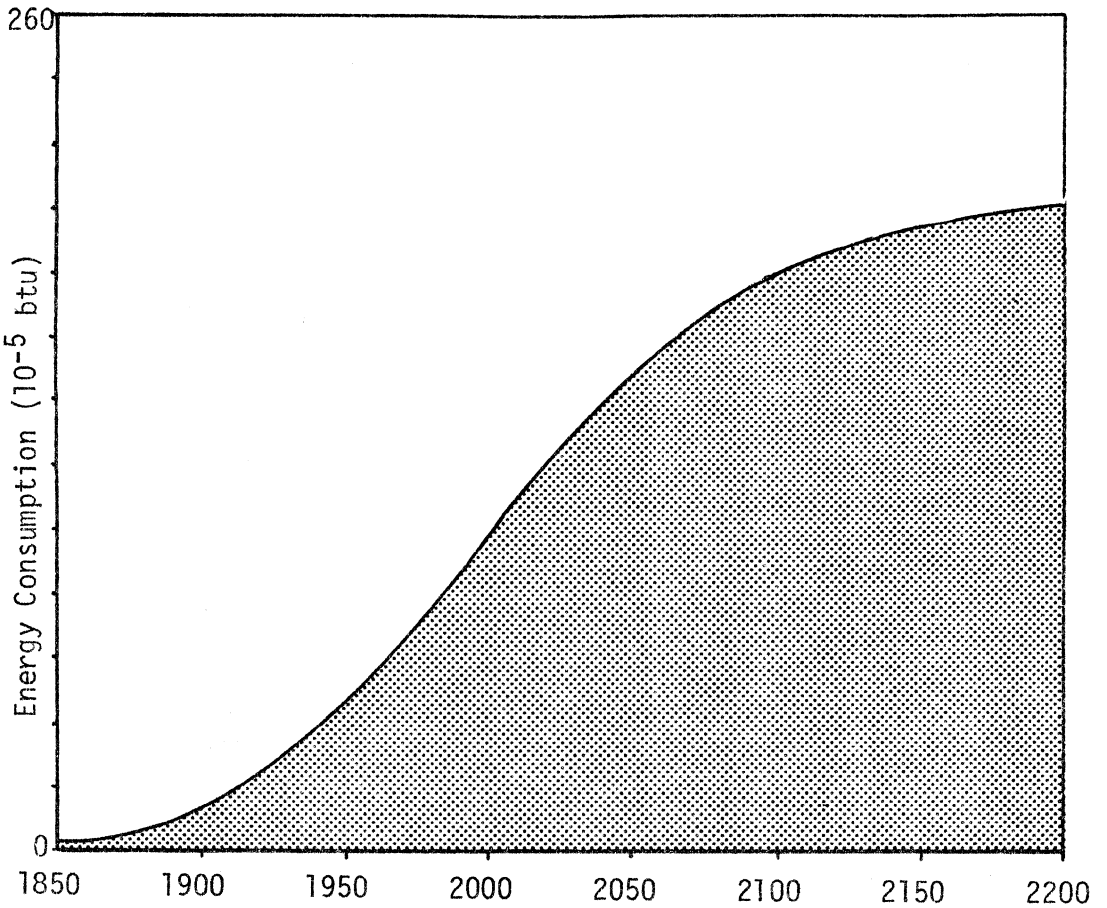


Figure 1. Energy consumption in the United States, past, present, and future.[5]

The real problem is that this increase is occurring at a time when the factors influencing the energy supply are restricting its expansion. Only a few years ago, nuclear energy was hailed as the answer to all of the questions concerning the energy supply. Yet nuclear power plants have encountered severe engineering and cost overrun problems during construction. Most are now far behind schedule.

Because of the federal restrictions on the selling price of

interstate natural gas, there has been a decrease in the exploration for new sources. Consequently, shortages are now occurring as demand increases. Liquified natural gas is being brought in from foreign countries by tanker to alleviate this shortage, but the cost is significantly higher.

Much of the oil for the United States comes from the Middle East. The oil producing nations in that area have banded together to extract the maximum royalty from those nations which must purchase oil. At every chance, such as the recent floating of the value of the dollar, they have demanded and received an increase in prices. The discovery of oil on the North Slope of Alaska caused quite a stir, but the cost of this oil will be high. It must be extracted from shale, which will be expensive. In addition, the engineering problems associated with the Alaskan climate and the protection of the environment will also increase costs.

Coal too is expected to have a continued sharp increase in price. When the prospects for nuclear power were being promoted, many coal companies delayed the opening of new mines. Now, with the failure of nuclear energy to produce as expected, and the increased use of coal by foreign countries coal prices have risen sharply. Another factor which will surely effect the cost of coal is the implementation of proposed mine safety regulations. At this time, they are essentially unenforced. When they do become enforced, the cost must again rise.

While each of these industries has its unique problems, there is one problem common to all, the new emphasis on the protection of the environment. New regulations, now being enacted, are almost certain to be a particular problem to the energy supply industries. They must find ways of producing power while lessening their impact on the environment. Electric plants could be very hard hit if required to reduce their emissions into both air and water to proposed levels. Nuclear plants have a considerable problem to overcome with thermal pollution. A good portion of the coal and oil industries output is of high sulfur content, making it unusable under clean air standards. To solve these respective problems, each industry is going to have to spend a great deal of money. As with the factors discussed earlier, this can only push prices for each of these fuels higher.

In a recent issue of Forbes[6] surveying the energy situation, this trend was reflected. The article pointed out that the price of coal has doubled over the past three years. The price of residual fuel oil in some areas has risen 60% in the past year alone. David Freeman, head of the energy section of the President's Office of Science and Technology, expects the cost of electricity to rise 30% in the next decade. John F. O'Leary, former head of the Bureau of Mines, said: "Over the next 15 years, we can expect a doubling of the real price of natural gas to the consumer."[7] Others make predictions more drastic than these, and when they are compared with

recent price trends, as shown in Figure 2, the more drastic predictions seem inevitable.

What impact does this have on building design? The concluding sentence from the Forbes report is: "We've lived high on our low cost raw materials, now we must reach into our pockets and at last pay the price for our style of living."[9] Thus, an item which previously had little effect on the design process is going to become increasingly important. As the cost of supplying energy for a building over its life rises, relative to the first cost of the building, owners are going to become increasingly insistant that full consideration be given these costs during building design. The increasing sophistication of owners, as well as the growing practice of hiring professional owner's representatives to review designs should further this trend.

Too many times in the past, the architect has failed to give sufficient consideration to energy conservation, primarily because of the relatively low cost of fuel. The building was simply given to the consulting engineer after completion of the design for "whatever it takes" to meet the energy requirements. Yet, as annual energy costs for buildings escalate, owners are going to demand that a design must incorporate concepts to reduce these energy costs. Consequently, the architect is going to need a method of evaluating these concepts and alternatives in the early stages of design.

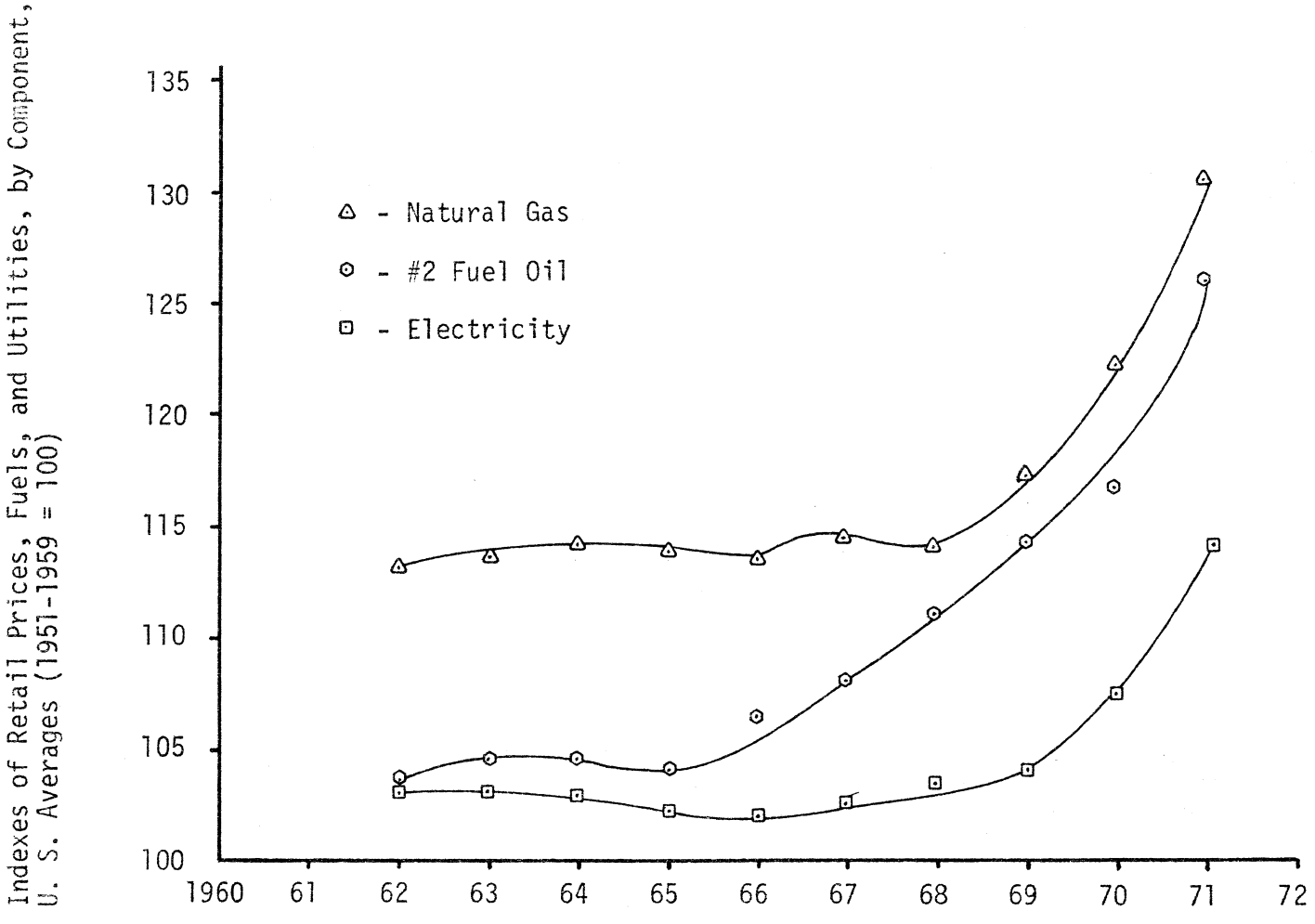


Figure 2. Average Cost per Year for Various Fuels[8]

Since this methodology must be used by the architect rather than the engineer, it will be different than that which is generally in use today. It must be structured in a way which removes the necessity for technical expertise and permits the architect to use his own terminology and set of variables. Because its use will be required before many of the details of the building are known, it should utilize the elements and scale appropriate to this level of design. It should also be simple to use and should not consume time which the designer needs to devote to other elements of the project.

The computer has been used with great success in solving problems of this type. There are several reasons for this, but the primary one is that the computer can store both large amounts of data and also the procedures for calculations with this data. The data can then be retrieved and calculations made in response to simple commands from a computer program. A system which uses this concept successfully for a similar problem is the ICES (Integrated Civil Engineering System) and one of its subsystems, STRUDL (Structural Design Language).[9] This system uses a library of procedures. The designer inputs a list of simple commands or operations to be executed, along with the data required. The program then processes these commands in sequence, recalling from the library all of the detailed algorithms required to process that particular command. A great amount of flexibility is available with this program since new procedures can be added to the library, as developed, for recall at any time.

This thesis presents a computer program for energy analysis by architects which has been developed to meet these requirements. It also documents the methodology used in the program. The five basic areas covered by the thesis are: (1) A review of existing programs, (2) New program requirements, (3) Methodology, (4) Results, (5) Conclusions.

CHAPTER II

A REVIEW OF EXISTING ENERGY LOAD PROGRAMS

Most of the existing programs are intended more for analysis than for design. However, by examining them, it is possible to determine the characteristics which a new program should have. The programs have generally come from four different sources: (1) utility companies, (2) universities, (3) private consulting and computer software firms, and (4) professional societies. Since programs from each of these sources are quite similar, it is easier to examine each group than all of the individual programs.

The programs developed by the utility companies are inadequate for several reasons. The most prominent is the lack of confidence by users in the results of these programs. Of course, this is easy to understand when, as has been reported, representatives of two different energy sources deliver results which "prove" that their source is the most economical for a particular project. This situation is compounded by the fact that since the programs were developed on a proprietory basis, the user is unable to examine the program logic and data. As with most of the existing programs, which were developed under the process used by architects and engineers in the past, these programs require input data that is much too detailed for use in design. Another disadvantage is that, because of the typical role of sales engineer for the company, it is frequently necessary to have another person between the user and computer which

further increases the turn-around time for the program. Consequently, it becomes unacceptable as a design tool. It appears that the primary appeal of these programs is to the small engineering firm without computer capability. If they are simply adding the necessary heating and cooling to a building which is already designed, and providing there is sufficient time, they can significantly reduce their office overhead costs by letting the utility company do the work of computing the loads.

University-developed energy load programs run the complete spectrum of sophistication. One of the most successful and sophisticated examples is the HEATRAN-2[11] program developed by the Architectural Engineering Department at Penn State University. It is actually a group of programs, developed under sponsorship of the National Science Foundation, for building energy design and evaluation. The basis for the program is the technique of Monte Carlo Simulation of Solar Radiation and Dry Bulb Temperatures for Air Conditioning Processes.[12] The HEATRAN-2 program is written in Fortran and requires a program deck obtained from the University. The basic operation is directed by a set of commands which process data with internal routines stored in the machine. Data generally consists of 40 cards, some of which might contain as many as 12 values, obtained from the Users Guide and transferred to the cards. Depending on the commands used, the output can consist of such items as monthly temperature conditions, heating/cooling loads for up to three wall configurations, maximum heat gain and loss, or a profile of this data.

The options currently available are:

1. Weather option - Provides yearly summaries of the weather data and is used to assure that proper data is being generated.
2. Air conditioning option - This option causes the main program to be executed, and, together with the command cards, determines the type of results which will be produced.
3. Output option - Currently this provides only an output of the solar absorption and transmission through glass for all incident angles between 0 and 90 degrees.

The basic approach of this program to the evaluation of the solar and transmission loads for walls, roofs, and glass comes closer to satisfying the criteria for a design program than almost any other available. In this area, its only disadvantage is dependence on the turn-around time for the computer installation being used. If an in-house computer is not available, this program becomes too cumbersome to use. If the turn-around time is a day or more, it is inconvenient, and thus may be used sparingly. With a fast turn around time, it is quite useful. But while it is quite useful for the evaluation of the material for these three elements, it does not calculate the actual air conditioning load of a building. It simply calculates the transmission load, based on the sol-air temperature, through the materials. It does not include ventilation load, infiltration, lighting, appliance, and occupancy load. Thus, its use

is restricted to one area of evaluation. It does, however, provide some very good techniques which might be adaptable to other programs.

Many of the consulting engineering firms who have acquired computer capability have also developed programs for energy analysis to use on these machines. The degree of sophistication of these programs varies greatly. Some firms have simply computerized their manual calculations. In one recent article,[13] the primary concern in their approach seemed to be the presentation of all output data on one page. Other firms, such as Caudill, Rowlett, and Scott have taken a more in-depth look at the whole process.[14] Understandably, their programs appear to provide easier input, and yet more accurate results. As did C.R.S., most of these firms have taken a deterministic approach in the generation of weather data simply because they feel that they cannot justify the increase in accuracy in terms of the added data collection, programming, and program execution costs. Despite the apparent success of some of these programs for the engineer, they suffer from the same problems of relatively slow turn-around time and the need for detailed information. This really makes no difference to the architect, however, since they are intended for in-house use and are unavailable to others.

Once these programs are developed and if the firm has sufficient hardware capacity, an increasing trend is for the large firms to spin-off a computer service company and sell the service to those who do not have computer capability. Two such firms and their respective systems are Herman Blum Consulting Engineers, Dallas, Texas - APACE[15]

and Syska and Hennessy Information Systems, New York - SHIS.[16] As with the utility company programs, clients send their data to the company for processing. Some of the advantages claimed for this mode of operation include:[17] (1) the client's personnel do not need to learn about machine operation or about the particular mechanics of using a program, other than its engineering aspects, (2) there are no fixed costs, as there might be with terminal operation, (3) liaison engineers have an opportunity to review submitted work and insure desired quality, (4) the client has constant access to the liaison engineer to have questions answered and to be sure that the program is fully exploited. These advantages are true, but as with the utility company programs, the time lag necessary for the transmittal of information makes this type of program acceptable for analysis but unacceptable for design.

Because of the expected potential, some of the computer hardware and software firms such as Honeywell Information Systems, Dyna Data Services, Westinghouse, and McDonnald Douglas Automation Company have also entered this area. One group of these firms operates much the same as the consulting engineers mentioned above, with the same advantages and disadvantages, although they might be able to offer slightly lower computer time charges because of their more efficient utilization and larger machines. There is some question as to whether they give the same attention to the quality of the results since they are not specialized in this area. Some claim that they simply give results based on the input data supplied by the user, right or wrong.

The other group of these firms operates a central processing facility with appropriate programs but differ in that they use an interactive terminal for transmission of information between the user and computer. Some of the reasons this method is claimed to be superior are as follows:[18]

1. Computer experts agree that if the problem to be solved has a relatively small amount of input and output data but the memory requirements are large and the program relatively complex, time sharing is a valid approach.
2. Time sharing is the correct technique if the user is looking for immediate response and the ability to interchange data.
3. If program is structured right, a great deal of computer knowledge by user can be omitted.
4. Time sharing reduces turn-around time significantly.
5. Time sharing allows correction of errors at the time they are made. Time delay for reruns is omitted.
6. Programs may be sectionalized and only part of the program run to avoid greater expense while reducing turn-around time.
7. Access is provided to a large machine rather than the IBM 1130's normally used by many small offices.
8. Time sharing costs are based on terminal usage. If programs exist, there are no programming costs or first costs for computers.

Some of the disadvantages include:

1. Programs must be structured for efficient input/output operation.
2. There is no opportunity for independent review of results. Keypunch errors can go undetected unless proper scrutiny is given by the user.

Despite these disadvantages, the on-line terminal appears to be the best mode of operation for design, primarily because it almost eliminates turn-around time, a drawback with almost all of the existing programs. The other major problem, too much detail required for input data, can then be solved through proper programming. In fact, this reduction of input data makes terminal operation even more efficient.

The fourth source of programs has been professional societies. There are two societies active in this area, ASHRAE and APEC. Their expressed goals have resulted in two very different approaches. ASHRAE has defined its role as the development of algorithms for calculations but not development of the programs themselves. They feel that this leaves the level of sophistication to which the program is developed to the user. APEC (Automated Procedures for Engineering Consultants) was formed to develop actual programs for consulting engineers and others engaged in the area of building design. Member firms pay dues for the use of these programs and to support the development of new programs.

The algorithms developed by ASHRAE represent an important contribution to research in this area because they employ some advanced concepts. Also important is that they have collected all of this information into one volume. However, the transfer of many of the techniques into actual practice may not come easy for the average office. The mathematical techniques are rather complex. Most of the smaller offices do not, unfortunately, have people experienced in Fourier Analysis. In addition, the procedures feature the use of Environmental Science Services Administration weather tapes. Again, this is too involved for general use and certainly too involved for use during the design phase of a project.

APEC programs do not suffer from these problems since they were specifically developed for use by consultants. But since the consultant normally does not make his energy load calculations until late in the project, this program has the same limitation as other programs already reviewed. In discussing the APEC programs with representatives of two firms, it appears that the pattern is to subscribe to the service and to use the central processing facility until computer capability is acquired. After use of a computer is obtained the programs are transferred to the users machine. Within three to four years, as a firm develops its own specific requirements, a new and more detailed program is written by the firm to satisfy those requirements.

It can be seen from this review of existing energy load programs that most are intended for use by the consulting engineer after the

architectural design of the building is complete. This, of course, is understandable since this has been the process which has been followed in the past. But with the increase in energy prices which has occurred in recent years, as shown in Figure 2, it is evident that this process will have to be modified. More programs will have to be developed which allow an analysis of the factors affecting the energy consumption during the architectural design. What is necessary is a program which allows annual energy consumption to be one of the determinants of architectural design rather than one of the consequences.

CHAPTER III

REQUIREMENTS FOR AN ENERGY LOAD CALCULATION PROGRAM FOR ARCHITECTS

Having examined the strengths and weaknesses of the various groups of existing programs, it is now possible to establish the requirements for a program which better fits the needs of the architect. The first step is to understand the types of energy consumption which occur within a building. There are several ways of classifying this utilization, one way is to categorize it by the type of load on the final energy conversion device. Using this system, the total load can be subdivided into four components:

1. Air Conditioning - all loads resulting from the air conditioning equipment. This can range from fuel oil for heating boilers to electricity for cooling tower fan motors.
2. Lighting - loads related to the building lighting system, both interior and exterior.
3. Process - loads resulting from processes occurring within a building which are required to produce the items or perform the service for which the space is being used. A drill press in a factory or a stock quotation board in a broker's office would both be included in this category.
4. Overhead - loads related to the miscellaneous equipment for indirect support of the process for which the space is being used. It generally includes the various "plug-in"

devices, such as office machines, used within a building. When considering these four types of loads from the architect's point of view, it is possible to categorize them in another way. The process load and overhead load are fixed loads. They are dictated by the functions which are to occur within the space. The architect has no control over them. The other two types of loads, air conditioning and lighting are variable loads to the architect because they are the ones over which he has control as a result of his design decisions. It is for these loads that a program for energy design is required. The process and overhead loads are considered only when they influence the other two. As a result of this, the primary orientation of the program is toward the thermal environment of the building.

Having fixed the basic program orientation, it is possible to further examine the needs of the architect to determine the requirements and then the final structure for such a program. It should be realized from the outset that what the architect requires is not the ability to make a precise calculation of the energy load for each room in a building, but rather the ability to make a realistic comparison between various alternatives which influence his design. This could be a comparison between different building configurations, such as shape or orientation, or between different types of materials, such as reflective or non-reflective glass. If the energy consumption were to be the primary determinant in the architectural design process, more precise calculations might be necessary. But since energy consumption remains as only one of several criteria for evaluation

of the design, sufficient accuracy for a valid comparison of alternatives is justified.

Since the calculation of energy loads have traditionally been performed by the engineer, most architects are not experienced in this aspect of design. Some schools of architecture attempt to give them a basic knowledge of the techniques, but apparently, after working for a few years, these techniques are forgotten. As a consequence, it is necessary to devise a process which can be performed without a specific knowledge of the techniques involved. Several methods are available, one being the use of a program handbook which tells the user what data is required and how to punch it on a computer card deck. Such a method is relatively cumbersome and time consuming. As a result, it does not readily lend itself to the pace of decision making which may be required for building design if a great number of possible combinations are to be evaluated.

The availability of the on-line computer terminal solves this problem because through its use, answers are produced rapidly. Another advantage of the on-line terminal is that through the use of conversational language, it can be programmed to ask questions of the user and obtain a reply. This is a great help in that no longer does the user have to know the process. The steps of the process are programmed in the machine. The user simply has to answer a series of questions by typing in the data for the building being designed. This is, in many ways similar to the interchange which previously took place between the engineer and the architect.

The architect simply supplied information about the building in response to questions from the engineer who then translated it into load determinations.

One drawback of this technique is that, because of the slow typing speed of the terminal, there is a relatively large amount of time spent in waiting for questions to be typed. Then too, after using the program for a while, the user knows each of the questions, and this process becomes tedious. Thus, some mechanism needs to be provided so that the questions themselves can be bypassed during execution of the program.

If this technique of using questions to request data from the user is applied, the form of the questions becomes important. Obviously, the text must present a clear idea of the information desired and the form and method of entering the data into the computer. It must be concise to conserve typing time. It must also use the terminology of the architect and use the same variables which he is accustomed to using such as expressing the type of walls in terms of the materials used rather than the u-factor. If the program is to free the user from dependence on charts or handbooks, a list of some values which are typically used should also be presented. This allows the user who is unfamiliar with typical values to use the program. It also provides an additional advantage in that if a designer is comparing two types of wall materials before he has made a decision on the type of roof to be used, he has a dummy value readily available which can be used until a roof material is selected.

While having a set of representative values available for use is important, it is impossible to present all of the possible combinations which could be used. Therefore, the user should be able to enter values other than those listed. If this is to be permitted, the program must be capable of informing the user on what the form and dimensions of this data are to be. It also should check to assure that the data entered is within a feasible range of values and give an error message or warning if the data entered is outside of this range.

There are other advantages to this on-line technique which apply particularly to the novice user. First is that little or no knowledge of the computer is required for its use. The few instructions necessary for use of a terminal can be contained in two or three pages. Once put on-line, the terminal itself can be used to instruct a person in the necessary procedures. The other advantage is that extensive error detection devices can be built into the program to immediately check the input from the user. With conversational language on a terminal, a message can be immediately given to tell the user that an error has occurred and to suggest what corrective action should be taken. This is contrasted to the use of card decks where there is a considerable time lag between submission of a program and the receipt of the results. Quite often, several of these submissions are required before the correct output is obtained.

There are several other features which should be incorporated in an energy calculation program relating to the scale at which the

architect works in the early stages of design. The program itself should be structured to consider the whole building or very large portions of it rather than individual rooms or areas. At this stage, the architect is manipulating large design elements such as the material for an entire building. He is making comparisons between 50% or 80% of glass on an exposure rather than between six or nine windows. The program should allow the user to express variables in this form and scale.

Naturally, if the program is to be useful, it must be applicable to any type of building. First, it should be able to be used, no matter what the primary function of the building is to be. It should work equally well in the design of a school or a department store. However, it should make the necessary adjustments in the energy load for the different types of buildings. Second, it should be able to work with any building form or shape. If one considers the infinite number of building shapes that could be conceived, the magnitude of this problem can be understood. The final program, then, must make some compromise between being able to consider all shapes and being able to consider those which are commonly used. The ability to satisfy this requirement determines, to a large extent, how effective the program will be.

Related to this requirement is the ability to work with various sections or parts of buildings. This is important for two reasons. As explained above, the success of a program is largely dependent upon its ability to deal with a great number of building shapes.

Since most buildings are a combination of several basic forms, one way of approaching the problem is to break the building into its various forms, calculate the energy load for that form, and then sum the loads for the various forms. The areas in common between two forms must, of course, be compensated for.

Another reason for being able to consider only certain sections of a building is because of the differences in load which occur in various sections. Perhaps the most significant difference is between the top floor of a building and one of the intermediate floors. The top floor calculations include the roof load, while the intermediate floor load include neither a roof load nor a floor load. If the user wishes to make a heating/cooling calculation for the purpose of determining the approximate size of air conditioning ducts, he has to be able to calculate the respective loads in each of these sections in order to obtain satisfactory results.

Of course, all of the other program requirements are inconsequential if the output of the program is presented in a format which is not clear and easy to understand. The dimensions in which the final results are expressed must also be meaningful. Units of energy such as watts or btu's are somewhat nebulous, even when reviewed by engineers. Thus, the results should be expressed in some form which is more easily understood. One possible way is to convert the energy load into dollars per year of energy costs. There are drawbacks to this idea because the energy to cost ratio depends on the type of system used. At this stage of the design,

it is unlikely that such a system has been chosen. If however, this limitation is understood, such a conversion can be made. Also since the primary use of the program is to be in comparing alternatives, and the same system is used for both alternatives, the effect of this conversion should be minimized.

There are two other requirements which should be met by any program for energy estimation. The first of these is that it should facilitate reruns with different data. This is especially important if several different comparisons are to be made. If it is an arduous task to change the data and reexecute the program, there will be a temptation to limit the number of comparisons made or combinations tested. For terminal use, there should be a provision for storing the input data from the previous run so that only the data which is to be changed has to be entered. If possible the results should contain a section which summarizes the differences in output from different runs.

The other requirement is that the program contain a technique for the updating of the program. There are two aspects which should be considered. One is that data which is stored in the program should be stored in a way which makes it accessible for change rather than imbedded in the program logic itself. The other is that with the terminal, some routine should be available to display what data is stored and allow change of that data with relative ease. This does not mean, however, that access to the procedure for changing this data should be available to everyone. Some form

of protection should be included to prevent unauthorized changing of data files.

This is a considerable list of requirements for a program to meet. It is probable that no program will completely satisfy all of them, but any program which is to be useful must meet all of them to some degree. To evaluate the programs which are now available as well as any program which is developed, it might be useful to summarize them into a set of criteria for evaluation.

The criteria are as follows:

REQUIREMENTS FOR AN ENERGY LOAD CALCULATION PROGRAM FOR ARCHITECTS

1. Program written primarily for a comparison of alternatives.
2. Interactive mode should be used to decrease turn-around time.
3. Program should be in the form of questions so that the architect does not have to know the process or computer programming.
4. Bypass of the question should be permitted once the program becomes well known.
5. Typical values should be provided for each question so that reference to secondary sources is eliminated.
6. Provision should be made for entering values other than reference values.
7. Extensive error checks should be provided for all data entered.
8. Program should be applicable to any type of building and any form or shape.
9. Capability for working with only certain sections of a building should be provided.

10. Variables should match those normally used by the architect in terminology and scale.
11. Results should have sufficient accuracy for a valid comparison of results.
12. Output should be in clear, understandable form.
13. Rerun of the program should be possible without having to enter all of the data each time.
14. There should be a provision both for display of data stored in the files and for changing that data.

CHAPTER IV

METHODOLOGY FOR ENERPAC

After considering the advantages afforded by use of the interactive mode of operation, this method was chosen for implementation of the program. At the time the program was begun, this decision dictated use of the Conversational Programming System developed by I.B.M. for interactive use of their Model 360 Computer. This system allows the user to write programs in CPS PL/I or CPS BASIC. The CPS PL/I language was chosen because it is a much more powerful language, especially in its capability for the manipulation of character strings, which were expected to be an important part of the program.

In evaluating the details of any computer program there are two aspects which should be discussed, the programming and the methodology. The programming deals with the mechanics, such as the restrictions imposed by the computer hardware or the advantages or limitations of a particular language. The program methodology investigates the techniques used, such as the method used for calculation of the transmission loss through a wall.

While there were some small restrictions in the language, such as the fact that structures could not be used, the most significant programming problems resulted from the storage and core space limitations. CPS PL/I allows only four 4000 byte blocks of storage capacity, called "pages" to be loaded into core at any

one time. Most programs of any consequence are much larger than this. As a result, when the program was developed it was necessary to segment the main program into several smaller units or subroutines. A short main program, ENERGY, was then provided to call each of these subroutines into core in the proper sequence and remove it after it had been used. Although the main procedure used only 2K, one complete "page" was the minimum storage block which could be used. Because of the four page limitation, this restricted all subroutines to three pages or 12,000 bytes, which proved to be too small for efficient operation. It also eliminated the technique of calling one subroutine and then having that subroutine call a second subroutine, which is standard programming practice. The consequence was that a great amount of time was consumed in organizing the content and sequence of the various subroutines for efficient operation.

The second limitation was that CPS PL/I allowed only 256 different symbols to be used for variable names, statement labels, or other identification functions. This quickly became a limitation. The problem was further complicated by the fact that the CPS PL/I manual[19] makes no mention of either symbols or this limitation. Only by experimenting with typical statements was it possible to determine how many symbols a particular statement used. Breaking the program into subsections, as discussed earlier, helped to alleviate the problem, since this practice allowed 256 symbols in each subroutine. In spite of this, the generous use of statement

labels to clarify the flow of the program, had to be avoided and certain techniques which required a large number of symbols could not be used.

Another problem encountered was that variable names and values were not shared by external subroutines within the same program. This meant that, if data was of sufficient quantity that it could not be transferred through arguments (about 20 items), it had to be written on a separate file and then read-in each time it was used in a different subroutine. This procedure of reading in values from the file at the beginning of each subroutine proved costly in storage space and symbol capacity. Then too, the files upon which this information was stored had to have all data with a similar format. This meant that data had to be stored in blocks which were equal to the longest record which was being stored. Since some records were 5 bytes long while others were 80, this was not too efficient.

The final problem concerned the size of the load/save file in which all of the programs with the same major account number are stored. This file is approximately 65K in size. If the total of all programs saved exceeds that number, no additional program space is available. Since this program required about 60K, there was little room left for other terminal users. Thus, after the storage space was all allocated, it became necessary to erase one subroutine to test another. This problem was never resolved.

In the development of this program, these problems associated with programming proved to be the primary determinant of the structure and the most time consuming. This occurred, not because of their complexity, but because of the limited amount of information available from the manual or other sources. They were solved mostly by trial and error which proved very costly in both time and computer use cost. In retrospect, they were of far greater consequence than any of the problems associated with the development of program methodology.

PROGRAM STRUCTURE

Organization

As a result of storage space limitations discussed earlier, the program was broken down into a number of subroutines. These were controlled by a short main program called ENERGY. A list of the subroutines and a description of their function is as follows:

1. INPUT - Types out questions to the user and stores data which is entered.
2. TEST - Generates typical data for testing operation of program.
3. CONVRT - Converts data from the form in which it is entered to the form required for calculations.
4. SUMMRY - Prints the main data which was entered to allow a check by the user.
5. OUTPUT - Calculates the various areas of the building, such as glass area or total floor area.

6. CALC - Performs the actual energy load calculations.
7. UPDATE - Allows reading and changing data in file "TEXT".

Basic Operation - Main Program: ENERGY

To illustrate the basic operation of the program, Figure 3 was developed. It shows, in block diagram form, the relationship between the main procedure, ENERGY, and the subroutines listed above. The first operation is the LOGIN procedure, described in the User's Manual, which connects the terminal and loads the program into the computer. After the EXECUTE command, operation of the program is initiated. The first event is a question listing the eight options available and a request for selection of an option by the user. These eight options are:

1. Energy calculation for entire building (Basement omitted)
2. Calculation for the top floor only
3. Calculation for the intermediate floors only
4. Calculation for the ground floors only
5. Calculation for the floors below grade only
6. Calculations for entire building with question text omitted
7. Listing of questions. (Requires 10 minutes)
8. UPDATE option for checking and changing information in data files

As shown in Figure 3, if option 7 or 8 is selected, control passes to the UPDATE subroutine which allows the reader to either read or change the data in the file "TEXT". After completion of the UPDATE subroutine, control is passed to the end of the program.

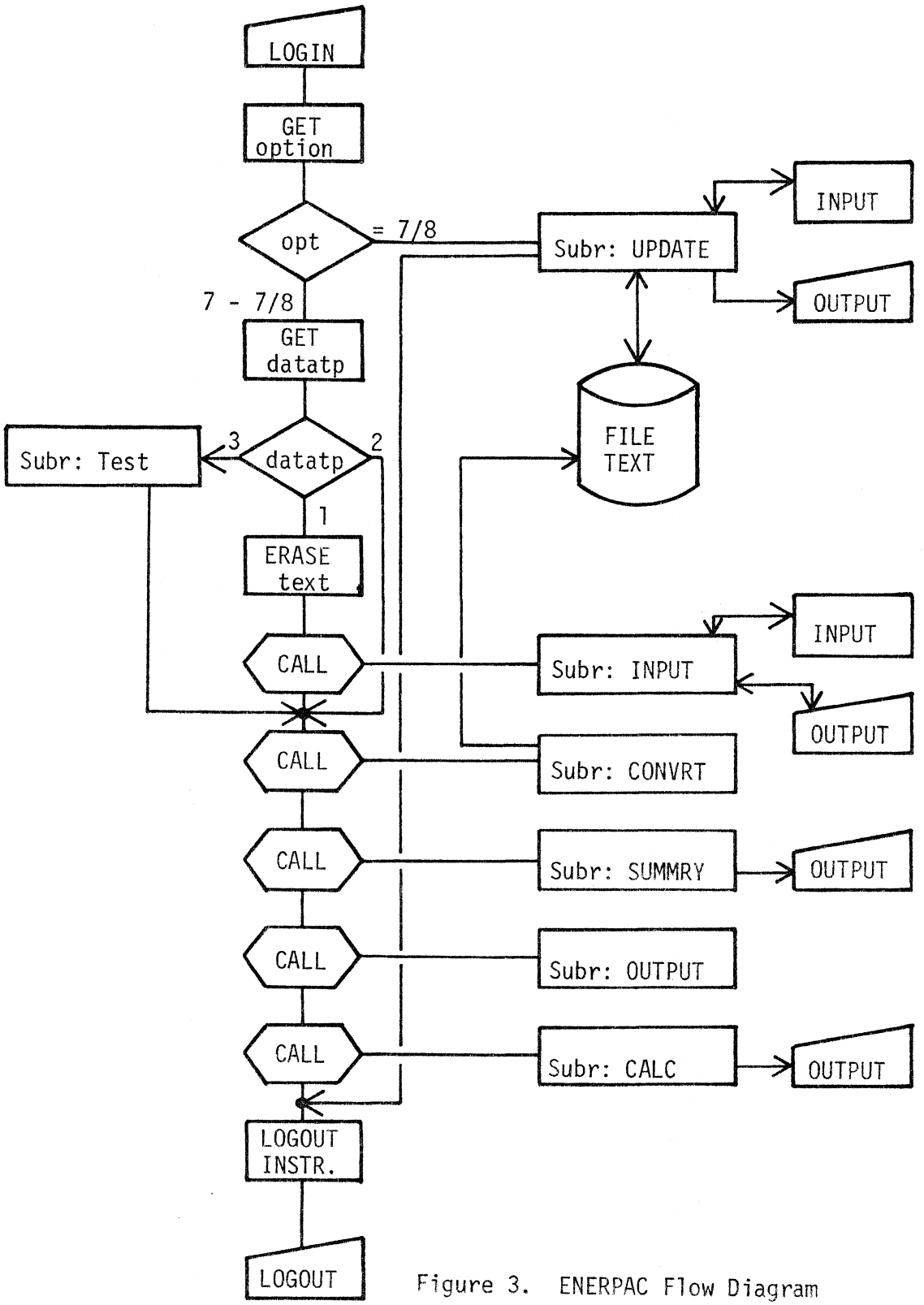


Figure 3. ENERPAC Flow Diagram

If any of the other options are selected, the types of data which can be used are listed. The three data options are:

1. New data, to be entered during execution of the program.
2. Existing data, from storage, to be used.
3. Test data to be generated.

The "test data" option calls the subroutine TEST which generates values for the same data which is typical of that normally entered by the user. This permits a check of the complete program sequence to assure that proper flow is maintained. It also produces known results to check the accuracy of the program. The "existing data" option permits the user to change selected data values and rerun the program without having to go through the complete input routine. This procedure is explained in the User's Manual. The "new data" option erases the data in storage from the last run of the program so that new data can be entered from subroutine INPUT.

Following this is a series of CALL statements which loads the desired subroutine into the program and executes it. The subroutines are processed in the sequence shown in Figure 3. A more detailed description of the operation of certain subroutines is given later in this chapter. After these subroutines have been executed and the results printed, control is returned to the main procedure which types instructions for ending operation of the program. The user then executes the LOGOUT procedure to disconnect the terminal from the computer.

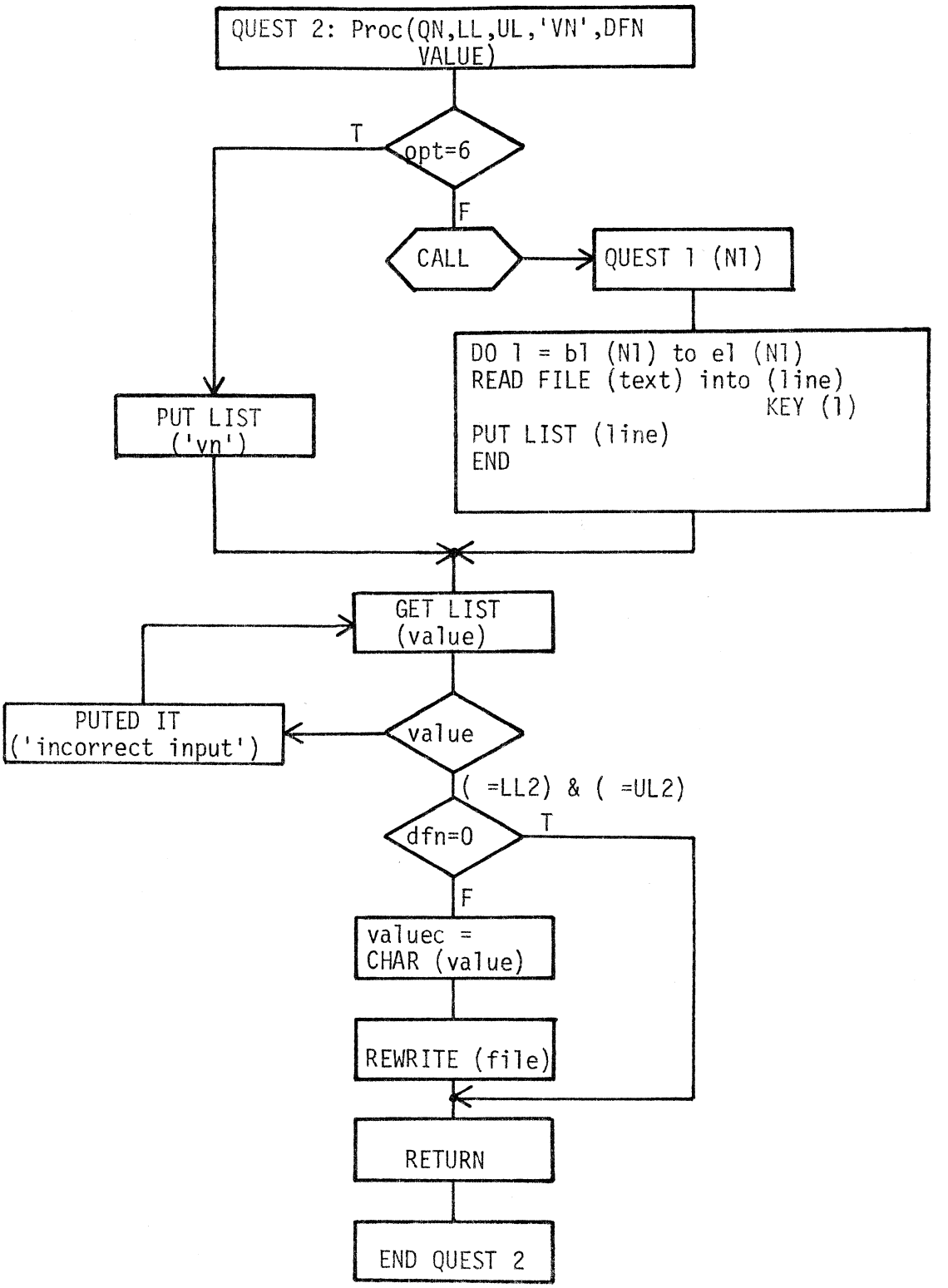


Figure 4. Flow Diagram for Subroutine INPUT/QUEST 2

Subroutine Input

The structure of the Input section is the most complicated subroutine, primarily because of the many options which must be permitted in a generalized program. To make the program simple for the user requires much more program logic than if the user could be depended upon to use his logic. Details of this logic can be found in the listing of the program given in the Appendix.

The INPUT subroutine consists of a series of questions which request information from the user. The sequence of these questions depends on the data which is entered in response to previous questions. But to understand operation of the subroutine, it is only necessary to understand the operation of one typical question. In the actual program, the technique is to use a call statement for each question. An internal subroutine to execute the actual logic of the question.

A typical call statement, and its associated arguments are of the form:

CALL quest 2 (N2, LL2, UL2, VN2, DFN2, VALUE)

where the arguments are as follows:

N2 - Number of the question being executed

LL2 - Lower limit which the value entered by the user may have

UL2 - Upper limit which the value entered by the user may have. If the value entered is not equal to or between LL2 and UL2, an error message will be generated.

VN2 - A character string used to write the name of the variable for which input data is required. It is only printed when the option omitting the question text is in effect.

DFN2 - The "data file number" or key to indicate the storage location in the file into which the data will be inserted.

VALUE - The argument which receives the value for the variable entered in the subroutine. This value can then be used in subsequent program logic.

A flow diagram for the logic of the question subroutine is shown in Figure 4. The first step in the subroutine is a check to see if the option omitting the question print-out, #6, is in effect. If so, the variable name for the input required is printed. If not, internal subroutine QUEST 1 is called. The argument N1 is simply the number of the question being printed. A do-loop then reads the question, line by line, from the file TEXT and prints the question for the user. A sample question is as follows:

```
Q(11) - Enter the number for the building shape which
        most nearly approximate your building:
        1. Square
        2. Rectangular
        3. L-Shaped
        4. Circular
```

This form of question, with the user entering an integer from 1 to 4, was chosen for most for several reasons. Most important, it eliminates the confusion which can occur when entering real numbers because of such things as dimensions, significant figures, or decimal point placement. Second, by entering an integer, it is easier to access

a group of values associated with that particular number by putting them in an array with that number as a subscript.

Having typed out either the question or simply the variable name, the computer then types the word "value" requesting the user to enter the desired value for the variable. If the value which is entered is equal to or between the two limits, LL and UL, then control passes to statement RS2. If however, the value is outside these limits, the message:

INCORRECT INPUT. Reenter number from 1 thru 4.

is printed where 1 and 4 are LL and UL respectively. Control is then returned to statement AS2a for another entry of the value. After an acceptable value has been entered, a check is made to determine whether the value is to be stored in the file TEXT. If the value is to be stored, it is first transformed from a floating point value or integer to a character string. It is then stored in the file with a REWRITE statement. If the value is not to be stored, the two previous steps are omitted. The final step in the subroutine is to return to the calling program for execution of the next statement. In the final version of the program, there are actually three subroutines used for questions:

1. QUEST 1 - Also shown in Figure 4, is used to print out the text of the question from BL(N1), the first line of question N1, through EL(N1), the last line.

2. QUEST 2 - As described above prints the question, requests a value and stores that value if needed.
3. QUEST 3 - Functions similar to QUEST 2 except the logic is altered for use with subscripted variables. It is used for all of the value which are dependent on the number of exposures, for example the percentage of glass in each wall or exposure, PCTGL(EX).

The file TEXT, which is used to store each line of text for the questions, is stored on 15 tracks of an on-line disc. The file was given REGIONAL (1) DIRECT organization. This allows reference to each line by specifying the "key" or number assigned to that particular line which contains one 80 character record. This differs from other data set organizations where the records are referenced solely on the basis of their sequence in the data set. The REGIONAL (1) organization does reduce the number of records that can be stored per track, because of the necessity of also storing a key, however it reduces the time required for accessing the records which is more important. Each record or line of text is a maximum of 80 bytes or characters long. This, rather than 130 characters, was done to keep the typed questions in a format similar to a regular printed page and also to facilitate the use of cards for a back-up copy of the data set.

Subroutine TEST

This subroutine is used to generate the same set of variables which the user would enter in the INPUT subroutine. This was done

so that the entire program can be checked rather than just the calculation section. In addition, it allows the accuracy of the program to be quickly checked to assure that program logic and internal data has not been changed. Although originally intended for use only in the debugging of the program, it has been retained because of these features.

Subroutine CONVRT

The CONVRT subroutine was necessitated by the decision to have the user enter an integer value corresponding to a particular variable rather than the actual value of the variable. As described in the INPUT section, this technique helps to reduce the amount of input requested from the user. By entering the value for the building type, and then retrieving the other values associated with that building type, such as the lighting load, from storage, it is possible to eliminate the need for the user to enter seven different variables related to that building type.

Subroutine SUMMRY

Although error tection for values outside of an acceptable range has been provided, it was considered necessary to provide a listing of the primary values which were used for the calculations. Not only does this provide a check against mistakes which occur within the acceptable range, but it also provides a permanent record of the input for each printout of the results. This is important if a great many combinations of variables are to be compared.

Subroutine OUTPUT

This subroutine is used to calculate each of the areas required for subsequent load calculations. These are primarily the wall, glass, and door areas for each of the exposures and the totals for the building. The roof area, floor area, and basement wall area are also calculated. A total floor area is calculated by multiplying the floor area by the number of stories in the building. Two basement wall areas are calculated. BMWLAH is the area of the basement wall area between ground level and eight feet below ground. BMWLAL is the area of basement wall greater than eight feet below ground. This was required because of the different techniques used for calculating the heat transfer through these two wall sections.

Subroutine CALC

The subroutine which performs the calculations for the various loads is, of course, the most important in the program. As such, it requires a detailed analysis to explain the techniques which were used. The calculations fall into several general groups: (1) transmission, (2) solar, (3) ventilation, (4) infiltration, (5) lighting, (6) occupancy, (7) power, and (8) costs.

Central to the development of a technique for estimating energy loads is the procedure for predicting weather data. Various techniques have been used, as described in the Chapter on "Existing Programs". Two factors influenced the choice of a procedure for this program. First, it was considered desirable to use a method which

required the input of a minimum amount of data from the terminal. Second, the routine had to be one which required a minimum of machine storage space. After reviewing the various methods, it was decided to use two separate short routines, one for predicting outside air temperatures and another for predicting the solar radiation.

The method chosen for predicting the outside air temperature was originated by Will Brown in an article, "Typical Year Temperature Data for System Evaluation."[20] The method uses one harmonic equation with different constants to predict the normal mean temperature for different latitudes. The equation is:

$$t_c = t_m + t_a \ (\text{Sin} \ \ (T-P)\theta) \qquad (1)$$

where:

t_c = calculated mean monthly temperature, ^oF

t_m = normal mean annual air temperature, ^oF

t_a = normal mean temperature amplitude, ^oF

T = time, months (expressed as an integer from 1 to 12)

P = phase angle for months

θ = conversion factor to translate T to angular measurement in degrees per month

The inside design temperature was assumed to be 75^oF. Thus, the formula used for the calculation of the transmission through all surfaces above ground was:

$$H_t = AU \ (t_c - 75^o) \qquad (2)$$

where:

H_t = heat loss/gain transmitted through surface, BTU/HR

A = area of surface, sq. ft.

U = coefficient of transmission (U-factor), BTU/HR-SQ.FT.-^{o}F

t_c = calculated outside air temperature (see eq. (1))

Transmission through basement walls was calculated from the procedure recommended in Carrier's Handbook of Air Conditioning System Design.[21] Using this procedure, a separate calculation was made for the transmission through, (1) the wall from the ground level to eight feet below ground, (2) the wall from eight feet below ground to the depth of the basement, and (3) the floor slab. The formulae used were:

Wall (0-8 ft. depth)	TRBWH = BWAH (.08)(DELT)	(3)
Wall (8 ft. to total depth)	TRBWL = BWAL (.08)(GRNDT)	(4)
Floor	TRFL = FLA (.05)(GRNDT)	(5)

where the variables correspond to similar variables in equation 2. DELT is the difference between the outside air temperature and 75^{o}F, the inside air temperature. GRNDT is the difference between the ground, which seldom goes above 65^{o}F, and the inside air temperature. The components were then summed to obtain the total transmission for the building section being considered.

The procedure used to calculate the solar load employed storage of the solar heat gain factors for each date, latitude and exposure in the file TEXT. These factors were then retrieved and

used for the respective calculations. The solar heat gain for the roof and walls was obtained by multiplying the solar heat gain factor for each exposure by the area for that exposure:

$$SLWL(EX) = SHGF(EX)\ (WLAN(EX)) \tag{6}$$

The solar load for the windows was calculated in the same way except a reflectivity components was added:

$$SLGL(EX) = SHGF(EX)\ (GLA(EX))\ (REF) \tag{7}$$

where; in equations 6 and 7:

SHGF(EX) = solar heat gain factor/exposure, btu/hr./sq.ft.

WLAN(EX) = net wall area/exposure, sq.ft.

GLA(EX) = glass area/exposure, sq.ft.

SLWL(EX) = wall solar load/exposure, btu/hr.

SLGL(EX) = glass solar load/exposure, btu/hr.

These values were then summed to get the total solar load.

Calculations for the other components of the building energy load use factors which are keyed to the building type. There are two components of the typical load which were not included, infiltration and appliance loads. Infiltration was not included because of the present practice of sealing buildings so that infiltration becomes a negligible part of the total load. Appliance loads were omitted since they are essentially fixed loads, as discussed earlier, and do not affect the architect's decisions.

The remaining loads were calculated as follows:

Ventilation:

$$OCCNO = \frac{FLAT}{OCC} \tag{8}$$

$$CFM = OCCNO\ (VENT) \tag{9}$$

$$VENTLD = 1.08\ (CFM)\ (t_c - 75) \tag{10}$$

Lighting:

$$LITELD = LITE(FLAT) \tag{11}$$

Occupancy:

$$OCCLDS = OCCNO(SHP) \tag{12}$$

$$OCCLDL = OCCNO(LHP) \tag{13}$$

Where:

OCCNO = number of occupants, occupants

FLAT = total floor area for building

OCC = occupancy load factor, sq.ft./occupant

CFM = amount of ventilation air required, cu.ft./min.

VENT = ventilation load factor, cfm/occupant

t_c = outside air temperature, ^oF

VENTLD = total ventilation load, btu/hr.

LITE = lighting load factor, btu/hr./sq.ft.

LITELD = total lighting load, btu/hr.

SHP = sensible heat load/occupant, btu/hr./occupant[29]

LHP = latent heat load/occupant, btu/hr./occupant[30]

OCCLDS = total sensible heat load, btu/hr.

OCCLDL = total latent heat load, btu/hr.

These loads are then added to the transmission loads and solar loads to determine the total load for the building.

The final group of calculations are those related to cost. Expected total construction costs for the building, expected

mechanical and electrical costs (included in the total cost), and expected mechanical equipment space are determined by:

$$EBC = BLDCST(FLAT) \quad (14)$$

$$EMEC = MECST(FLAT) \quad (15)$$

$$EMES = MESPA(FLAT) \quad (16)$$

Where:

FLAT = total floor area of the building, sq. ft.

BLDCST = building cost factor, dollars/sq.ft.

EBC = expected building cost, dollars

MECST = mechanical/electrical cost factor, dollars/sq.ft.

EMEC = expected mechanical/electrical cost, dollars

MESPA = mechanical/electrical space factor, sq. ft. (mech)/ sq.ft. (total)

EMES = expected mechanical/electrical space, sq. ft.

Annual operating costs are determined by multiplying heating/cooling load totals by the operating cost for a typical system for that type of building.

Obviously, a discussion of the methodology used for a computer program of any size can only focus on the major elements without becoming tedious. If a more detailed investigation is desired, it is necessary to investigate a listing of the program itself. With this particular program, the storage space limitations of the hardware have necessitated the omission of any documentation or "comment" statements within the program. Also, only a minimum number of statement labels were used. The programming techniques used,

however, do not require an advanced knowledge of PL/I. Thus, while time consuming, it is still relatively easy to follow the actual techniques used in the program.

CHAPTER V

ANALYSIS OF THE RESULTS

The first section of the "Results" is an Input Summary of the most important input values used in the calculations. This summary, shown in Figure 5, provides a listing which can be used to assure that the intended values were used for calculations. It also provides a record of these values when the results are referred to at a later date. The input variables actually printed in the Summary include only the building dimensions and other major variables. It may be an advantage to give a complete listing of all variables used. Doing this would prevent any mistakes concerning the input variables used for the calculations. Following this is a listing of the variables as they are read from the file into the CALC subroutine. A procedure was found which eliminated this print-out between the Input Summary and the Results, but it consumed too much space in subroutine CALC.

After this, the Results are printed. As can be seen in Figure 6, the results are in the form of the b.t.u. hour required for the month with the maximum heating load and maximum cooling load. When the basic concepts of the program were first being formulated, it was decided that two types of output would be included. The first would be a yearly output showing the various

```
               ***************
               *INPUT SUMMARY*
               ***************


OPTION =     1
BUILDING TYPE =      1
BUILDING SHAPE =      2

BUILDING DIMENSIONS:
      LENGTH(1) =     50   FEET
      LENGTH(2) =    100   FEET
      LENGTH(3) =     50   FEET
      LENGTH(4) =    100   FEET
      HEIGHT =        50   FEET

NUMBER OF STORIES =          3
ORIENTATION OF EXPOSURE =      1

            U-FACTORS:
WALLS         0.19
GLASS         1.13
DOOR          0.55
ROOF          0.08
FLOOR         0.29
BSMT. WALL    0.00

LATITUDE OF BUILDING SITE =  37.2  DEG.
INSIDE DESIGN TEMPERATURE =  75 DEG. F.
```

Figure 5. Typical Input Summary

```
***************
*   RESULTS   *
***************
```

	MAXIMUM HEAT	MAXIMUM COOL
MONTH	1	7
TRANSMISSION LOAD:		
WALLS	-62047	1612
GLASS	-389806	10126
DOORS	-37046	986
ROOF	-9199	239
FLOOR	-2500	-2500
BSMT. WALLS	0	0
SOLAR LOAD/GLASS	1000	1000
VENTILATION LOAD	-638671	16591
LIGHTING LOAD	150000	150000
OCCUPANCY LOAD	192857	192857
TOTAL LOAD	-796312	370911

EXPECTED COST AND SPACE REQUIREMENTS:

TOTAL COST OF BUILDING = $ 348750.00

COST OF MECHANICAL AND ELECTRICAL EQUIPMENT = $ 108000.00
(INCLUDED IN TOTAL COST)

MECHANICAL EQUIPMENT SPACE REQUIRED = 675.00 SQ. FT.

Figure 6. Typical Results from Program ENERPAC.

components of the load and cost data. If the user then wanted a more detailed look at the monthly values for each component of the load, it could be displayed. This is the way the program was first written. It was discovered, however, that the statements which were required for display of the monthly summary used a large amount of storage space on the load/save file. When space became a problem, it was decided to erase this portion of the output and to provide a display loads for only the two critical months. It should be understood that the logic to make calculations for each month is still contained in the program, it is only in the actual display of information that a change was made.

Also omitted in the final stages of programming was a print-out of the net wall area, glass area, and door area for each exposure as well as roof, floor, and basement wall areas. These intermediate values serve no purpose except to assure the user that correct values were used in the calculations. However, after extensive testing of the program to verify the accuracy and logic of the area calculations, it was decided that this information could be omitted. To restore this feature, it is only necessary to reinstall a sub-routine to extract these numbers from the file TEXT and print them in the format desired.

Another consequence of the storage space problems was that it was necessary to omit solar load routine which had been developed. Originally, the technique used was to store the solar heat gain factors for each month, developed by ASHRAE, for a horizontal surface

and each exposure at several different latitudes. These values were then extracted from storage and used in the calculations. Unfortunately, this required a 12 by 18 array both for the solar heat gain factor and for the solar heat gain. The declaration of these two arrays without any logic whatsoever consumed 3456 bytes of the 12,000 available in the calculation subroutine. Obviously, this amount of space could not be spared. The result was that the original solar heat gain technique could not be included in the calculation subroutine. Average values for solar load had to be incorporated into the program logic resulting in slight inaccuracies in the solar loads. Transmission values are also affected by this change because the sol-air temperature could not be used. Thus, the cooling load values are lower than they would be if the sol-air temperature were used.

The final problem created by insufficient load/save space was that monthly costs for the heating/cooling load could not be included. The original intent was to associate a particular air conditioning system with each building type used in the program. Typical operating costs, expressed in dollars/btu, would then be multiplied by the load for the month to obtain monthly costs. In researching the existing literature, however, it was found that little information on the operating costs for various systems is available. Most of the data available was insufficiently documented for further use. There is a need for further development of this type of data for future research.

All of the calculations based on the average load factors provided accurate results. But because they are based on average values, the user should exercise caution if radical design solutions are being evaluated. In this event, the user should consult the sources given in the User's Manual to obtain more precise factors to enter as input.

It can be seen that the storage space limitations have restricted the type but not the accuracy of the results obtained. If the program is converted to TSO, and a routine for calculating the sol-air temperature added, the resultant program should produce the full range of information necessary for the architect to effectively choose between alternatives being investigated.

CHAPTER VI

CONCLUSIONS

In the office of T. Vincent Learson, Chairman of I.B.M., is a plaque paraphrasing a line from Machiavelli's The Prince, "It must be remembered that there is nothing more difficult to plan, more doubtful of success, nor more difficult to manage, than the creation of a new system." After having tried to create a new system, albeit simple, it is easy to understand why the plaque is displayed.

In reviewing the performance of the program, perhaps the first step would be to evaluate the choice of an interactive terminal. The terminal system itself operated quite well, the only disadvantage being the relatively long time required to type some questions. However, when comparing the time required to make calculations using the program, to the time required for manual calculations, the savings in time is quite apparent. Storage space limitations, as discussed, provided the most significant problem. This was further compounded by the total lack of information from the IBM CPS Manual, the only reference on the system. Since a new system, TSO, has now replaced the CPS system, it is hoped that this problem will be minimized.

The other hardware problem was the operation of the computer itself. Sporadic operation of the system, coupled with the repeated and unexplained loss of parts of the program, resulted in a tremendous amount of lost time. If faced with the project, the first task would be to develop a set of job control language (JCL) utility

programs for reproducing all data sets and any part of the load/save file desired. Card back-up of the program would then be made at regular intervals or after every major addition or modification to the program.

To evaluate the program itself, the list of Requirements for an Energy Load Calculation Program for Architects developed in Chapter III can be used. Since the program was developed from the list, it meets these criteria quite well. The first four are met completely. The elimination of reference to a secondary source, #5, was met on all but the question concerning the latitude of the building site. The original concept was to give the user a list of latitudes for local sites as reference and then request input of the actual latitude of the site. If this was unknown, the user could then simply input the name of the state and the program would use the latitude for the capitol of that state. The method was tested, and worked as desired. When storage space became a problem, however, this subroutine appeared the least critical to the operation of the program relative to the space consumed. The STATE subroutine was therefore erased. The format of the latitude question was then changed to list fourteen local cities and their latitudes. It is expected that most users will be able to use these values directly or be able to interpolate. For other sites, a map of the United States showing latitudes is provided in the User's Manual.

There was no problem in meeting 6 and 7, but 8 and 9 were only partially satisfied. Requirement 8 was that the program be applicable

to any type of building with any form and shape. Related to this was requirement 9, that a capability for working with any section of the building be provided. Every one of the space allocation programs developed thus far has had considerable difficulty with this problem. The usual approach to defining the shape of a building has been to develop either a grid or radius system and then to approximate the building shape using short line segments. The problem becomes much more complicated in three dimensions. All of the techniques investigated required a great number of data points to work with a building of irregular shape. This contradicts the idea of terminal use that data input should be minimized. Finally, four basic building shapes were chosen. The assumption was made that they had vertical walls and flat roofs. This was necessary to proceed with development of the program. This problem remains as one to be solved if the program is to become more general. The use of nine different building types and the provision for entering data for the parameters used, met the requirement for applicability to any building.

Since the program was designed for use by architects, requirement 10, that the variables should match those normally used, was easily satisfied. If the program was going to be used by an office, they would most probably insert material types and other values which they frequently use and would modify the number of items in storage to make the program efficient for their particular office.

The results, of course, are the most important part of the program. Not only are they required to be accurate, but also must

have dimensions and a format which makes them understandable. This was discussed in Chapter V. As shown there, the results are of sufficient accuracy for the purpose of this program. The format presents the desired information. It does, however, suffer from the fact that the Summary and Results are not presented together.

Another of the consequences of insufficient storage space was that the routine adopted for changing a limited amount of data and rerunning the program is relatively inefficient. The method finally chosen was to use the UPDATE subroutine, already a part of the program, for changing the desired data in the file and then providing an option to execute the program using the data stored in the file. The reason for the inefficiency is that the UPDATE option of the program must be executed for each item of data which is to be changed. Then too, the user must use the data file schedule to find the location in which data is to be inserted. This provides a great chance for error. If the storage capacity were available, a program to perform this operation should require only that the user type in the name of the variable and the value placement in the file would be automatic. It should also allow any number of values to be changed before each rerun.

Requirement 14, that there should be a provision for display and change of data in the file TEXT has been met with the UPDATE subroutine. The only drawback of this subroutine is that the data file schedule must be consulted to determine the number for the lines involved.

Thus, it can be seen that, with a few exceptions, the program met all of the requirements set for it. Simplicity has been maintained. It is only necessary to know the procedure for connecting the terminal with the computer to execute the program. From that point, the program provides instructions for its operation. Because of the question technique and the inherent advantages of interactive operation, no knowledge of computers or load estimating techniques is necessary. The program performs as intended.

Despite the success of the program, the limitations of the CPS PL/I language have had their effect on the program so that there are still improvements which can be made. This is especially true since TSO, with its exceptionally large program storage capacity has become the primary terminal system language. It is expected that this may be 150K in core for TSO versus 16K for CPS. The advantages are immediately apparent. So the first improvement in the program under this system would be the incorporation of all of the subroutines into one program in which all of the variables and their values are known throughout. This would eliminate many of the file input/output routines which now consume both time and storage space. Another advantage of this change is that it would eliminate the typing of all of the variable names between the print-out of Summary and Results, improving the appearance and perhaps effectiveness of the output.

Another major improvement which could be made under TSO is the enlargement of the weather generation program. Since a large amount of data could be stored, it might be appropriate to incorporate a

statistical routine, such as that used for HEATRAN-2, in which the user first enters the name of the state in which the building site is located and is then presented with several cities or distinct weather areas within the state. Appropriate factors, keyed to that input would be retrieved from storage after selection of an area. To take advantage of the increased accuracy of the weather data, it would then be necessary to change the ieration loops so that hourly calculations for the various loads are made. This would make the results as accurate as possible.

The final improvement would be to solve the problem of describing the geometry of the building in a way which makes the program applicable to a building of any shape. As discussed, most of the efforts of others have thus far failed either because of their lack of sophistication of their programs or because of the excessive amount of data required. This problem becomes particularly important in the calculation of solar loads because of the need to determine the orientation of all surfaces with respect to the sun. It also is important in calculating shadow areas on parts of the building. Another feature related to this is an improved ability to treat only sections of a building. Only vertical sectioning was incorporated in the ENERPAC program, horizontal sectioning should also be possible.

Although it would represent a completely different approach, the ultimate refinement of the program would be to adapt it for use on an interactive graphics terminal. The basic framework is provided

with this program. Only the changes necessary to display the existing information on a Cathode Ray Tube (CRT) and to input data from the console would be required. The CRT display would offer several advantages. Question text could be displayed instantly rather than typed letter by letter. This would permit longer questions and thus more typical values to select from. The biggest advantage would be gained on those questions which utilize graphics. With the proper programming, it would be possible to let the user draw the actual shape of the building with a light pen. Input of various dimensions could be requested by making that dimension blink. For the input of the latitude of the building site, a light pen could be touched to a map of the United States displayed on the screen. There are many possibilities, and since the basic techniques have been developed with ENERPAC, the change-over would be simplified.

Related to future improvements in the program itself is the development of an integrated system of programs for architectural design. Such a development would effect the program in two ways. First, a system dealing with architectural design could provide a consistent methodology for use in this program, as well as others. Second, by having a complete group of programs, more people would become acquainted with this particular program, thus prompting greater usage. Such a package should include programs for space allocation, building code requirement verification, structural design, duct design, electrical design, and others.

The first effort should be directed toward the development of a methodology for the definition of building shape which could be used in all future programs. Having done this, each subsequent program could be structured to fit within this framework. During the development of each new package, great care must be exercised to assure consistency of approach, as well as units, variable names, and other details. The establishment of such a series of programs will undoubtedly require a great amount of effort. Once completed, however, it would provide architects with a very important tool for building design.

The final step with any program is its implementation. There are certainly advantages to using the ENERPAC system for making energy load calculations. Since architecture students are expected to be the primary users, the freedom from knowledge of the process involved is expected to be the most significant change from present practice. Second in importance would be the ability to evaluate several alternatives. The saving in time over manual calculations, also makes the system attractive. Hopefully, the users will have greater confidence in the computer results than they currently have in their manual calculations. However, it will only be after the system has been in use for a period of time that a final evaluation of its effectiveness be possible.

LISTED REFERENCES

1. Alvin Toffler, Future Shock. New York: Bantam Books, 1971, p. 23.

2. Ibid.

3. "Heat or Light," Barrons, February 22, 1971, p. 1.

4. Ibid.

5. Samuel Walters. "Power in the Year 2001." Mechanical Engineering. Vol. 93 (September, 1971), 24-26.

6. "Raw Materials, You Get What You Pay For." Forbes. Vol. 108, No. 3 (August 1, 1971), 20.

7. Ibid.

8. U. S. Department of Labor. Bureau of Labor Statistics Retail Prices and Indexes of Fuels and Utilities. January, 1962-1971.

9. "Raw Materials, You Get What You Pay For," p. 21.

10. Robert D. Logcher. "ICES STRUDL: An Integrated Approach to a Computer System for Structural Engineering," Emerging Methods in Environmental Design and Planning Proceedings of the Design Methods Group First International Conference. Edited by Gary T. Moore. Cambridge, Mass. p. 79.

11. D. R. Witt and L. O. Degelman. HEATRAN 2 User's Guide. Report No. 70-7 University Park, Pa.: Department of Architectural Engineering, Pennsylvania State University, 1970.

12. L. O. Degelman. Monte Carlo Simulation of Solar Radiation and Dry Bulb Temperatures. Report No. 70-9 University Park, Pa.: Department of Architectural Engineering, Pennsylvania State University, 1970.

13. Roger Wilson, "Computerized Energy Analysis." Building Systems Design. Vol. 68, No. 3 (March, 1971), 25.

14. Joe B. Thomas and Gustave Akselrod. "CRS: The Computer in Energy Analysis." Building Systems Design. Vol. 68, No. 11 (November, 1971), 22.

15. Alan Curl. "APACE: Bridging the Gap." Building Systems Design. Vol. 68, No. 9 (September, 1971), 19.

16. Mike Gilford. "SHIS: The Liaison Function." Building Systems Design. Vol. 68, No. 9 (September, 1971), 23.

17. Ibid.

18. Harold S. Samuels, "Dynadata: Boon to the Neophyte." Building Systems Design. Vol. 68, No. 9 (September, 1971), 23.

19. Conversational Programming System (CPS) Terminal User's Manual. Program No. 360D-03.4-016 New York: IBM Corporation, 1970.

20. Will K. Brown, Jr. "Typical Year Temperature Data for System Evaluation." Heating, Piping, and Air Conditioning. Vol. 41, No. 10 (October, 1969), 61.

21. Carrier Air Conditioning Company. Handbook of Air Conditioning System Design. New York: McGraw-Hill, 1965, pp. 1-82.

22. ASHRAE, Inc. ASHRAE Handbook of Fundamentals. New York: ASHRAE, Inc., 1967, p. 497.

23. Ibid.

24. Robert N. S. Chiang. "Design of Heating, Ventilating, and Air Conditioning Systems." Unpublished Class Notes, 1971, pp. 126-137.

25. C. G. Ramsey and Harold R. Sleeper. Architectural Graphic Standards. 6th Edition edited by Joseph N. Boaz. New York: John Wiley and Sons, 1970, p. 320.

SELECTED BIBLIOGRAPHY

"A Round Table on Energy Conservation Through Higher Quality Design," Architectural Record, January, 1972, p. 97.

ASHRAE, Inc. ASHRAE Guide and Data Book, Equipment. New York: ASHRAE, Inc., 1969.

ASHRAE, Inc. ASHRAE Guide and Data Book, Systems. New York: ASHRAE, Inc., 1970.

Bates, Frank, and Douglas, Mary. Programming Language/One, Second Edition, Englewood Cliffs, N. J.: Prentice-Hall, 1970.

Beck, J. H. "How to Figure a Fair Price for Purchased Heating-Cooling." Heating, Piping, and Air Conditioning, Vol. 42 (December, 1970).

Down, P. G. Heating and Cooling Load Calculations. Oxford, England: Pergamon Press, 1969.

Dagleman, L. O., and Witt, D. R. HEATRAN User's Guide (Heat Transmission Analyzer for Building Air Conditioning). University Park, Pa.: Department of Architectural Engineering, Pennsylvania State University, 1968.

Douglass, E. S., and Reeves, G. "Electric Utilities Offer New Computerized Method for Consulting Engineers." ASHRAE JOURNAL, Vol. 13 (November, 1971), 41.

Environmental Policy Division, Library of Congress for the House Subcommittee on Science, Research, and Development. "An Energy Resource Bibliography." Professional Engineer, February, 1972, p. 18.

Evers, W. E. "Ecube Computerized Energy Analysis: Energy Equipment Economics." ASHRAE Journal, Vol. 13 (September, 1971), 46.

Flynn, J. E., and Segil, A. W. Architectural Interior Systems. New York: Van Nostrand Reinhold Company, 1970.

Gupta, C. L. "A Systematic Approach to Optimum Thermal Design." Building Science, Vol. 5 (December, 1970), 165-174.

Herzog, P. and Kunstadt, H. "Energy Analysis for High Rise Apartment Buildings." Building Systems Design, December 1970, p. 17.

Jordan, R. C., and Liv., B. Y. H. "Analysis of Solar Energy Data Applicable to Building Design." ASHRAE Journal, Vol. 4 (December, 1962), 31-41.

Jordan, R. C., and Threlkeld, J. L. "Direct Solar Radiation Available on Clear Days." ASHRAE Transactions, Vol. 74 (1968), 189.

Kunstadt, Herbert. "Data Sheet: Electrical and Mechanical Service Power Loads." Building Systems Design, April, 1970, p. 35.

MacCurdy, D. W. "Calculating Solar Heat Loads." Progressive Architecture, Vol. 43 (December, 1962), 124-129.

Mitlas, G. P., and Stephenson, D. G. "Cooling Load Factor by the Thermal Response Factor Method." ASHRAE Transactions, Vol. 72, Pt. 1 (1967), III .1.1.

Mitlas, G. P., and Stephenson, D. G. "Room Thermal Response Factors." ASHRAE Transactions, Vol. 72, Pt. 1 (1967), III .2.1.

Moon, P. "Proposed Standard Solar Radiation Curves for Engineering Use." Journal of the Franklin Institute, Vol. 230, No. 5 (November, 1940), 583-617.

Parmelee, G. "Irradiation of Vertical and Horizontal Surfaces by Diffuse Solar Radiation from Cloudless Skys." ASHRAE Transactions, Vol. 60 (1954), 341.

Pfeiffer, David C. "SMU Study Gives Large Chiller O & M Costs." Heating, Piping, and Air Conditioning, Vol. 43, No. 5 (May, 1971), 65.

Porembski, T. T. "Predicting Annual Heating-Cooling Loads." Air Conditioning, Heating, and Ventilation, Vol. 60, No. 3 (March, 1963), 66-76.

Reynolds, J. R. "Energy for Industrial Air Conditioning." Air Conditioning, Heating, and Ventilation, Vol. 64, No. 9 (September, 1967), 55-58.

Ringquist, C. L. "First Cost Vs. Owning and Operating Cost for Air Conditioning." Air Conditioning, Heating, and Ventilation, Vol. 66, No. 5 (May, 1969), 38-44.

Schrei, H. J. "Energy Analysis for Air Conditioning Systems." Heating, Piping, and Air Conditioning, April, 1968, p. 151.

Smith, T. "Computer Service by Wire or Mail." Building Systems Design, Vol. 66, No. 8 (August, 1969), 44-48.

Spofford, W. A. "Air Conditioner Cooling and Heating Capacity and Air Flow Calculated by Computer. ASHRAE Transactions, Vol. 75, Part 1 (1969), 207.

Stephenson, D. G. "Equations for Solar Heat Gain Through Windows." Solar Energy, Vol. 9, No. 2 (1965), 81-86.

Stephenson, D. G. "Calculation of Cooling Load by Digital Computer." ASHRAE Journal, (April, 1968), 41.

Stoecker, W. F. "A Generalized Program for Steady State System Simulation." ASHRAE Transactions, Vol. 77, Part 1, (1971), 79.

Stoecker, W. F. Design of Thermal Systems. New York: McGraw-Hill Book Company, 1971.

Sun, T. Y. "Shadow-Area Calculation for Window Overhangs and Side Fins and Their Applications in Computer Calculations." ASHRAE Transactions, Vol. 74, Part 1 (1968) I.1.1.

Sun, T. Y. "Computer Evaluation of the Shadow Area on a Window Cast by the Adjacent Building." ASHRAE Journal, Vol. 10, No. 9 (September, 1968), 66.

Thom., H. C. S. "Cooling Degree Days and Energy Consumption." Air Conditioning, Heating, and Ventilation, Vol. 63, No. 9 (September, 1966), 53-54.

Threlkeld, J. L. "Solar Irradiation of Surfaces on Clear Days." ASHRAE Journal, Vol. 4, No. 11 (November, 1962), 43-54.

Threlkeld, J. L. Thermal Environmental Engineering. Second Edition. Englewood Cliffs, N. J.: Prentice-Hall, Inc., 1970.

Tull, R. H. "ASHRAE Program on Energy Requirements for Heating and Cooling." ASHRAE Journal, Vol. 10, No. 4 (April, 1968), 40.

Wetherington, T. I. "Energy Use in High Rise Apartments and Motels." Building Systems Design, May, 1971, p. 31.

Wilson, Roger. "Computerized Energy Analysis." Building Systems Design, March, 1971, p. 25.

Witt, D. R. The Development of a Computerized System for Annual Energy Analysis of Heat Flow Through Exterior Glass in Buildings. University Park, Pa.: Department of Architectural Engineering, Pennsylvania State University, 1970.

Yellott, J. I. "Solar Optical Properties, Heat Transfer Coefficients, and Shading Coefficients for Architectural Glass." ASHRAE Journal, Vol. 13, No. 3, (March, 1971), 41.

APPENDIX I

USER'S MANUAL

USER'S MANUAL

FOR

ENERPAC

Harry F. Moate

Department of Environmental and
Urban Systems

College of Architecture
Virginia Polytechnic Institute
and State University
Blacksburg, Virginia

March, 1972

PROGRAM DESCRIPTION

ENERPAC is an interactive computer program for estimating building energy loads. It was designed to provide architects and students of architecture with a procedure for comparing various design concepts and evaluating their impact on the energy consumption of the building. The program consists of a main controlling section, ENERGY, and a group of subroutines which perform the desired operations.

The procedure used is for the terminal to type questions for the user, requesting input of the required data. Extensive error checks are provided to assure that this data is within the proper range. A summary of the input data is given and then the results are printed, showing the cooling and heating loads for the building.

INSTRUCTIONS ON USE OF THE TERMINAL

The Time Sharing Option (TSO) is the system now available for terminal use. The ENERPAC program is written in CPS PL/I which runs as a subset on the TSO language. The steps shown below are written specifically for this program. If other options are desired, contact Systems Consulting for instructions.

A. Preparation (May be done in any order)

1. Contact the Professor in charge of the terminal to get an account number, subaccount number, and password.
2. Place telephone adjacent to acoustic coupler.
3. Slide the acoustic coupler switch, beside the red pilot light to the right or "on" position.
4. Press the "on-off" switch on the right side of the terminal keyboard to the "on" position.
5. Move the "local-remote" switch on the lower left side of the keyboard to "remote."

B. Dial-up

1. After lighting the receiver and getting a dial tone, dial 9. After getting a second tone, dial 951-3230. The answer will be a continuous, high pitched, tone which indicates the computer is ready for connection.
2. Quickly place the telephone receiver into the acoustic coupler. Only 6 seconds is allowed for this operation. If not done within this time, the number must be dialed again.
3. If the connection is made, the "ready" and "proceed" lights on the left of the keyboard will light and the neon light on the acoustic coupler will glow.

C. Connecting to the Computer

1. When the receiver is placed in the coupler, the terminal should type:

```
OW+43929P  US¢UN   YQAQS
#IKJ53020A  ENTER   LOGON
```

If it does not, depress the return button and it will. Following this, the sequence below must be completed with the user typing in those statements shown as "typed by user." The machine will respond as shown. If any typing errors are made, the machine will type an error message. The user must then reenter the command.

```
OW+43929P  US¢UN   YQAQS
#IKJ53020A  ENTER   LOGON
logon (typed by user)
IKJ56700A  ENTER  USERID -
(subaccount/password typed by user)
LOGON IN PROGRESS AT 17:41:22 on MARCH 3, 1972
TSO IS NOW VPI'S PRIMARY TIME-SHARING SYSTEM
REPORT PROBLEMS TO SYSTEMS CONSULTING ext. 6156 or 6603
READY
cps (typed by user)
ENTER MASTER ACCOUNT NAME
(master account name typed by user)
GOOD AFTERNOON;   THE TIME IS  17:42:13  3/03/72;
```

D. Executing the Program

1. After completing the sequence above, the program is loaded from machine storage into core by typing in small letters:

load (ENERGY, ARCH)

When the program has been loaded, the machine will type an underbar.

To execute the program, it is then only necessary to type:

XEQ

Execution of the program will then commence.

E. Disconnecting from the Computer

If the user desires to disconnect from the computer he first presses the "ATTN" button on the upper right of the keyboard.

After this, or if execution of the program has been completed, the user types "LOGOUT." The terminal will type "READY" after which the user types "LOGOFF." The final step is to hang up the telephone which is actually what disconnects the terminal from the computer.

PROGRAM OPERATION

Answering Questions

For each item of input data, the terminal will type a question, usually listing several alternatives from which to choose an answer. Following this, the terminal will type the word VALUE or VALUE 3 (N) and an underbar. The user must then type in the requested data and then press the "RETURN" key on the right side of the terminal keyboard. If the value is within the correct range, another question will be typed. If not, an error message will be printed. The user must enter the correct value. It should be noted that if an error is made on the subscripted variables, all values for that variable must again be entered.

To Rerun The Program Without All New Data

If it is desired to change only a few values of input data and rerun the program, the following procedure must be used:

1. After completion of the initial execution of the program, select the data to be changed and find the "key" numbers for that data from "Schedule of Data Location in File TEXT" given in the manual.

2. Type XEQ to execute the program again.

3. Choose option #8 in answer to question Q-1.

4. Type in 2 in answer to the question on changing file data.

5. Enter the key number in response to both MINLIN and MAXLIN.

6. Type in the new data enclosed in parenthesis, i.e. '36.5' After this control will again be passed to the end of the main program. To rerun the program type XEQ.

7. If more than one value is to be changed before rerunning the program, this process must be executed once for each change desired.

INPUT DATA SHEET

To be used for preparing input data using List of Questions in User's Manual. Letters in parentheses are variable names used in program. Numbers in parentheses give acceptable range of value.

1. Program Option (opt) = _____ (1-8).

2. Data Type (datatp) = _____ (1-3).

3. Building Type (bldgtp) _____ (1-10).

4./3. Occupancy Load Factor (occ) = _____ (-) persons/sq.ft.

Ventilation Load Factor (vent) = _____ (-) cfm/sq.ft.

Lighting Load Factor (lite) = _____ (-) watts/sq.ft.

Power Loads (pow) = _____ (-) watts/sq.ft.

Building Costs (bldgcst) = _____ (-) $/sq.ft.

Mechanical and Electrical Costs (mecst) = _____ (-) $/sq.ft.

Mechanical Equipment Space (mespa) = _____ (-) $/sq.ft.

5. Building Shape Diagram (diag) = _____ (1-2).

6. Building Shape - See Diagram (bldshp) = _____ (1-4).

7. Length of Each Wall (ln(1)) = _____ (1-500) feet

(ln(2)) = _____ (1-500) feet

(ln(3)) = _____ (1-500) feet

(ln(4)) = _____ (1-500) feet

(ln(5)) = _____ (1-500) feet

(ln(6)) = _____ (1-500) feet

8./7. Building Diameter (d) = _____ (1-500) feet

9. Height of Building (h) = _____ (1-500) feet

Note that this should be only the height of the section for which calculations are being made.

10. Number of Stories in Building Section (stor) = _____ (1-60).

11. Building Orientation (bldor) = _____ (1-8).

12. Wall Type (wltp) = _____ (1-6) or U-Factor (wluf) = _____ (0.1-4.0).

13. Glass Type (gltp) = _____ (1-7) or U-Factor (gluf) = _____ (0.1-4.0).

14. Window Type (wntp) = _____ (1-2).

15. Percentage of Glass per Exposure (pctgl(ex)) = _____ (0.01-1.00).

16./15. Window Height (wnh) = _____ (0.5-20) feet

17./15. Window Width (wnw) = _____ (0.5-30) feet

18./15. Number of Windows in Each Exposure (wn(1)) = _____ (-)

(wn(2)) = _____ (-)

(wn(3)) = _____ (-)

(wn(4)) = _____ (-)

(wn(5)) = _____ (-)

(wn(6)) = _____ (-)

19. Door Type (drtp) = _____ (1-7) or U-Factor, druf = _____ (0.01-4.0)

20. Door Height, drh = _____ (5-15) and width, drw = _____ (2-30)

21. Number of Doors per Exposure, dr(1) = _____ (-)

dr(2) = _____ (-)

dr(3) = _____ (-)

dr(4) = _____ (-)

dr(5) = _____ (-)

dr(6) = _____ (-)

22. Roof Type (rftp) = _____ (1-6) or U-Factor (rfuf) = _____ (0.01-4.0).

23. Basement Wall Type (bwtp) = _____ (1-6) or U-Factor (bwuf) = _____ (0.01-4.0).

24. Floor Type (fltp) = _____ (1-6) or U-Factor (fluf) = _____ (0.01-4.0).

25. Latitude of Building Site, See Latitudes Map (lat) = _____(25-60).

DATA ADDRESS SCHEDULE IN FILE "TEXT"

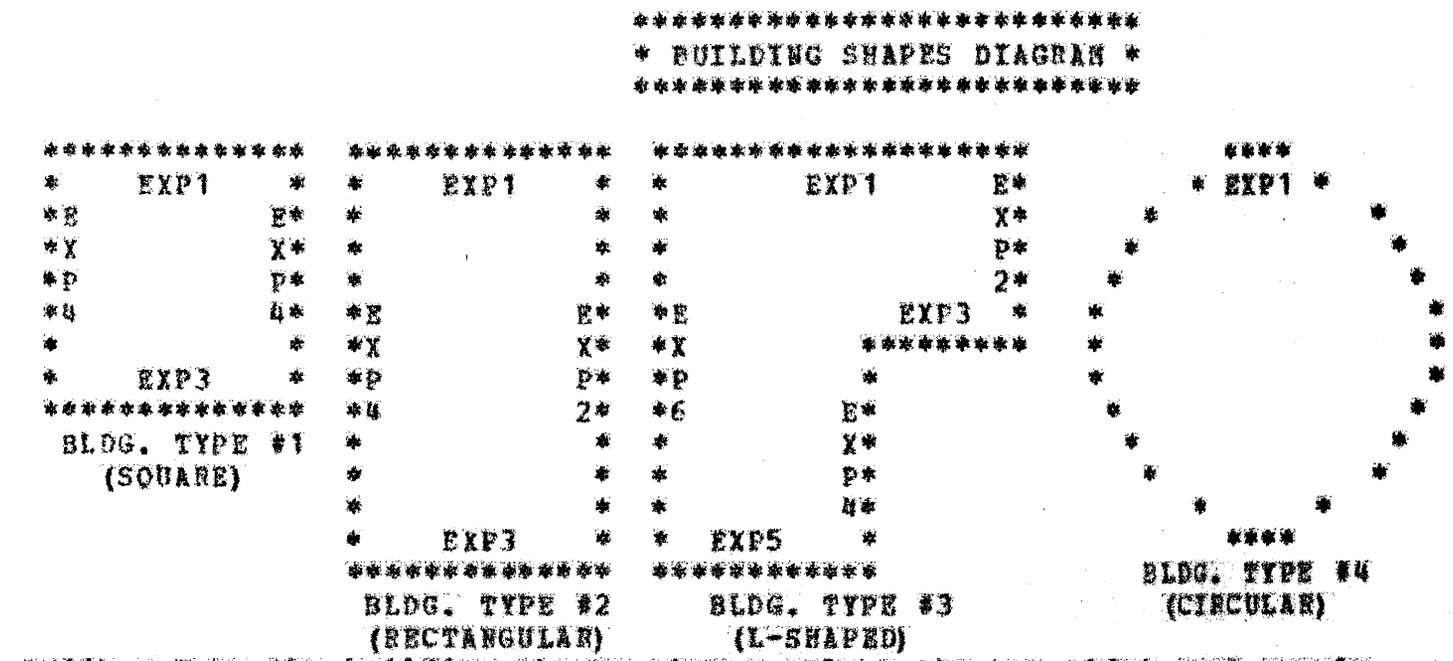
KEY NO.	VARIABLE	SYMBOL	ACCEPT. RANGE
401	Reserved	-	-
402	Program Option	OPT	1-8
403	Building Type	BLDTP	1-10
404	Building Shape	BLDSHP	1-4
405	Occupancy Load Factor	OCC	
406	Ventilation Load Factor	VENT	
407	Lighting Load Factor	LITE	
408	Power Load Factor	POW	
409	Reserved	-	
410	Building Cost	BLDCST	
411	Mech./Elect. Cost	MECST	
412	Mech. Equipment Space	MESPA	
413	Length of Exposure 1	LN(1)	1-500
414	Length of Exposure 2	LN(2)	1-500
415	Length of Exposure 3	LN(3)	1-500
416	Length of Exposure 4	LN(4)	1-500
417	Length of Exposure 5	LN(5)	1-500
418	Length of Exposure 6	LN(6)	1-500
419	Diameter	D	1-500
420	Height	H	1-500
421	Reserved	-	-
422	Reserved	-	-
423	Masimum Number of Exposures	EXMAX	1-8
424	Number of Stories	STOR	1-50
425	Building Orientation	BLDOR	1-8
426	Wall Type	WLTP	1-6
427	Wall U-Factor	WLUF	0.01-4.00
428	Glass Type	GLTP	1-7
429	Glass U-Factor	GLUF	0.01-4.00
430	Window Type	WNTP	1-2
431	Percentage Glass in Exposure 1	PCTGL(1)	0.01-1.00
432	Percentage Glass in Exposure 2	PCTGL(2)	0.01-1.00
433	Percentage Glass in Exposure 3	PCTGL(3)	0.01-1.00
434	Percentage Glass in Exposure 4	PCTGL(4)	0.01-1.00
435	Percentage Glass in Exposure 5	PCTGL(5)	0.01-1.00
436	Percentage Glass in Exposure 6	PCTGL(6)	0.01-1.00
437	Reserved	-	-
438	Reserved	-	-
439	Window Height	WNH	0.05-20.0
440	Window Width	WNW	0.05-30.0

KEY NO.	VARIABLE	SYMBOL	ACCEPT. RANGE
441	Number Windows in Exposure 1	WN(EX)	
442	Number Windows in Exposure 2	WN(EX)	
443	Number Windows in Exposure 3	WN(EX)	
444	Number Windows in Exposure 4	WN(EX)	
445	Number Windows in Exposure 5	WN(EX)	
446	Number Windows in Exposure 6	WN(EX)	
447	Reserved	-	-
448	Door Type	DRTP	1-7
449	Door U-Factor	DRUF	0.01-4.00
450	Door Height	DRH	5.0-15.0
451	Door Width	DRW	2-15.0
452	Number Doors in Exposure 1	DR(EX)	1-10
453	Number Doors in Exposure 2	DR(EX)	
454	Number Doors in Exposure 3	DR(EX)	
455	Number Doors in Exposure 4	DR(EX)	
456	Number Doors in Exposure 5	DR(EX)	
457	Number Doors in Exposure 6	DR(EX)	
458	Reserved	-	-
459	Reserved	-	-
460	Roof Type	RFTP	1-6
461	Roof U-Factor		
462	Basement Wall Type	BMWLTP	1-6
463	Basement Wall U-Factor	BMWLUF	0.01-4.00
464	Floor Type	FLTP	1-6
465	Floor U-Factor	FLUF	0.01-4.00
466	Reserved		
467	Latitude of Site	LAT	25-60
468	Reserved	-	-
469	Reserved	-	-
470	Reserved	-	-
471		TM	
472		TA	
473		P	
474	First Line - Solar Heat Gain Factors	SHGFBL	541-577
475	Last Line - Solar Heat Gain Factors	SHGFEL	552-588
476	Reserved	-	-
477	Reserved	-	-
478	Reserved	-	-
479	Reserved	-	-
480	Net Wall Area - Exposure 1	WLAN(EX)	-
481	Net Wall Area - Exposure 2	WLAN(EX)	-
482	Net Wall Area - Exposure 3	WLAN(EX)	-
483	Net Wall Area - Exposure 4	WLAN(EX)	-
484	Net Wall Area - Exposure 5	WLAN(EX)	-
485	Net Wall Area - Exposure 6	WLAN(EX)	-

KEY NO.	VARIABLE	SYMBOL	ACCEPT. RANGE
486	Total Wall Area	WLAT	-
487	Reserved	-	-
488	Glass Area - Exposure 1	GLA(EX)	-
489	Glass Area - Exposure 2	GLA(EX)	-
490	Glass Area - Exposure 3	GLA(EX)	-
491	Glass Area - Exposure 4	GLA(EX)	-
492	Glass Area - Exposure 5	GLA(EX)	-
493	Glass Area - Exposure 6	GLA(EX)	-
494	Total Glass Area	GLAT	-
495	Reserved	-	-
496	Door Area - Exposure 1	DRA(EX)	-
497	Door Area - Exposure 2	DRA(EX)	-
498	Door Area - Exposure 3	DRA(EX)	-
499	Door Area - Exposure 4	DRA(EX)	-
500	Door Area - Exposure 5	DRA(EX)	-
501	Door Area - Exposure 6	DRA(EX)	-
502	Total Door Area	DRAT	-
503	Reserved	-	-
504	Roof Area	RFA	-
505	Lower Basement Wall Area	BWAL	-
506	Higher Basement Wall Area	BWAH	-
507	Floor Area/Story	FLA	-
508	Total Floor Area	FLAT	-
509	Reserved	-	-
510	Reserved	-	-

```
ENERPAC is a program for building energy design.  It is hoped that this
program will aid you in the design of mechanical systems.  If further
information is required, consult the User's Manual
 Q(1) - This program allows eight options.  After the terminal has typed "opt"
        type in the number of the option to be executed and press the "RETURN"
        key on the right side of the keyboard.  Expect a slight pause.
        1. Energy calculation for the entire building.(Basement omitted)
        2. Calculation for the top floor only.
        3. Calculation for the intermediate floors only.
        4. Calculation for the ground floors only.
        5. Calculation for the floors below grade only.
        6. Calculation for the entire building with question text omitted.
        7. Listing of questions in program.(Requires 10 minutes)
        8. UPDATE option for checking and changing information in data files.
 Q(2) - Data from the last run of the program is now in storage.  Enter the
        number for the type of data you wish to use:
        1. New data, to be supplied during this execution of the program.
        2. Existing data, now in storage, to be used.
        3. Test data, to be generated.
 Q(7) - Enter the number for the building type being used:
        1. School buildings
        2. Classroom buildings
        3. Office buildings
        4. Department stores
        5. Hospitals
        6. Hotels and apartment buildings
        7. Theaters and auditoriums
        8. Libraries or museums
        9. Specialty stores
       10. Building type and load data to be supplied by user
 Q(8) - For a special building type, the following data is required:
        1. Occupancy load(OCC) in persons/sq.ft.
```

```
        2. Ventilation load(VENT) in CFM/sq. ft.
        3. Lighting load(LITE) in watts/sq. ft.
        4. Power loads(POW) in watts/sq. ft.
        5. Infiltration loads(INFIL) in CFM
        6. Building costs(BLDCST) in dollars/sq. ft.
        7. Mechanical and electrical costs(MECST) in dollars/sq. ft.
Enter this data, in the sequence shown above, after the terminal types the
variable name.  Sources for this information are shown in the User's Manual.
Q(9) - The Building Shapes Diagram is shown in the User's Manual or can be
        typed by the terminal.  Type in the number for the method to be used:
        1. Diagram typed by terminal.(Requires approximately 2 minutes)
        2. Diagram known or obtained from Users Manual
```



Q(11) - From the building shapes shown, select the one which most nearly approximates your building:
1. Square
2. Rectangular
3. L-Shaped
4. Circular

Q(12) - Enter the length of each exposure, in feet, for each of the exposures shown in the Building Shapes Diagram, starting with Exposure 1.

Q(13) - Enter the diameter of the building, in feet.
as shown in the diagram, and "h", the height of the building section being considered. Dimensions should be in feet.

Q(14) - Enter the height, in feet, for the section of the building for which the calculations are being made.
being considered. Dimensions should be in feet.
Q(15) - Enter the building dimensions of "d", the diameter of the building, and "h", the height of the building section beinb considered. Dimensions should be in feet.
Q(16) - Enter the number of stories in the building height for which calculations are being made.
Q(17) - Enter the number of the direction which you would like the exposure marked "EXP1" on the Building Shapes Diagram to face:
1. North(To be oriented as shown on diagram)
2. Northeast
3. East
4. Southeast
5. South
6. Southwest
7. West
8. Northwest

Q(19) - Enter the number for the wall type being used:
1. 25/32" Wood siding, building paper sheathing, & 1/4" insulation board U-factor=0.19
2. 4" Face brick veneer, 1/2" insul.bd. sheathing, & 3/8" gypsum board interior U-factor=0.25
3. 4" Face brick, 12" block, & 5/8" plaster U-factor=0.27
4. 4" Face brick, 8" block, U-factor=0.30
5. 8" 140 #/CF Poured concrete U-factor=0.36
6. U-factor supplied by user

Q(20) - Enter the value for the wall u-factor.
Q(21) - Enter the number for the glass type being used:
1. Single flat glass U-factor=1.13
2. Double insulating glass, 1/4" air space U-factor=0.65
3. Single glass with storm windows U-factor=0.56

```
        4. Double insulating glass, 1/2" air space   U-factor=0.48
        5. Future
        6. U-factor supplied by user
        7. No glass used on building
Q(22) - Enter the value for the glass u-factor.
Q(23) - Enter the number for the window type.
        1. Dimensions of windows known
        2. Window dimensions unknown, percentage of glass to be supplied.
Q(24) - Enter the percentage of glass for each exposure shown.
        Value should be integer(no decimal), i. e. 50
Q(25) - Enter the size, in feet, of the window height(WHH).
Q(26) - Enter the size, in feet, of the window width(WHW).
Q(27) - Enter the number of windows in each exposure shown.
Q(28) - Enter the number for the door type being used:
        1. 3/4" Glass door   U-factor=0.55
        2. 1 1/4" Wood door   U-factor=0.51
        3. 1 1/2" Wood door   U-factor=0.49
        4. 1 1/2" Wood door with storm door   U-factor=0.33
        5. 2" Wood door   U-factor=0.43
        6. U-factor and dimensions to be supplied by user
        7. Neglect door heat loss/gain
Q(29) - Enter the value for the door u-factor.
Q(30) - Enter the size, in feet, of the door height(DRH).
Q(31) - Enter the size, in feet, of the door width(DRW).
Q(32) - Enter the number of doors in each exposure shown.
Q(34) - Enter the number for the type of roof being used:
        1. 4" Concrete slab, 1 1/2" insulation board, & 3/4 suspended acoust.
           tile ceiling   U-factor=0.08
        2. Flat metal roof deck, insulation(R=4.17) on top, & 3/4" acoust.
           tile on furring   U-factor=0.13
        3. 4" Conc. slab & insulation(R=4.17) on top   U-factor=0.18
        4. Flat metal roof deck, insulation(R=0.72) on top, 3/8" gypsum bd.,
```

```
          1/2" plaster   U-factor=0.25
       5. Future
       6. U-factor to be supplied by user
Q(35) - Enter the value for the roof u-factor.
Q(36) - Enter the nuaber for the basement wall type:
       1. 8" Poured concrete(80 #/CF)   U-factor=0.10
       2. 6" Poured concrete(140 #/CF)   U-factor=0.75
       3. 8" Poured concrete(140 #/CF)   U-factor=0.67
       4. 10" Poured concrete(140 #/CF)   U-factor=0.61
       5. 12" Poured concrete(140 #/CF)   U-factor=0.55
       6. U-factor to be supplied by user
Q(37) - Enter the value for the basement wall u-factor.
Q(38) - Enter the number for the type of floor being used:
       1. Wood subfloor on joists, 3/8" insul. bd., 1/4" hard board, &
          floor tile   U-factor=0.29
       2. Wood subfloor & 3/4" hardwood floor   U-factor=0.34
       3. 4" Concrete deck and floor tile   U-factor=0.60
       4. 4" Concrete deck, plywood subfloor, and floor tile   U-factor=0.30
       5. 8" Concrete deck, wood subfloor, & 3/4" hardwood floor
          U-factor=0.22
       6. U-factor to be supplied by user
Q(39) - Enter the value for the u-factor.
Q(41) - Latitudes for several local cities are shown.  Enter the latitude,
       to nearest tenth of a degree, from this list or estimate the value
       for other cities.  If value is unknown, consult map in the Users Manual.
       1. Blacksburg, Va.          Lat=37.2
       2. Roanoke, Va.             Lat=37.3
       3. Arlington, Va.           Lat=38.9
       4. Richmond, Va.            Lat=37.5
       5. Norfolk, Va.             Lat=37.0
       6. Bluefield, W. Va.        Lat=37.2
       7. Charleston, W. Va.       Lat=38.4
```

```
 8. Wheeling, W. Va.          Lat=38.5
 9. Winston-salem, N. C.      Lat=36.0
10. Charlotte, N. C.          Lat=35.2
11. Wilmington, N. C.         Lat=34.2
12. Baltimore, Md.            Lat=39.3
13. Salisbury, Md.            Lat=38.3
14. Hagerstown, Md.           Lat=39.6

Enter the number for the UPDATE option desired:
1. Read file "TEXT"
2. Change file "TEXT"
```

APPENDIX II

PROGRAM LISTING OF ENERPAC

```
 1.              DECLARE text FILE ENV( REGIONAL(1) F(80) OLD  ) ;
 2.              DECLARE input ENTRY  EXT KEY(arch);
 3.              DECLARE result ENTRY  EXT KEY(arch);
 4.              DECLARE update ENTRY  EXT KEY(arch);
 5.              DECLARE convrt ENTRY  EXT KEY(arch);
 6.              DECLARE summry ENTRY  EXT KEY(arch);
 7.              DECLARE calc ENTRY  EXT KEY(arch);
 8.              DECLARE test ENTRY  EXT KEY(arch);
 9.              DECLARE output ENTRY  EXT KEY(arch);
10.              DECLARE cx CHAR(80);
11.              DECLARE line CHAR(80);
12.              OPEN FILE(text) DIRECT UPDATE TITLE('harry.moate.a30722') ;
13.     q1:      DO I=1 TO 14;
14.              READ FILE(text) INTO(line) KEY(I) ;
15.              PUT LIST(line);
16.              END q1;
17.     a1:      GET LIST(opt);
18.              IF opt>=1&opt<=8 THEN GO TO r1;
19.              PUT LIST('INCORRECT INPUT.  Reenter number from 1 thru 8');
20.              GO TO a1;
21.     r1:      IF opt=7|opt=8 THEN CLOSE FILE(text) ;
22.              IF opt=7|opt=8 THEN CALL update(opt);
23.              IF opt=7|opt=8 THEN GO TO r100;
24.              IF opt=6 THEN GO TO a2;
25.     q2:      DO I=15 TO 19;
26.              READ FILE(text) INTO(line) KEY(I) ;
27.              PUT LIST(line);
28.              END q2;
29.     a2:      GET LIST(datatp);
30.              IF datatp>=1&datatp<=3 THEN GO TO r2;
31.              PUT LIST('INCORRECT INPUT.  Reenter number from 1 thru 3');
32.              GO TO a2;
33.     r2:      IF datatp=2|datatp=3 THEN CLOSE FILE(text) ;
34.              IF datatp=2 THEN GO TO r7;
35.              IF datatp=3 THEN CALL test;
36.              IF datatp=3 THEN GO TO r6;
37.              cx='00';
38.              DO I=401 TO 510;
39.              REWRITE FILE(text) FROM(cx) KEY(I) ;
40.              END ;
41.              CLOSE FILE(text) ;
42.     r5:      CALL input(opt);
43.     r6:      CALL convrt(opt);
44.     r7:      CALL summry;
45.     r8:      CALL output(opt);
46.     r9:      CALL calc(opt);
47.     r100:    PUT LIST('END OF PROGRAM.  TYPE IN "LOGOUT" AND THEN "LOGOFF"');
```

```
 1.     input:  PROCEDURE (opt);
 2.             opt=1;
 3.             DECLARE text FILE ENV( REGIONAL(1) F(80) OLD  ) ;
 4.             DECLARE b1(40)  DEC(3), e1(40)  DEC(3);
 5.             DECLARE b11 CHAR(80), b12 CHAR(80), e11 CHAR(80), e12 CHAR(80);
 6.             DECLARE bb1 CHAR(5), ee1 CHAR(5);
 7.             DECLARE line CHAR(80), ln(6) , pctg1(6) , wn(6) , dr(6) , valuec CHAR(80), value3(6) , valued CHAR(80);
 8.             OPEN FILE(text) DIRECT UPDATE TITLE('HARRY.MOATE.A30722') ;
 9.             b11='00101500000000000002003104204607608308709100009910200012213213414314514915l';
10.             e11='01401900000000000003004004506508108408709300010011100013013214114314715015l';
11.             b12='15215315517517717817918118519619720421021922l';
12.             e12='15215316217517717817918319419620320421821923 9';
13.             ll=1;
14.             DO i=1 TO 40;
15.             bb1=substr(b11,ll,3);
16.             ee1=substr(e11,ll,3);
17.             b1(i)=float(bb1);
18.             e1(i)=float(ee1);
19.             ll=ll+3;
20.             IF i^=25 THEN GO TO r1;
21.             ll=1;
22.             b11=b12;
23.             e11=e12;
24.     r1:     END ;
25.             valuec=char(opt);
26.             REWRITE FILE(text) FROM(valuec) KEY(402) ;
27.             CALL quest2(7,1,10,'bldtp',403,bldtp);
28.             IF bldtp^=10 THEN GO TO q9;
29.             IF opt=6 THEN GO TO r8;
30.             CALL quest1(8);
31.     r8:     GET LIST(occ,vent,lite,pow,bldcst,mecst,mespa);
32.             valuec=char(occ);
33.             REWRITE FILE(text) FROM(valuec) KEY(405) ;
34.             valuec=char(vent);
35.             REWRITE FILE(text) FROM(valuec) KEY(406) ;
36.             valuec=char(lite);
37.             REWRITE FILE(text) FROM(valuec) KEY(407) ;
38.             valuec=char(pow);
39.             REWRITE FILE(text) FROM(valuec) KEY(408) ;
40.             valuec=char(bldcst);
41.             REWRITE FILE(text) FROM(valuec) KEY(410) ;
42.             valuec=char(mecst);
43.             REWRITE FILE(text) FROM(valuec) KEY(411) ;
44.             valuec=char(mespa);
45.             REWRITE FILE(text) FROM(valuec) KEY(412) ;
46.     q9:     CALL quest2(9,1,2,'diag',0,diag);
47.             IF diag=2 THEN GO TO q11;
48.             CALL quest1(10);
49.     q11:    CALL quest2(11,1,4,'bldshp',404,bldshp);
50.             IF bldshp=1|bldshp=2 THEN exmax=4;
```

```
51.             IF bldshp=3 THEN exmax=6;
52.             IF bldshp=4 THEN exmax=1;
53.             valuec=char(exmax);
54.             REWRITE FILE(text) FROM(valuec) KEY(423) ;
55.             IF bldshp=4 THEN GO TO q13;
56.             CALL quest3(12,1,500,'length',413);
57.             GO TO q14;
58.     q13:    CALL quest2(13,1,500,'diameter',419,d);
59.     q14:    CALL quest2(14,1,500,'height',420,h);
60.             CALL quest2(16,1,50,'stories',424,stor);
61.             CALL quest2(17,1,8,'bldor',425,bldor);
62.             IF opt=5 THEN GO TO q36;
63.             CALL quest2(19,1,6,'wltp',426,wltp);
64.             IF wltp=6 THEN CALL quest2(20,.01,4,'wluf',427,wluf);
65.             CALL quest2(21,1,7,'gltp',428,gltp);
66.             IF gltp=7 THEN GO TO q28;
67.             IF gltp=6 THEN CALL quest2(22,.01,4,'gluf',429,gluf);
68.             CALL quest2(23,1,2,'wntp',430,wntp);
69.             IF wntp=1 THEN GO TO q25;
70.             CALL quest3(24,.01,1,'pctgl',431);
71.             GO TO q28;
72.     q25:    CALL quest2(25,.5,20,'wnh',439,wnh);
73.             CALL quest2(26,.5,30,'wnw',440,wnw);
74.             CALL quest3(27,.5,10,'wn/ex',441);
75.     q28:    CALL quest2(28,1,7,'drtp',448,drtp);
76.             IF drtp=7 THEN GO TO q34;
77.             IF drtp=6 THEN CALL quest2(29,.01,4,'druf',449,druf);
78.             CALL quest2(30,5,15,'drh',450,drh);
79.             CALL quest2(31,2,30,'drw',451,drw);
80.             CALL quest3(32,0,10,'dr/ex',452);
81.             IF opt=3 THEN GO TO q40;
82.             IF opt=4 THEN GO TO q38;
83.     q34:    CALL quest2(34,1,6,'rftp',460,rftp);
84.             IF rftp=6 THEN CALL quest2(35,.01,4,'rfuf',461,rfuf);
85.             IF opt=2 THEN GO TO q40;
86.             IF opt¬=5 THEN GO TO q38;
87.     q36:    CALL quest2(36,1,6,'bmwltp',462,bmwltp);
88.             IF bmwltp=6 THEN CALL quest2(37,.01,4,'bmwluf',463,bmwluf);
89.     q38:    CALL quest2(38,1,6,'fltp',464,fltp);
90.             IF fltp=6 THEN CALL quest2(39,.01,4,'fluf',465,fluf);
91.     q40:    CALL quest2(40,25,60,'lat',467,lat);
92.             CLOSE FILE(text) ;
93.     quest1: PROCEDURE (n1);
94.             DO i=b1(n1) TO e1(n1);
95.             OPEN FILE(text) DIRECT UPDATE TITLE('harry.moate.a30722') ;
96.             READ FILE(text) INTO(line) KEY(i) ;
97.             PUT LIST(line);
98.             END ;
99.     quest2: PROCEDURE (n2,l12,u12,v2,dfn2,value);
100.            IF opt=6 THEN GO TO as2;
```

```
101.              CALL quest1(n2);
102.              GO TO as2a;
103.      as2:    PUT LIST(vn2);
104.      as2a:   GET LIST(value);
105.              IF value>=l12&value<=u12 THEN GO TO rs2;
106.              PUT EDIT('INCORRECT INPUT.  Reenter number from ',l12,' thru ',u12)(A(37),F(5,2),A(6),F(5,2));
107.              GO TO as2;
108.      rs2:    IF dfn2=0 THEN GO TO rs2a;
109.              valuec=char(value);
110.              OPEN FILE(text) DIRECT UPDATE TITLE('harry.moate.a30722') ;
111.              REWRITE FILE(text) FROM(valuec) KEY(dfn2) ;
112.      rs2a:   RETURN ;
113.              END quest2;
114.      quest3: PROCEDURE (n3,l13,u13,vn3,dfn3);
115.              IF opt=6 THEN GO TO as3;
116.              CALL quest1(n3);
117.      as3:    PUT LIST(vn3);
118.              dfn3a=dfn3;
119.      as3a:   DO ex=1 TO exmax;
120.              GET LIST(value3(ex));
121.              IF value3(ex)>=l13&value3(ex)<=u13 THEN GO TO rs3;
122.              PUT EDIT('INCORRECT INPUT.  Reenter number from ',l13,' thru ',u13)(A(37),F(5,2),A(6),F(5,2));
123.              ex=ex-1;
124.              dfn3a=dfn3a-1;
125.              GO TO as3a;
126.      rs3:    valued=char(value3(ex));
127.              OPEN FILE(text) DIRECT UPDATE TITLE('harry.moate.a30722') ;
128.              REWRITE FILE(text) FROM(valued) KEY(dfn3a) ;
129.              dfn3a=dfn3a+1;
130.              END as3a;
131.              RETURN ;
132.              END quest3;
133.              CLOSE FILE(text) ;
134.              END input;
```

```
test:   PROCEDURE ;
        DECLARE text FILE ENV( REGIONAL(1) F(80) OLD  ) ;
        OPEN FILE(text) DIRECT UPDATE TITLE('harry.moate.a30722') ;
        DECLARE a(30) , namec CHAR(80);
        DECLARE ln(6) , pctgl(6) , dr(6) ;
        opt,a(1)=1;
        bldtp,a(2)=1;
        bldshp,a(3)=2;
        ln(1),a(4)=50;
        ln(2),a(5)=100;
        ln(3),a(6)=50;
        ln(4),a(7)=100;
        h,a(8)=50;
        exmax,a(9)=4;
        stor,a(10)=3;
        bldor,a(11)=1;
        wltp,a(12)=1;
        gltp,a(13)=1;
        wntp,a(14)=2;
        pctgl(1),a(15),a(16),a(17),a(18)=.5;
        drtp,a(19)=1;
        drh,a(20),drw,a(21)=10;
        dr(1),a(22),a(23),a(24),a(25)=1;
        rftp,a(26)=1;
        bmwltp,a(27)=1;
        fltp,a(28)=1;
        lat,a(29)=37.2;
        n=1;
        DO i=402 TO 404,413 TO 416,420,423 TO 426,428,430 TO 434,448,450 TO 455,460,462,464,467;
        name=a(n);
        namec=char(name);
        REWRITE FILE(text) FROM(namec) KEY(1) ;
        n=n+1;
        END ;
        PUT LIST('END OF SUBROUTINE "TEST"');
        CLOSE FILE(text) ;
        END test;
```

```
convrt: PROCEDURE (opt);
        DECLARE text FILE ENV( REGIONAL(1) F(80) OLD  ) ;
        OPEN FILE(text) DIRECT UPDATE TITLE('harry.moate.a30722') ;
        DECLARE strg CHAR(80), str(11)  CHAR(80);
        DECLARE occc CHAR(80), ventc CHAR(80), lltec CHAR(80), powc CHAR(80), bldcsc CHAR(80);
        DECLARE mecstc CHAR(80), mespac CHAR(80);
        DECLARE wlufc CHAR(80), glufc CHAR(80), drufc CHAR(80), rfufc CHAR(80), bmwluc CHAR(80), flufc CHAR(80)
;
        DECLARE tmc CHAR(80), tac CHAR(80), pc CHAR(80);
        DECLARE state CHAR(80);
        I=1;
        DO I=402,423,426,428,448,460,462,464,467;
        READ FILE(text) INTO(strg) KEY(I) ;
        str(I)=strg;
        I=I+1;
        END ;
        INCLUDE STRING str ;
        GET LIST(bldtp,exmax,wltp,gltp,drtp,rftp,bmwltp,fltp,lat);
        INCLUDE * ;
c1:     IF bldtp^=1 THEN GO TO c2;
        occ=35;
        vent=30;
        llte=10;
        pow=5;
        ;
        bldcst=23.25;
        mecst=7.2;
        mespa=.045;
c2:     IF bldtp^=2 THEN GO TO c3;
        occ=20;
        vent=25;
        llte=12;
        pow=6;
        bldcst=29.85;
        mecst=8.6;
        ;
        mespa=.043;
c3:     IF bldtp^=3 THEN GO TO c4;
        occ=105;
        vent=20;
        llte=17;
        pow=5;
        bldcst=25.75;
        mecst=7.6;
        mespa=.025;
c4:     IF bldtp^=4 THEN GO TO c5;
        occ=40;
        vent=15;
        llte=14;
        pow=5;
        bldcst=13.45;
```

```
51.             mecst=4.7;
52.             ;
53.             mespa=.045;
54.     c5:     IF bldtp~=5 THEN GO TO c6;
55.             occ=125;
56.             ;
57.             vent=25;
58.             llte=12;
59.             pow=5;
60.             bldcst=38;
61.             mecst=13.85;
62.             mespa=.055;
63.     c6:     IF bldtp~=6 THEN GO TO c7;
64.             occ=170;
65.             vent=25;
66.             llte=3;
67.             pow=4;
68.             bldcst=21.35;
69.             mecst=6.75;
70.             mespa=.02;
71.     c7:     IF bldtp~=7 THEN GO TO c8;
72.             occ=8;
73.             vent=10;
74.             llte=5;
75.             pow=3;
76.             bldcst=23;
77.             mecst=5.85;
78.             mespa=.04;
79.     c8:     IF bldtp~=8 THEN GO TO c9;
80.             occ=60;
81.             vent=10;
82.             llte=12;
83.             pow=3;
84.             bldcst=26.7;
85.             mecst=8.15;
86.             mespa=.044;
87.     c9:     IF bldtp~=9 THEN GO TO c10;
88.             occ=90;
89.             vent=10;
90.             llte=14;
91.             pow=6;
92.             bldcst=14.4;
93.             mecst=3.85;
94.             mespa=.015;
95.     c10:    ;
96.             IF bldtp=10 THEN GO TO c11;
97.             occc=char(occ);
98.             ventc=char(vent);
99.             lltec=char(llte);
100.            powc=char(pow);
```

```
101.              bldcsc=char(bldcst);
102.              mecstc=char(mecst);
103.              mespac=char(mespa);
104.              REWRITE FILE(text) FROM(occc) KEY(405) ;
105.              REWRITE FILE(text) FROM(ventc) KEY(406) ;
106.              REWRITE FILE(text) FROM(lltec) KEY(407) ;
107.              REWRITE FILE(text) FROM(powc) KEY(408) ;
108.              REWRITE FILE(text) FROM(bldcsc) KEY(410) ;
109.              REWRITE FILE(text) FROM(mecstc) KEY(411) ;
110.              REWRITE FILE(text) FROM(mespac) KEY(412) ;
111.     c11:     IF wltp¬=1 THEN GO TO c12;
112.              wluf=.19;
113.     c12:     IF wltp¬=2 THEN GO TO c13;
114.              wluf=.25;
115.     c13:     IF wltp¬=3 THEN GO TO c14;
116.              wluf=27;
117.     c14:     IF wltp¬=4 THEN GO TO c15;
118.              wluf=.3;
119.     c15:     IF wltp¬=5 THEN GO TO c16;
120.              wluf=.36;
121.     c16:     IF wltp=6 THEN GO TO c20;
122.              wlufc=char(wluf);
123.              REWRITE FILE(text) FROM(wlufc) KEY(427) ;
124.     c20:     IF gltp¬=1 THEN GO TO c21;
125.              gluf=1.13;
126.     c21:     IF gltp¬=2 THEN GO TO c22;
127.              gluf=.65;
128.     c22:     IF gltp¬=3 THEN GO TO c23;
129.              gluf=.56;
130.     c23:     IF gltp¬=4 THEN GO TO c24;
131.              gluf=.48;
132.     c24:     IF gltp=5 THEN gluf=0;
133.     c25:     IF gltp=6 THEN GO TO c30;
134.              glufc=char(gluf);
135.              REWRITE FILE(text) FROM(glufc) KEY(429) ;
136.     c30:     IF drtp¬=1 THEN GO TO c31;
137.              druf=.55;
138.     c31:     IF drtp¬=2 THEN GO TO c32;
139.              druf=.55;
140.     c32:     IF drtp¬=3 THEN GO TO c33;
141.              druf=.49;
142.     c33:     IF drtp¬=4 THEN GO TO c34;
143.              druf=.33;
144.     c34:     IF drtp¬=5 THEN GO TO c35;
145.              druf=.43;
146.     c35:     IF drtp=6 THEN GO TO c40;
147.              drufc=char(druf);
148.              REWRITE FILE(text) FROM(drufc) KEY(449) ;
149.     c40:     IF rftp¬=1 THEN GO TO c41;
150.              rfuf=.08;
```

```
151.    c41:    IF rftp~=2 THEN GO TO c42;
152.            rfuf=.13;
153.    c42:    IF rftp~=3 THEN GO TO c43;
154.            rfuf=.18;
155.    c43:    IF rftp~=4 THEN GO TO c44;
156.            rfuf=.25;
157.    c44:    IF rftp~=5 THEN GO TO c45;
158.            rfuf=0;
159.    c45:    IF rftp=6 THEN GO TO c50;
160.            rfufc=char(rfuf);
161.            REWRITE FILE(text) FROM(rfufc) KEY(461) ;
162.    c50:    IF bmwltp=1 THEN bmwluf=.1;
163.            IF bmwltp=2 THEN bmwluf=.75;
164.            IF bmwltp=3 THEN bmwluf=.67;
165.            IF bmwltp=4 THEN bmwluf=.61;
166.            IF bmwltp=5 THEN bmwluf=.55;
167.            IF bmwltp=6 THEN GO TO c60;
168.            bmwluc=char(bmwluf);
169.            REWRITE FILE(text) FROM(bmwluc) KEY(463) ;
170.    c60:    IF fltp=1 THEN fluf=.29;
171.            IF fltp=2 THEN fluf=.34;
172.            IF fltp=3 THEN fluf=.6;
173.            IF fltp=4 THEN fluf=.3;
174.            IF fltp=5 THEN fluf=.22;
175.            IF fltp=6 THEN GO TO c70;
176.            flufc=char(fluf);
177.            REWRITE FILE(text) FROM(drufc) KEY(449) ;
178.    c70:    IF lat>29 THEN GO TO c72;
179.            tm=74.4;
180.            ta=9.5;
181.            p=4.22;
182.            GO TO c75;
183.    c72:    IF lat>36 THEN GO TO c73;
184.            tm=66;
185.            ta=17.7;
186.            p=4.2;
187.            GO TO c75;
188.    c73:    IF lat>42.5 THEN GO TO c74;
189.            tm=52.6;
190.            ta=23.7;
191.            p=4.18;
192.            GO TO c75;
193.    c74:    tm=44.4;
194.            ta=28;
195.            p=4.17;
196.    c75:    tmc=char(tm);
197.            tac=char(ta);
198.            pc=char(p);
199.            REWRITE FILE(text) FROM(tmc) KEY(471) ;
200.            REWRITE FILE(text) FROM(tac) KEY(472) ;
```

```
201.            REWRITE FILE(text) FROM(pc) KEY(473) ;
202.            CLOSE FILE(text) ;
203.            PUT LIST('END OF SUBROUTINE "CONVRT"');
204.            END convrt;
```

```
summry: PROCEDURE ;
        DECLARE text FILE ENV( REGIONAL(1) F(80) OLD  ) ;
        DECLARE str(25)  CHAR(80), strg CHAR(80), ln(6) ;
        DECLARE lat DEC(5.1);
        OPEN FILE(text) DIRECT UPDATE TITLE('harry.moate.a30722') ;
        no=1;
        DO i=402 TO 404,413 TO 420,424,425,427,429,449,461,463,465,467;
        READ FILE(text) INTO(strg) KEY(i) ;
        str(no)=strg;
        no=no+1;
        END ;
        INCLUDE STRING str ;
        GET LIST(opt,bldtp,bldshp,ln(1),ln(2),ln(3),ln(4),ln(5),ln(6),d,h,stor,bldor,wluf,gluf,druf,rfuf,bmwluf
,fluf,lat);
        INCLUDE * ;
        PUT EDIT('****************')(SKIP(5),COLUMN(15),A(15));
        PUT EDIT('*INPUT SUMMARY*')(COLUMN(15),A(15));
        PUT EDIT('****************')(COLUMN(15),A(15));
        PUT EDIT('OPTION = ',opt)(SKIP(2),A(9),F(4));
        PUT EDIT('BUILDING TYPE = ',bldtp)(A(13),F(5));
        PUT EDIT('BUILDING SHAPE = ',bldshp)(A(17),F(4));
        PUT EDIT('BUILDING DIMENSIONS:')(A(20));
        IF bldshp=4 THEN GO TO r2;
        IF bldshp=4 THEN GO TO r2;
        PUT EDIT('LENGTH(1) = ',ln(1),'  FEET')(X(5),A(12),F(5),A(7));
        PUT EDIT('LENGTH(2) = ',ln(2),'  FEET')(X(5),A(12),F(5),A(7));
        PUT EDIT('LENGTH(3) = ',ln(3),'  FEET')(X(5),A(12),F(5),A(7));
        PUT EDIT('LENGTH(4) = ',ln(4),'  FEET')(X(5),A(12),F(5),A(7));
        IF bldshp¬=3 THEN GO TO r3;
        PUT EDIT('LENGTH(5) = ',ln(5),'  FEET')(X(5),A(12),F(5),A(7));
        PUT EDIT('LENGTH(6) = ',ln(6),'  FEET')(X(5),A(12),F(5),A(7));
        GO TO r3;
r2:     PUT EDIT('DIAMETER = ',d,'  FEET')(X(5),A(11),F(7),A(7));
r3:     PUT EDIT('HEIGHT = ',h,'  FEET')(X(5),A(9),F(8),A(7));
        PUT EDIT('NUMBER OF STORIES = ',stor)(A(20),F(5));
        PUT EDIT('ORIENTATION OF EXPOSURE 1 = ',bldor)(A(28),F(4));
        PUT EDIT('U-FACTORS:')(SKIP,COLUMN(12),A(10));
        PUT EDIT('WALLS',wluf)(A(5),X(7),F(5,2));
        PUT EDIT('GLASS',gluf)(A(5),X(7),F(5,2));
        PUT EDIT('DOOR',druf)(A(4),X(8),F(5,2));
        PUT EDIT('ROOF',rfuf)(A(4),X(8),F(5,2));
        PUT EDIT('FLOOR',fluf)(A(5),X(7),F(5,2));
        PUT EDIT('BSMT. WALL',bmwluf)(A(10),X(2),F(5,2));
        PUT EDIT('LATITUDE OF BUILDING SITE = ',lat,'  DEG. F.')(SKIP,A(28),F(5,1),A(5));
        PUT EDIT('INSIDE DESIGN TEMPERATURE = 75 DEG. F.')(A(38));
        CLOSE FILE(text) ;
        END summry;
```

```
output: PROCEDURE (opt);
        DECLARE text FILE ENV( REGIONAL(1) F(80) OLD  ) ;
        DECLARE strg CHAR(80), str1(12)  CHAR(80), str2(16)  CHAR(80), str3(9)  CHAR(80);
        DECLARE var1c CHAR(80), var2c CHAR(80), var3c CHAR(80), var4c CHAR(80), var5c CHAR(80), var6c CHAR(80),
var7c CHAR(80);
        DECLARE ln(6) , pctgl(6) , wn(6) , dr(6) ;
        DECLARE wla(6) , gla(6) , dra(6) , wlan(6) , bmwltp(6) ;
        OPEN FILE(text) DIRECT UPDATE TITLE('harry.moate.a30722') ;
        no=1;
        DO i=404,413 TO 418,419,420,423 TO 425,428,430 TO 436,439 TO 446,448,450 TO 457;
        READ FILE(text) INTO(strg) KEY(i) ;
        IF i>425 THEN GO TO 12;
        str1(no)=strg;
        GO TO 14;
12:     IF i>446 THEN GO TO 13;
        IF i=428 THEN no=1;
        str2(no)=strg;
        GO TO 14;
13:     IF i=448 THEN no=1;
        str3(no)=strg;
14:     no=no+1;
        END ;
        INCLUDE STRING str1 ;
        GET LIST(bldshp,ln(1),ln(2),ln(3),ln(4),ln(5),ln(6),d,h,exmax,stor,bldor);
        INCLUDE * ;
        INCLUDE STRING str2 ;
        GET LIST(gltp,wntp,pctgl(1),pctgl(2),pctgl(3),pctgl(4),pctgl(5),pctgl(6),wnh,wnw,wn(1),wn(2),wn(3),wn(4
),wn(5),wn(6));
        ;
        INCLUDE * ;
        INCLUDE STRING str3 ;
        exmax=4;
        GET LIST(drtp,drh,drw,dr(1),dr(2),dr(3),dr(4),dr(5),dr(6));
        INCLUDE * ;
        IF opt=5 THEN GO TO 185;
        IF bldshp=4 THEN GO TO 170;
        DO ex=1 TO exmax;
        wla(ex)=ln(ex)*h;
        PUT LIST(wla(ex));
        END ;
        GO TO 171;
170:    wla(1)=3.1416*d*h;
171:    glat=0;
        IF gltp¬=7 THEN GO TO 172;
        DO ex=1 TO exmax;
        gla(ex)=0;
        END ;
        GO TO 174;
172:    IF wntp=1 THEN GO TO 173;
        DO ex=1 TO exmax;
        gla(ex)=wla(ex)*pctgl(ex);
        PUT LIST(gla(ex));
```

```
51.             glat=glat+gla(ex);
52.             END ;
53.             GO TO 174;
54.     173:    wna=wnh*wnw;
55.             DO ex=1 TO exmax;
56.             gla(ex)=wna*wn(ex);
57.             glat=glat+gla(ex);
58.             END ;
59.     174:    drat=0;
60.             IF drtp¬=7 THEN GO TO 175;
61.             DO ex=1 TO exmax;
62.             dr(ex)=0;
63.             END ;
64.             GO TO 187;
65.     175:    adr=drh*drw;
66.             DO ex=1 TO exmax;
67.             dra(ex)=adr*dr(ex);
68.             PUT LIST(dra(ex));
69.             drat=drat+drat+dra(ex);
70.             END ;
71.     187:    wlat=0;
72.             DO ex=1 TO exmax;
73.             wlan(ex)=wla(ex)-gla(ex)-dra(ex);
74.             PUT LIST(ex,wla(ex),gla(ex),dra(ex),wlan(ex));
75.             wlat=wlat+wlan(ex);
76.             END ;
77.             IF opt=3|opt=4|opt=5 THEN GO TO 178;
78.             IF bldshp¬=1|bldshp¬=2 THEN GO TO 176;
79.             rfa=ln(1)*ln(2);
80.             PUT LIST(rfa);
81.     176:    IF bldshp¬=3 THEN GO TO 177;
82.             rfa=ln(2)*ln(3)+ln(4)*ln(5);
83.     177:    IF bldshp¬=4 THEN GO TO 179;
84.             rfa=3.1416*(d/2)**2;
85.             GO TO 179;
86.     178:    rfa=0;
87.     179:    IF opt¬=5 THEN GO TO 186;
88.             rfa=0;
89.             IF bldshp=4 THEN GO TO 180;
90.             lnt=0;
91.             DO ex=1 TO exmax;
92.             lnt=lnt+ln(ex);
93.             END ;
94.             GO TO 181;
95.     180:    lnt=3.1416*d;
96.     181:    bmwlah=lnt*8;
97.             bmwlal=lnt*(h-8);
98.             bmwlat=bmwlah+bmwlal;
99.             GO TO 182;
100.    186:    bmwlal=0;
```

```
101.              bmwlah=0;
102.     182:     IF bldshp=1|bldshp=2 THEN fla=ln(1)*ln(2);
103.     183:     IF bldshp=3 THEN fla=ln(2)*ln(3)+ln(4)*ln(5);
104.     184:     IF bldshp=4 THEN fla=3.1414*(d/2)**2;
105.     185:     flat=fla*stor;
106.              IF opt=2|opt=3 THEN fla=0;
107.              l=480;
108.              dummy=0;
109.              CALL sub1(wlan(1),wlan(2),wlan(3),wlan(4),wlan(5),wlan(6),wlat);
110.              CALL sub1(gla(1),gla(2),gla(3),gla(4),gla(5),gla(6),glat);
111.              CALL sub1(dra(1),dra(2),dra(3),dra(4),dra(5),dra(6),drat);
112.              CALL sub1(rfa,bmwlal,bmwlah,fla,flat,dummy,dummy);
113.     sub1:    PROCEDURE (var1,var2,var3,var4,var5,var6,var7);
114.              PUT LIST(var1,var2,var3,var4,var5,var6,var7);
115.              var1c=char(var1);
116.              REWRITE FILE(text) FROM(var1c) KEY(1) ;
117.              l=l+1;
118.              var2c=char(var2);
119.              REWRITE FILE(text) FROM(var2c) KEY(1) ;
120.              l=l+1;
121.              var3c=char(var3);
122.              REWRITE FILE(text) FROM(var3c) KEY(1) ;
123.              l=l+1;
124.              var4c=char(var4);
125.              REWRITE FILE(text) FROM(var4c) KEY(1) ;
126.              l=l+1;
127.              var5c=char(var5);
128.              REWRITE FILE(text) FROM(var5c) KEY(1) ;
129.              l=l+1;
130.              var6c=char(var6);
131.              REWRITE FILE(text) FROM(var6c) KEY(1) ;
132.              l=l+1;
133.              var7c=char(var7);
134.              REWRITE FILE(text) FROM(var7c) KEY(1) ;
135.              l=l+2;
136.              END sub1;
137.              CLOSE FILE(text) ;
138.              PUT LIST('END OF SUBROUTINE "OUTPUT"');
139.              END output;
```

```
   1.    calc:   PROCEDURE (opt);
   2.            DECLARE text FILE ENV( REGIONAL(1) F(80) OLD  ) ;
   3.            DECLARE strg CHAR(80), str1(19)  CHAR(80), str2(10)  CHAR(80), oat(12) , delt(12) , grndt(12) , trwl(12
) , trgl(12) , trdr(12) , trrf(12) ;
   4.            DECLARE trbwh(12) , trbwl(12) , trbw(12) , trfl(12) , tr(12) , ventld(12) , llteld(12) , occlds(12) , o
ccldl(12) , occld(12) , load(12) ;
   5.            DECLARE sl(12) , tslgl(12) ;
   6.            cool,heat=0;
   7.            OPEN FILE(text) DIRECT UPDATE TITLE('harry.moate.a30722') ;
   8.            no=1;
   9.            DO l=405 TO 412,427,429,449,461,463,465,471 TO 473,486,494,502 TO 508;
  10.            READ FILE(text) INTO(strg) KEY(l) ;
  11.            IF l>=486 THEN GO TO c2;
  12.            str1(no)=strg;
  13.            GO TO c3;
  14.    c2:     IF l=486 THEN no=1;
  15.            str2(no)=strg;
  16.    c3:     no=no+1;
  17.            END ;
  18.    c4:     INCLUDE STRING str1 ;
  19.            GET LIST(occ,vent,llte,pow,no9,hldcst,mecst,mespa,wluf,gluf,druf,rfuf,bwuf,fluf,tm,ta,p);
  20.            INCLUDE * ;
  21.    c5:     INCLUDE STRING str2 ;
  22.            GET LIST(wlat,glat,drat,no503,rfa,bwal,bwah,fla,flat);
  23.            INCLUDE * ;
  24.            DO mo=1 TO 12;
  25.            oat(mo)=tm+ta*sind((mo-p)*30);
  26.            delt(mo)=oat(mo)-75;
  27.            /*     NOTE! THIS WILL MAKE HEAT LOSS NEGATIVE     */;
  28.            IF oat(mo)<-30 THEN grndt(mo)=-35;
  29.            IF oat(mo)>-30&oat(mo)<-20 THEN grndt(mo)=-30;
  30.            IF oat(mo)>-20&oat(mo)<-10 THEN grndt(mo)=-25;
  31.            IF oat(mo)>-10&oat(mo)<0 THEN grndt(mo)=-20;
  32.            IF oat(mo)>0&oat(mo)<10 THEN grndt(mo)=-15;
  33.            IF oat(mo)>10 THEN grndt(mo)=-10;
  34.            END ;
  35.            ttr,ttrwl,ttrgl,ttrdr,ttrrf,ttrbwl,ttrbwh,ttrbw,ttrfl,tvent,tllte,tocc,tload=0;
  36.            occno=flat/occ;
  37.            DO mo=1 TO 12;
  38.            trwl(mo)=wluf*wlat*delt(mo);
  39.            ttrwl=ttrwl+trwl(mo);
  40.            trgl(mo)=gluf*glat*delt(mo);
  41.            ttrgl=ttrgl+trgl(mo);
  42.            trdr(mo)=druf*drat*delt(mo);
  43.            ttrdr=ttrdr+trdr(mo);
  44.            trrf(mo)=rfuf*rfa*delt(mo);
  45.            ttrrf=ttrrf+trrf(mo);
  46.            trbwh(mo)=.08*bwah*delt(mo);
  47.            ttrbwh=ttrbwh+trbwh(mo);
  48.            trbwl(mo)=.08*bwal*grndt(mo);
  49.            ttrbwl=ttrbwl+trbwl(mo);
  50.            trbw(mo)=trbwh(mo)+trbwl(mo);
```

```
51.             trfl(mo)=.05*fla*grndt(mo);
52.             ttrfl=ttrfl+trfl(mo);
53.             tr(mo)=trwl(mo)+trgl(mo)+trdr(mo)+trrf(mo)+trbw(mo)+trfl(mo);
54.             ttr=ttr+tr(mo);
55.             END ;
56.             cfm=occno*vent;
57.             DO mo=1 TO 12;
58.             ventld(mo)=1.08*cfm*delt(mo);
59.             tvent=tvent+ventld(mo);
60.             END ;
61.             DO mo=1 TO 12;
62.             lliteld(mo)=llte*flat;
63.             tllte=tllte+lliteld(mo);
64.             END ;
65.             shp=250;
66.             lhp=200;
67.             DO mo=1 TO 12;
68.             occlds(mo)=occno*shp;
69.             occldl(mo)=occno*lhp;
70.             occld(mo)=occlds(mo)+occldl(mo);
71.             tocc=tocc+occld(mo);
72.             END ;
73.             DO mo=1 TO 12;
74.             sl(mo)=1000;
75.             tslgl(mo)=1000;
76.             load(mo)=tr(mo)+sl(mo)+ventld(mo)+lliteld(mo)+occld(mo);
77.             IF load(mo)>=heat THEN GO TO c6;
78.             heat=load(mo);
79.             heatmo=mo;
80.             GO TO c7;
81.     c6:     IF load(mo)<cool THEN GO TO c7;
82.             cool=load(mo);
83.             coolmo=mo;
84.     c7:     tload=tload+load(mo);
85.             END ;
86.             ebc=flat*bldcst;
87.             emec=flat*mecst;
88.             emes=flat*mespa;
89.             PUT EDIT('****************')(SKIP(4),COLUMN(20),A(15));
90.             PUT EDIT('*   RESULTS    *')(COLUMN(20),A(15));
91.             PUT EDIT('****************')(COLUMN(20),A(15));
92.             PUT EDIT('MAXIMUM HEATING','MAXIMUM COOLING')(SKIP(2),COLUMN(30),A(12),X(6),A(12));
93.             PUT EDIT('MONTH',heatmo,coolmo)(SKIP(1),A(5),X(25),F(12),X(6),F(12));
94.             PUT EDIT('TRANSMISSION LOAD:')(SKIP(1),A(18));
95.             ;
96.             PUT EDIT('WALLS',trwl(heatmo),trwl(coolmo))(X(4),A(5),X(21),F(12),X(6),F(12));
97.             PUT EDIT('GLASS',trgl(heatmo),trgl(coolmo))(X(4),A(5),X(21),F(12),X(6),F(12));
98.             PUT EDIT('DOORS',trdr(heatmo),trdr(coolmo))(X(4),A(5),X(21),F(12),X(6),F(12));
99.             PUT EDIT('ROOF',trrf(heatmo),trrf(coolmo))(X(4),A(4),X(22),F(12),X(6),F(12));
100.            PUT EDIT('FLOOR',trfl(heatmo),trfl(coolmo))(X(4),A(5),X(21),F(12),X(6),F(12));
```

```
101.            PUT EDIT('BSMT. WALLS',trbw(heatmo),trbw(coolmo))(X(4),A(11),X(15),F(12),X(6),F(12));
102.            PUT EDIT('SOLAR LOAD/GLASS',tslgl(heatmo),tslgl(coolmo))(SKIP(1),A(16),X(14),F(12),X(6),F(12));
103.            PUT EDIT('VENTILATION LOAD',ventld(heatmo),ventld(coolmo))(SKIP(1),A(16),X(14),F(12),X(6),F(12));
104.            PUT EDIT('LIGHTING LOAD',liteld(heatmo),liteld(coolmo))(SKIP,A(13),X(17),F(12),X(6),F(12));
105.            PUT EDIT('OCCUPANCY LOAD',occld(heatmo),occld(coolmo))(SKIP,A(14),X(16),F(12),X(6),F(12));
106.            PUT EDIT('____________','____________')(X(30),A(12),X(6),A(12));
107.            PUT EDIT('TOTAL LOAD',load(heatmo),load(coolmo))(SKIP,A(10),X(20),F(12),X(6),F(12));
108.            PUT EDIT('EXPECTED COST AND SPACE REQUIREMENTS:')(SKIP(2),A(37));
109.            PUT EDIT('TOTAL COST OF BUILDING = $',ebc)(SKIP,A(26),F(12,2));
110.            PUT EDIT('COST OF MECHANICAL AND ELECTRICAL EQUIPMENT = $',emec)(SKIP,A(47),F(12,2));
111.            PUT EDIT('(INCLUDED IN TOTAL COST)')(A(28));
112.            PUT EDIT('MECHANICAL EQUIPMENT SPACE REQUIRED = ',emes,'  SQ. FT.')(SKIP,A(38),F(12,2),A(9));
113.            CLOSE FILE(text) ;
114.            PUT LIST('END OF SUBROUTINE "CALC"');
115.            END calc;
```

```
 1.    update: PROCEDURE (opt);
 2.            DECLARE text FILE ENV( REGIONAL(1) F(80) OLD  ) ;
 3.            OPEN FILE(text) DIRECT UPDATE TITLE('Harry.moate.a30722') ;
 4.            DECLARE line CHAR(80), lin CHAR(10);
 5.            IF opt¬=7 THEN GO TO q1;
 6.            minlin=1;
 7.            minlin=250;
 8.            GO TO r2;
 9.    q1:     DO l=248 TO 250;
10.            READ FILE(text) INTO(line) KEY(l) ;
11.            PUT LIST(line);
12.            END ;
13.    a1:     GET LIST(edit);
14.            IF edit>=1&edit<=2 THEN GO TO r1;
15.            PUT LIST('INCORRECT INPUT.  Reenter either 1 or 2');
16.            GO TO a1;
17.    r1:     GET LIST(minlin,maxlin);
18.            IF edit¬=1 THEN GO TO r3;
19.    r2:     DO l=minlin TO maxlin;
20.            READ FILE(text) INTO(line) KEY(l) ;
21.            PUT EDIT(l,line)(F(3),COLUMN(5),A(80));
22.            END ;
23.            GO TO r6;
24.    r3:     IF edit¬=2 THEN GO TO r4;
25.            DO l=minlin TO maxlin;
26.            GET LIST(line);
27.            REWRITE FILE(text) FROM(line) KEY(l) ;
28.            END ;
29.    r6:     CLOSE FILE(text) ;
30.            PUT LIST('END OF UPDATE SUBROUTINE');
31.            END update;
```

The vita has been removed from the scanned document

ENERPAC: AN INTERACTIVE PROGRAM, IN THE CONVERSATIONAL MODE, FOR BUILDING ENERGY DESIGN

by

Harry Frank Moate, III

(ABSTRACT)

Energy shortages in urban areas and rapidly rising energy costs will soon make building energy consumption an important element in architectural design. Previously, this factor had been given consideration only in the latter stages of design, and then by the consulting engineer. Now, the architect will have to use it as one of the criteria in evaluating alternative solutions.

To aid the architect in this analysis, ENERPAC has been developed. ENERPAC is a computer program for building energy design. It is designed especially for the architect to use in the early stages of a project by using variables of the type and scale appropriate to this level of design. The program is written in conversation language, mainly in the form of questions directed to the user. After interrogating the user to obtain the necessary data, calculations are made to determine the building energy load. The user may then change the variables, as desired, to evaluate their effect on the load.

In addition to its use in new building design, the program has also proven useful for obtaining a quick determination of building

loads for cost estimates, such as adding air conditioning (cooling) to existing buildings.

The five areas of investigation and documentation covered by this thesis were (1) existing programs, (2) program requirements, (3) methodology, (4) results, and (5) conclusions. A listing of the program and a User's Manual are included in the Appendix.