

# MACRO CONTROL STRUCTURES FOR STRUCTURED PROGRAMMING IN ALC

THESIS

Presented to the Graduate Council of the North Texas State University in Partial Fulfillment of the Requirements

For the Degree of

MASTER OF SCIENCE

By

Kim G. Connally, B.A.

Denton, Texas

December, 1975

Connally, Kim G., <u>Macro Control Structures for</u> <u>Structured Programming in ALC</u>. Master of Science (Computer Sciences), December 1975, 187 pp., 4 tables, 27 illustrations, bibliography, 38 titles.

This thesis describes a set of computer program control structures which permits the application of certain structured programming techniques to the IBM/360 assembly language (ALC). The control structures are implemented by programmerdefined instructions known as macros.

A history of computer software is presented, providing a basis for the emergence of structured programming. A survey of the major concepts of structured programming with special attention to control structures and their significance to structured programming follows.

The macros developed in this study include DO, ENDDO, LEAVE, CASE, and ENDCASE. They provide a looping control structure, a loop-escape construct, and a selective control structure. Examples of usage are given.

# TABLE OF CONTENTS

LIST OF	Page TABLES
LIST OF	ILLUSTRATIONS vi
Chapter	
I.	INTRODUCTION 1
	The Evolution of Software Purpose of the Study Justification
II.	STRUCTURED PROGRAMMING
	Dijkstra's Work Mills' Work Modular Programming Program Clarity Open Programming Summary
III.	CONTROL STRUCTURES
	Bohm and Jacopini's Work Significance of Simple Control Structures The GOTO Controversy Subroutines Conclusion
IV.	CONTROL STRUCTURES IMPLEMENTED WITH MACROS FOR ASSEMBLY LANGUAGE PROGRAMMING 67
	Repetitive Control Structure Macros The LEAVE Macro Selective Control Structure Macros
ν.	SUMMARY, CRITIQUE, AND CONCLUSIONS 95
	Summary Critique Conclusions

TABLE OF CONTENTS (Continued)

																								1	Page
APPENDIX	A	•	•	•	٠	•	•	•	•	•	•	٠	٠	•	•	•	٠	•	•	•	•	ŧ	٠	٠	104
APPENDIX	В	•	•	•	•	•	٠	•	٠	•	•	•	•	"	•	٠	•	*	٠	٠	۴	٠	÷	٠	130
APPENDIX	С	•	•	•	•	•	٠	•	•	•	•	•	•	•	•	•	•	•	٠	4	•		•	•	151
APPENDIX	D	٠	•	•	•	•	•	•	•	٠	•	•	•	•	•	٠	٠	٠	•	٠	•	•	٠	•	172
BIBLIOGRA	∕ÞH	ΙY	•			•	•	•	•	•		•		•	•		•	•	•	•	•	•	•	•	185

# LIST OF TABLES

Table										Page
I.	Keyword	Parameters	for	the	DO Macro	٠	•	٠	•	70
II.	Keyword	Parameters	for	the	LEAVE Macro .	٠	•	٠	•	81
III.	Keyword	Parameters	for	the	CASE Macro	٠	٠	•	•	92
IV.	Keyword	Parameters	for	the	ENDCASE Macro	•	•	•	•	9.2

# LIST OF ILLUSTRATIONS

Figure		Page
1.	The Excessive Use of GOTO Statements in a Program	13
2.	Development Support Library	24
3.	Spacing and Indentation	32
4.	A Sequencing Structure	39
5.	A Selective Structure	40
6.	A DO WHILE Structure	41
7.	A DO UNTIL Structure	42
8.	Nested Control Structures	44
9.	Decomposition of Program Structures	45
10.	Improper Use of Control Structures	47
11.	Multivalued Selective Structure	49
12.	Multivalued Selective Structure Implemented by Nesting Simple Selective Structures	50
13.	Abnormal Termination of a Repetitive Structure	51
14.	A DO Loop with Multiple Exits	58
15.	Examples of DO Macro Parameters	72
16.	Flowchart of a DO and ENDDO Construct	74
17.	DO Macros Nested to the Second Level	<b>7</b> 5
18.	Flowchart of Stack Logic for Code in Figure 17	76
19.	Examples of LEAVE Macro Parameters	80

LIST OF ILLUSTRATIONS (Continued)

Figure	Page
20. Flowchart of Execution of a LEAVE Macro	82
21. Properly Coded LEAVE Macros	84
22. Improperly Coded LEAVE Macros	84
23. Flowchart of the Execution of a CASE and ENDCASE Construct	86
24. Example of CASE Macros Using Option=1	88
25. Example of CASE Macros Using OPTION=2	90
26. An Example of Kessler's CASE Macro	101
27. Flowchart Showing Structured Logic of Program M	175

#### CHAPTER I

#### INTRODUCTION

# The Evolution of Software

During the first generation of the computer industry, the late 1940's through the middle 1950's, the emphasis on the development of computer software was negligible (2, p. 470). At the time technology was focused on the development of functional hardware. The vacuum tube proved so unreliable that extensive maintenance was required; consequently, only a minimum amount of programming could be done. The few computers available were typically one-of-a-kind and short-lived, so the need for standard software packages was not felt. Since the early machines were slow in execution and contained small memories, the programmer was severely restricted by the capabilities of his computer (3, p. 860).

The lack of software development then was a direct result of the primitive and unreliable hardware in existence; moreover, many programmers believed that with the improvement of computer machinery the burden of the programmer would be lifted (3, p. 860). The programmer would no longer have to contend with the limitations of the

hardware but would enjoy a new freedom, and programming would no longer be a problem.

As a result of the early experiences with computers, the important role that software and programming would assume in future machines was grossly underestimated (2, p. 469). The second generation of computers, extending from the late-1950's to the mid-1960's, gave the programmer his larger and faster machine, and in so doing, completely altered the role of software in the computer industry.

The programmers of the second generation faced the task of developing software to match the advances in electronic technology. The transistorized machines of the second generation introduced an unforeseen complexity to programming. Programmers had to deal with such problems as I/O interrupts, multilevel stores and multiprogramming. The limitations of computing were shifting from the hardware to the software.

Another problem confronted the programmers of the second generation. With these new and more powerful machines the widespread use of computers in industry became feasible (2, p. 472). There was a tremendous demand for programmers and software by business and industry. Not only was the computer industry lacking in software development, but there was also an inadequate number of experienced programmers available to produce the needed software.

The problems of the software industry continued to grow with time. During the mid-1960's, the beginning of the third generation computers, integrated circuit technology came into use. Again computer hardware increased in speed and complexity. There now existed what has been termed the "software crisis" (2, p. 474). In short this crisis represented the disparity between the sophistication and capabilities of computer machinery and the inadequate and functionally underdeveloped software used by these machines.

In summary, as the technology of computer hardware improved, providing smaller, faster, and more complex machines, programmers were faced with the increasingly difficult task of designing software for these machines and programming the many problems of business and industry which these machines were capable of solving. To quote Dijkstra,

. . . as long as there were no machines, programming was no problem at all; when we had a few weak computers, programming became a mild problem, and now that we have gigantic computers, programming has an equally gigantic problem (3, p. 861).

A crisis has developed as a result of the inabilities of the programmers and their software to meet this problem.

Today this problem is best reflected in the software produced in industry. Industrial programs are usually very expensive since they typically require many man-hours to code, and quite often they are error-prone because their complexity prevents adequate debugging. Industrial software

is usually difficult to understand, hard to modify, requires constant maintenance and cannot be adequately tested for correctness. This is not surprising since many industrial programmers have had no formal training in organized program production.

The typical industrial programmer produces on the order of five to ten lines of code per day over the average life of a project because most of his time is spent debugging (1, p. 58). Programming techniques in present use are not producing either the quantity or quality of software that is in demand.

The computer industry has realized the inadequacies of software and programming techniques for quite some time. Accordingly, within the last seven years a growing number of programmers have expounded various concepts and methodologies which they believe will improve software design and will increase the productivity, reliability, maintainability, and extendability of programs. These various concepts and techniques have been referred to as "structured programming." At this time there is disagreement as to what does or does not constitute structured programming. Some programmers feel that it encompasses a wide variety of techniques while others view it as a single or perhaps limited aspect of programming. It remains to be seen exactly what will eventually be included in a definition of structured

programming, but its development has been a direct result of the software crisis.

Chapter II presents a survey of the major concepts associated with structured programming. The work of E. W. Dijkstra is first presented. It includes the concepts of "GOTO-less" programming, hierarchy of software modules, abstract resources and the principle of non-interference. Dijkstra's work is followed by a discussion of the contributions of H. D. Mills, which includes such concepts as the "top-down" approach to programming, the chief programmer team method, the one-entry and one-exit rule for program module design, and the development support library. A number of other important structured programming concepts which are not associated with a particular contributor are also discussed. These concepts include modular programming, program clarity, and open programming.

Of the various concepts involved in the structured programming controversy one of the most commonly implicated is the issue of control structures. A control structure is a program instruction or set of instructions which determines the order in which other program instructions will be executed or the number of times a particular set of instructions are to be executed. Simple control structures include the unconditional branch statement, the branchon-high and the branch-on-low instructions. More

sophisticated control structures include the FORTRAN DO statement, PL/I IF-THEN-ELSE statement, and program interrupts caused by external events.

The controversy centers around the correct use of control structures. What set of control structures should the programmer be allowed to use? Should he have a large set of structures at his disposal or should he be limited to a small but sufficient set? Should the GOTO statement be allowed or should it be removed completely? What discipline should be imposed on interrupt programming methods? These are some of the questions to which programmers are addressing themselves in an effort to determine the proper set and use of control structures in programming.

## Purpose of the Study

The purpose of this thesis is to describe the development of a small but sufficient set of control structures to facilitate structured programming in ALC, the macro assembly language of the IBM 360 computer.

An ALC macro call consists of a single programmerdefined macro instruction which may be inserted into an ALC program. At assembly time the macro instruction is replaced by a predetermined set of ALC instructions associated with it. With a set of ALC macro definitions for various higher-level language control structures an assembly

language programmer is able to code programs using the concepts of structured programming which require these more sophisticated structures. The programmer uses the macro instructions to implement the logic defined by higherlevel language control structures.

Before describing the macros developed in this thesis, a survey of the major concepts of structured programming is presented both to familiarize the reader with structured programming and to set in proper perspective the important role that control structures play in structured programming. The concepts of control structures are more fully expanded in a separate chapter which precedes the discussion of the macros.

## Justification

The numerous articles which discuss structured programming and the control structures of structured programming limit themselves almost without exception to a discussion of concepts which apply to higher-level languages. Kessler (4) has published a noteworthy report which did attempt to apply the concepts of structured programming directly to assembly language. Given the present state of software and the prevalent use of assembly languages in industrial programming, it should be evident that there exists a need for a thorough investigation into

the application of structured programming techniques to the control structures of assembly language programming.

Assembly language programs by the nature of their instruction sets are difficult to code and debug; thus the need for improving assembly language programming techniques is obviously present. Since assembly language programs usually require many more lines of code than equivalent higher-level language programs, the necessity of improving assembly language programs would seem to be greater than that of higher-level languages. One of the major intentions of structured programming is to make programs more readable; the assembly language program is in most instances more difficult to read than a comparable higher-level language program.

The use of structured programming control structures in higher-level languages facilitates a block-structured or modular program design. If these control structures are applied to assembly languages, then block-structured programming could be more easily implemented in assembly language programs. The control structures or branching mechanisms which are presently available in most assembly languages do not readily lend themselves to modular or block-structured programming.

Higher-level languages usually contain a variety of control structures, some of which may be very powerful.

In contrast, the typical assembly language contains an unconditional branch, unconditional subroutine linkage, and a small set of very simple conditional branches such as a branch-on-equal, branch-on-high, branch-on-negative, etc. With the implementation of a more powerful and problemoriented set of control structures in assembly languages the methods of structured programming could be applied more directly to assembly language programming.

This research was undertaken in light of the fact that there exists a need to improve the techniques of assembly language programming. If the concepts of control structures of higher-level languages within the scope of structured programming are applied at the assembly language level, perhaps significant improvements in assembly language programming will be achieved.

Before entering a discussion of control structures and their proper use, an overview of the development of structured programming, its major concepts, and its major contributors will be given in Chapter II in order to establish the significance and relationship of control structures to structured programming.

## CHAPTER BIBLIOGRAPHY

- Baker, F. T. and H. D. Mills, "Chief Programmer Teams," <u>Datamation</u>, XIX (December, 1973), 58-61.
- Bauer, F. L., "Software and Software Engineering," <u>SIAM Review</u>, XV (April, 1973), 469-480.
- 3. Dijkstra, E. W., "The Humble Programmer," <u>Communica-</u> <u>tions of the ACM</u> (October, 1972), 859-866.
- 4. Kessler, M. M., <u>\*Concepts\* Report 4</u>, <u>OS/360</u> <u>Assembly</u> <u>Language Block Structured Programming Macros, IBM.</u>
- Rosen, Saul, "Electronic Computers: A Historical Survey," <u>Computing</u> <u>Surveys</u> (March, 1969), 7-36.

## CHAPTER II

#### STRUCTURED PROGRAMMING

The origin of structured programming is usually traced to a letter by E. W. Dijkstra (7), which appeared in the March, 1968, issue of the <u>Communications of the ACM</u>. The letter entitled "GOTO Statement Considered Harmful" warned programmers of the potential problems which the GOTO statement may introduce into programs.

# Dijkstra's Work

In his letter Dijkstra pointed out that the unrestricted use of the GOTO statement may unnecessarily complicate the flow of control within a program. Because of the complexity of such a program difficulties may be encountered if debugging or modification is required.

If a program contains many GOTO statements, it is likely that the program will have a nonlinear flow of control. This means that its instructions will not be executed in the same order as they are written. In contrast, when a program contains no GOTO's or other branching constructs, its instructions are executed in a purely sequential manner. The first instruction is executed first, the second next, and so forth until termination

occurs. Such a program is said to have a linear flow of control. While few programs can be written in a purely linear fashion, the free use of GOTO's may completely destroy any semblance of linearity and produce unnecessarily complicated program logic.

Figure 1 illustrates the type of program which may result from the blatant use of the GOTO statement. Notice that the program has a very complicated execution sequence. There is no correspondence between the order in which the statements appear and their order of execution. Although the program of Figure 1 is rather small, it would be difficult to debug or prove correct because of its nonlinear flow of control.

Since the time of Dijkstra's letter so much literature has been published on the use of GOTO statements that structured programming and "GOTO-less" programming are sometimes used synonymously. This, however, is an oversimplification. A program which contains GOTO's may be very well structured in the sense that it maintains an almost linear or sequential flow of control. On the other hand the flow of control of a program coded without GOTO's is not inherently linear or necessarily well structured. Dijkstra cautioned the programmer that the GOTO statement lends itself to misuse, i.e., it may disrupt the sequential execution of the program. He was not suggesting that programming without

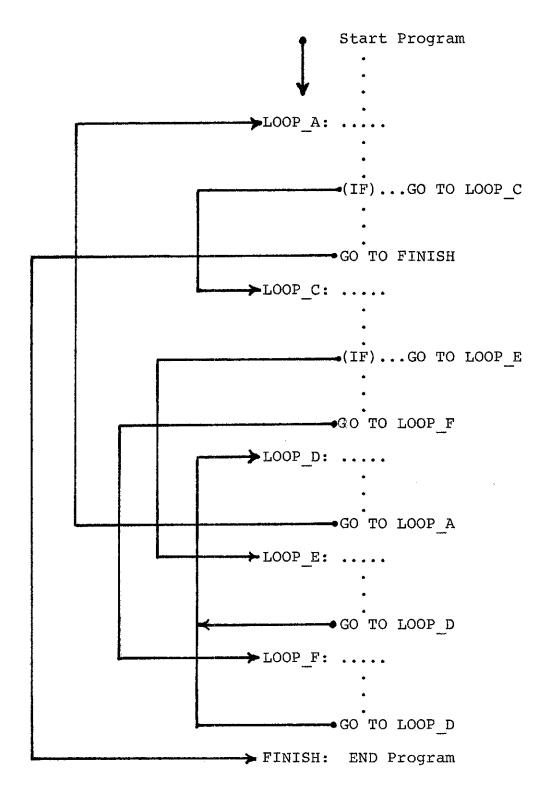


Fig. 1--The excessive use of GOTO statements in a program.

the GOTO statement would in itself produce well-structured programs.

It has been shown in a paper by Bohm and Jacopini (4) that the following three control structures are a sufficient set to define any program logic: a sequence control structure, a repetitive control structure, and a selective control structure. Using these three structures only, the GOTO statement is not needed nor for that matter are any other control structures required.

At this time there is a controversy among programmers as to whether or not the GOTO statement should be completely eliminated since it is not strictly required. Those in favor of its removal argue that it is merely a temptation toward poor programming practices. Those not in favor of its removal argue that circumstances occur when the judicious use of an occasional GOTO statement will not interfere with the program's structure and will, in fact, provide a solution to a logic problem which may otherwise require excessive code.

The concept of programming without the GOTO statement is by no means the only contribution of Dijkstra to structured programming. Concerned with the development of software systems, he published his experience with the "THE" Multiprogramming System which made use of what he calls abstract resources in the design of software systems (6).

Dijkstra views a software system as a hierarchy of software modules or machines. Each level of the hierarchy produces an abstract resource which is supported by the level below it. Each of the resources is available to the level directly above it. Any module then consists of a set of programs which manipulate the abstract resource of the next lower level and produce an abstract resource which can be manipulated by the next higher level.

An example will illustrate Dijkstra's concepts of abstract resources, hierarchy layers, and levels of abstraction.

A software package which will read a file is to be written. The file is considered the highest level of abstraction and will consequently, be the highest layer in the hierarchy. The next level is a record since a file can be considered a collection of records. A record is a collection of words, a word is a collection of bytes and a byte is a collection of bits so that the hierarchy of resources needed to read a single bit is as follows:

Highest Level: FILE

RECORD WORD

BYTE

#### Lowest Level: BIT

There are five abstract resources and hence, five levels of abstraction. Each level is a layer in the hierarchy and represents a functional part or machine in the system. A set of programs would be written to implement each level. For instance, at the lowest level a set of programs would be written which operate on bits and thus alter bytes. At the highest level a set of programs would operate on records to manipulate files.

Dijkstra emphasizes that each level has a specific relation to the level immediately above and below it, and that special care should be taken to insure that each level is consistent in that the operations executed at one level will be supported by those below and will, in turn, support those in the level immediately above.

In developing his software system Dijkstra attempted to use a design which would lend itself to thorough testing and which could be proven logically correct. Using his method of hierarchy of levels, Dijkstra was applying what he refers to as the "principle of non-interference" (3, p. 143). This principle suggests that the correctness of the software system can be more readily determined if the system is divided into a set of smaller problems which are logically independent and which can be united into the software package at the functional level. These small problems must be proven correct at every level of integration.

To test the example above using this principle it would be necessary to test each layer. For example, the

middle layer would be independently tested to insure that it correctly manipulates bytes to produce words. Likewise, when each layer had been thoroughly and independently tested, the system would be considered correct.

It should be noted at this time that Dijkstra's hierarchy of levels or layered approach to software design appears to be equivalent to a modular approach to program-In fact, Dijkstra's approach is modular, but he has ming. imposed other restrictions such as the hierarchy concept on the modules formed by the design. The sample given above is modular in that the byte programs are independent of the bit and word programs, however, there is the added restriction that the word programs may manipulate only bytes and have no direct effect or connection to record, file or bit programs. Modular programming, which will be discussed later, does not necessarily place such restrictions on the relationships among various modules. There is no hierarchy or layering of modules in the conventional approach to modular programming.

Dijkstra seems to be most concerned with developing techniques for designing and encoding programs which will lend themselves to testing and proof of correctness. He advocates the avoidance of GOTO statements because they tend to add complexity to programs and hence, increase the difficulty of proving their correctness (7). Dijkstra approaches software design with his hierarchy technique

since to verify the software system, it is necessary to prove correct only the independent layers rather than the system as an entity. In designing his multiprogramming system Dijkstra was assisted by a group of programmers who were mostly mathematicians; thus, an indication of the emphasis he places on the necessity of proving program correctness (6).

## Mills' Work

Harlen Mills, F. T. Baker, and others at the IBM Corporation have developed some operational procedures for the design and encoding of large reliable programs. The techniques they have developed represent a major contribution to structured programming.

Mills (14) describes what he calls a "top-down" approach to program design and coding. He also elaborates on a number of programming techniques, which he collectively refers to as structured programming. These techniques include the use of a standard set of control structures or branching conventions, a modular or segmenting method of programming, and a restriction on any module that it have only one entry and one exit point.

The top-down approach to programming is a technique whereby the initial problem is repeatedly broken down into a hierarchy of program modules or segments. The highest module might represent a control or supervisory function.

The various routines called by the control module would represent the next lower-level of the hierarchy. Likewise, the second-level routines reference various lower-level routines. In effect, a tree structure containing program modules is formed. Coding begins at the highest level and proceeds downward with "program stubs," dummy names to represent uncoded segments being inserted into the code where references to lower-level segments are made.

The program modules or segments are carefully limited in size so that they may be coded on a single page. A segment defines a function having one entry and one exit point. The function merely transforms data which may or may not represent another segment.

The branching conventions or control structures which may be used in coding a segment include a simple sequencing of code, a selective branching structure such as an IF THEN ELSE statement and a repetitive control structure such as a DO WHILE statement. The GOTO statement is not permitted.

By requiring that each program segment be designed and coded according to these particular structured programming techniques, a certain amount of program uniformity is insured (12, pp. 22-23). Each segment will be reasonably small and will not possess any complicated control structures to interfere with module testing or readability (13, p. 156). Module interfacing is simplified when the one

entry and one exit rule is used since there can be only one path of connection between program segments.

A top-down design method permits program testing concurrently with program coding (13, pp. 9-10). Once a particular segment on the hierarchy structure has been coded, it may be tested by having the programmer provide the necessary input to evaluate the correctness of the segment. This input, of course, will eventually be supplied by the lower-level segments which are called by the segment being tested. In this manner the program can be verified as each module is coded; thus the reliability of the entire system is proven without having to rely on exhaustive testing once the system is complete.

Mills' top-down programming technique is similar to Dijkstra's layered approach to program design; however, the two methods seem to depend on slightly different concepts and each emphasizes somewhat different aspects of design. While Dijkstra is chiefly concerned with the separation of abstract resources during his hierarchy development (13, p. 57), Mills is mostly interested in achieving a hierarchy tree structure of one entry and one exit modules. Dijkstra emphasizes a design approach which is intended to be highly testable and provable while Mills, though he is concerned with the ability to test programs seems to emphasize simplicity and clarity in program design (12, p. 57).

Another concept of structured programming developed at IBM and described by Mills (13) is the "chief programmer team," which is an organizational and managerial technique for large program production.

The chief programmer team consists of a chief programmer, a backup programmer, a programming librarian, and other junior programmers required by the particular production. These team members utilize a development support library and apply the principles of top-down design and structured programming described by Mills in the development of large programs.

The chief programmer is a technical manager responsible for designing and coding the most important segments of the program. All other team members receive their responsibilities and coding assignments from the chief programmer. He coordinates all program interfaces and supervises all coding to insure the proper use of top-down structured programming techniques. Quite naturally the chief programmer must be a highly experienced and competent programmer since the success of the team and the project are largely dependent upon his decisions.

The backup programmer is a research assistant to the chief programmer and helps significantly in the design and coding of the major portion of the project. He must be completely familiar with the entire project and be ready to assume the position of chief programmer should the need

arise. The backup programmer shares the burden of responsibility, allowing the chief programmer to concentrate on the major problems encountered in the project. The backup programmer may, for example, develop all testing procedures without the assistance of the chief programmer.

The programming librarian is responsible for maintaining all listings and records of the project. This information is kept in both an internal, machine-readable, and an external, human-readable, form. The librarian maintains this information in a development support library.

The development support library contains a set of organized listings which detail the current status and previous development of the project. These listings represent the external project records. Among these records are notebooks which are headed by a directory and contain an alphabetized list of the program modules. A journal is kept to record all changes in updating the directory. All results of testing procedures are also recorded in a journal.

By maintaining a detailed record of the project's development the programmers have an accurate account of all program bugs encountered and tests made at any given time. Since these records are maintained by a programming librarian, the programmers are not burdened with timeconsuming clerical work.

Besides maintaining and updating the various records which are kept in the library, the librarian has a significant amount of paperwork to do concerning the documentation of the design and coding phases of the program. Mills (2) emphasizes that the librarian is a key member of the team rather than a part-time assistant to programmers.

The development support library consists of a number of office and machine procedures for maintaining programmer-generated material such as coded program segments, for maintaining files and records of the project in the external records, for processing data in the internal library which is on disk, and for performing all runs during each stage of program development including testing procedures.

During the course of the project programmers make corrections in status notebooks, introduce new or altered coding sheets, and request various runs. It is the responsibility of the programming librarian to invoke the necessary office or machine procedures to accomplish these tasks. He is responsible for preparing and executing all program runs and posting the results in the external and internal files.

Figure 2 shows the relationship of the librarian and programmers within the development support library (2). It is evident in the diagram that the librarian plays a key role in the success of the library.

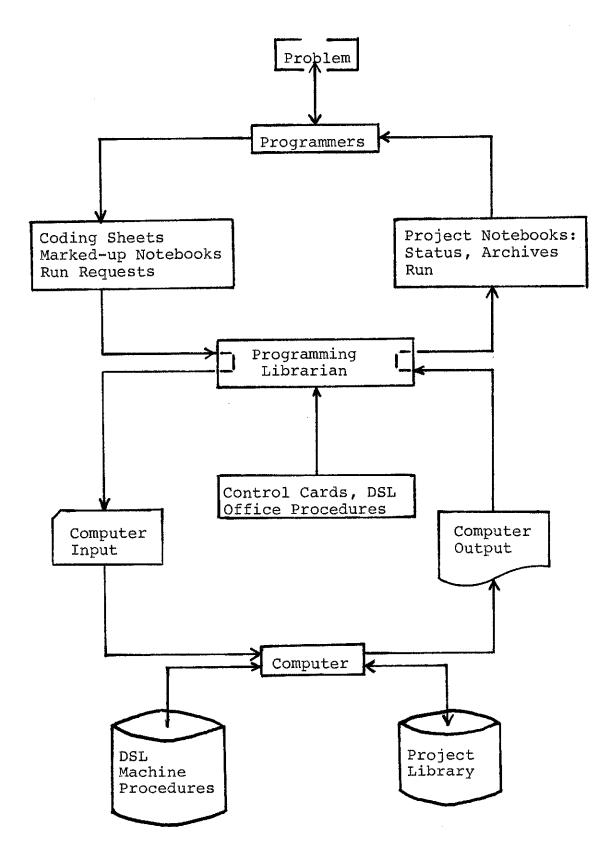


Fig. 2--Development support library

By using the development support library within the chief programmer team method, the team members are working on a common product rather than merely coding independently, separate segments of a large program. Since the members are working together, there is less chance of duplicating code or coding errors. The library provides an in-depth documentation of the development of the project. This documentation reflects the progress of the team.

The chief programmer team concept was developed in order to improve the organization, communication, and productivity of programmers involved in large-scale programming. The top-down method is applied to program design under the chief programmer team method, and the structured programming techniques are applied in code segments within the top-down method.

F. T. Baker (1) described an industrial project which was developed using a chief programmer team. The project required twenty-two man-months to design and code, involving the production of over 83,000 lines of code. Only twenty-one errors were found during formal testing of the completed system.

Baker concluded that the use of the chief programmer team, top-down programming, and the application of structured programming techniques contributed to the success of the project. He suggested that the top-down method may not always be applicable to some types of projects, and that

other methods may be more feasible in the design of some programs. For instance, when a program organization, viewed as a tree structure, is narrow and tall, then a strictly top-down approach may require too much time to be practical (1, p. 343). Baker also suggested that the chief programmer should do more code reviewing and allow the other members to do the major portion of coding. In the project described by Baker the chief and backup programmers did most of the functional coding; consequently, there was little time to review code, especially that written by junior programmers.

As a result of the development of top-down programming and the chief programmer team at IBM, the concept of structured walk-through evolved at that corporation (16, p. 31). A structured walk-through is a series of progress meetings which are held at various times in the design and development of a programming project.

A committee of approximately five members discuss the completeness, accuracy, and general quality of a project's development (10, p. 30). Each member of the committee, or reviewer as he is called, presents a brief introduction to his portion of the project and then "walks" the other committee members through the specific function of his area of responsibility. In this way each member becomes familiar with the purpose and progress of all aspects of the project.

One committee member, the recording secretary, records errors and inconsistencies which are discovered during each reviewer's walk-through. When the meeting is completed, each reviewer is given a copy of the secretary's notes. It is the responsibility of each member to insure that any problems within his area are resolved and that he notifies other committee members of any corrective action he takes.

The purpose of a structured walk-through is not to evaluate the ability or effectiveness of its committee members, but rather, through the exchange of information and ideas, determine the progress of the project, and detect the errors existing in the production at its current level of development (16, p. 30). Committee members are encouraged to exchange ideas, offer constructive criticism, and view the meetings as educational experiences.

The use of structured walk-through provides a method for not only measuring a project's development, but also for discovering production errors at the earliest possible time when they are easiest to correct and have the smallest impact on the production (10, p. 35).

## Modular Programming

Modularizing a program refers to the technique of isolating sections or functions in the program so they may be designed and coded independently; hence the original problem is reduced to a set of smaller ones. Program

modules might be coroutines, subroutines, programmer-defined functions, loops, or merely sequences of instructions which are logically grouped and coded together. The concept of modularity is not new to programming, but it has recently been given new emphasis as a structured programming technique.

The advantages of modular programming include: simplification of program design and coding by reducing a large problem into a set of smaller ones, extendability of coded modules since a functional program module can be inserted into other programs, ease of program modification where only effected modules need to be altered, and clarity of program design since the function of the entire program can be viewed as the interaction of the individual program modules.

Within the realm of structured programming programmers are well aware of the benefits of modular programming. They do not agree, however, on the principles which should govern program modularizing. Two methods which form a basis for program modularization have already been discussed, namely Dijkstra's levels of abstraction which modularizes according to what he calls abstract resources, i.e., the various elements or data types on which a program operates, and Mills' top-down design approach which modularizes according to a stepwise refinement of program segments. Besides these methods there exist two other important techniques which

involve program modularity. They are compartmentalization and information hiding.

Compartmentalization is an approach to program coding whereby program modules are formed on the basis of their relationship to a particular design decision. Examples of design decisions include such things as I/O formats, arithmetic precision, and variable declarations. The advantage of this approach to program modularity is its ease of modification when new design specifications must be incorporated. For example, in a compartmentalized program if a new input format were required, only the module which generated formats would need to be changed. There would be no need to search through various program segments to alter each format statement.

To implement compartmentalized modules the most appropriate technique is by means of macros. One macro may expand to produce various format statements while another may, for instance, generate declaration statements. Such macros may provide the basis for program modules rather than other program features such as functions or subroutines.

Parnas (15) has suggested a method of modular programming which attempts to reduce the interface requirements and relationships between modules and thus reduce module connectivity. His method, information hiding, stresses the need to code some modules without utilizing characteristics or features of other interfacing modules. The

programmer is supplied with the information needed to code his module, and information relating to the connecting modules is deliberately withheld or "hidden" from him. This technique may require additional coding to facilitate the proper interfacing of modules, but information hiding produces program modules which are inherently nonrestrictive and independent in nature.

It appears that no single modularizing technique is sufficient to always produce the most efficient and wellstructured program; rather the technique to use seems to be dictated by the nature of the problem. The current interest in modular programming should yield valuable methods for the decomposition of programs and contribute significantly to structured programming.

## Program Clarity

It was once considered sufficient to produce a program which would satisfy the problem at hand. It did not really matter if the program could be interpreted by anyone other than its author. As programs increased in size, however, and as the need for program modification became greater, program managers began to insist on code which could be understood and readily interpreted.

One of the basic goals of structured programming is to establish methods which will increase the readability of programs. This means writing a program in such a way that

programmers unfamiliar with it will be able to understand what it is attempting to do.

The use of documentation is very important in providing program clarity. If there are explanatory comments at the beginning of each program module, the purpose of the code will be more apparent. Besides explaining code documentation should specify any restrictions or exceptions to code function such as input format requirements, arithmetic precision of output, or "special cases" not handled properly by the program.

The use of spacing and indentation will also improve the clarity of program documentation. Spacing to separate program modules and indentation to represent loop nesting and extent of control within a loop will help produce more legible code.

Figure 3 illustrates the use of spacing and indentation. Modules A and B in the figure are separated by spacing to show their logical independence. Module B has two levels of indentation to indicate both the range of the IF statement within the outer DO, and the range of the THEN DO and ELSE DO statements within the IF. Notice that spacing within the IF statement clarifies the effect of the THEN DO and the ELSE DO routines.

The use of meaningful labels and variable names will also improve program readability. Acronyms and word abbreviations may be necessary if the language being used

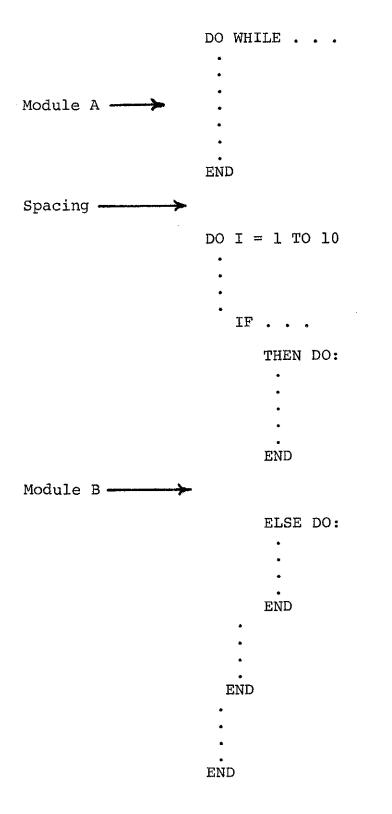


Fig. 3--Spacing and indentation

places restrictions on the lengths of labels and variable names.

# Open Programming

Programming was once considered an art which only a few people knew how to do, but today it is more and more becoming a professional skill which can be taught and improved upon with proper guidance. Indeed, the basis of structured programming is a set of methods and techniques which assumes that programming can be taught and improved upon with the application of these techniques.

Open programming is a method for teaching and improving programming skills. It involves a group approach to learning. Programmers compare and discuss each other's work making evaluations and suggestions for improvement. An open panel discussion may be used to critique programs. There is frequent exchange of experiences and problems with fellow programmers.

The basic goal of open programming is self-improvement of programming techniques by studying the programming methods of others and having others evaluate one's own work. Weinberg (16) speaks of this practice as "egoless" programming. Each programmer must be willing to accept the constructive criticism of others while, at the same time, share his programming abilities with his critics.

#### Summary

This chapter has presented the major concepts associated with the term "structured programming." A variety of methods and techniques have been discussed, and it remains to be seen just which of them will be included in a formal definition of structured programming.

Intuitively it appears that any definition of structured programming will have to be of a general nature encompassing many concepts and techniques since the solution of the multitude of problems presented to programmers today require a variety of approaches and methods, especially with regard to program design.

Chapter III will discuss more fully the role of control structures in structured programming. So far it has been suggested that the proper use of control structures is to adapt a small but sufficient set of branching conventions and that the use of the unconditional GOTO branching statement should be avoided. Chapter III will present some of the theory and application of control structures with a more thorough analysis of the GOTO controversy.

#### CHAPTER BIBLIOGRAPHY

- Baker, F. T., "System Quality Through Structured Programming," Proc. FJCC, (1972), 339-343.
- , and H. D. Mills, "Chief Programmer Teams," <u>Datamation</u>, XIX (December, 1973), 58-61.
- Benson, Jeoffrey, "Structured Programming Techniques," <u>Record of the 1973 IEEE Symposium on Computer</u> <u>Software Reliability</u>, New York, (April, 1973), <u>143-147.</u>
- Bohm, C. and G. Jacopini, "Flow Diagrams, Turing Machines and Languages with Only Two Formation Rules," <u>Communications of the ACM</u>, IX (May, 1966), 366-371.
- Denning, P. J., "Is It Not Time to Define 'Structured Programming'?" <u>SIGPLAN Notices</u>, (February, 1974, 6-7.
- Dijkstra, E. W., "The Structure of the 'THE' Multiprogramming System," <u>Communications of the ACM</u>, XI (May, 1968), 341-356.
- 7. \_\_\_\_\_, "GOTO Statement Considered Harmful," <u>Communications of the ACM</u>, XI (March, 1968), 147-148.
- 8. \_\_\_\_\_\_, "Complexity Controlled by Hierarchecal Ordering of Function and Variability," <u>Software</u> <u>Engineering</u>, Report on a conference sponsored by the NATO Science Committee, Garmish, Germany, October, 1968, 181-185.
- 9. Donaldson, James, "Structured Programming," <u>Datamation</u>, XIX (December, 1973), 52-54.
- 10. Improved Programming Technologies Management Overview, IBM Corporation, Data Processing Division, Systems Marketing, Installation Productivity Programs Department (August, 1973).
- 11. McCracken, Daniel, "Revolution in Programming: An Overview," <u>Datamation</u>, XIX (December, 1973), 50-52.

- 12. Miller, E. and G. Lindamood, "Structured Programming: Top-down Approach," <u>Datamation</u>, XIX (December, 1973), 55-57.
- 13. Mills, H. D., <u>Chief Programmer Teams</u>: <u>Principles and</u> <u>Procedures</u>, Report No. FSC 71-5108, IBM Federal Systems Division, Gaithersburg, Maryland.
- 14. , "On the Development of Large Reliable Programs," Record of the 1973 IEEE Symposium on Computer Software Reliability, New York (May, 1973), 155-159.
- Parnas, D. L., "Information Distribution Aspects of Design Methology," <u>Information Processing</u>, LXXI, 339-344.
- 16. Weinberg, G., <u>The Psychology of Computer Programming</u>, New York, Von Nostrand Reinhold Co., 1971.

#### CHAPTER III

#### CONTROL STRUCTURES

Almost every paper on structured programming alludes to the proper use of control structures. This is not surprising since the structures used can affect many aspects of a program. The degree of linearity in program execution is a direct result of the structures used. Program modularity can be maximized with effective control structures. Testing a program and proving its correctness may be feasible only if the program's structures allow the isolation of program segments. The clarity and readability of a program may be significantly hampered if it contains complicated control mechanisms.

# Bohn and Jacopini's Work

The importance of control structures has not been overlooked by those interested in improving programming techniques. A paper by Bohm and Jacopini (2) is frequently referenced in structured programming articles. The paper shows mathematically that the control of any flowchartable program can be logically defined using only three control structures. These structures include 1) a sequencing procedure, 2) a selective structure, and 3) a repetitive

structure. Each structure is characterized by a single entry and single exit point.

The sequencing structure, shown in Figure 4, represents the normal execution of instructions, i.e., in the order the instructions were written. Figure 4 represents a linear flow of control. A program containing no branching structures would be completely sequential or linear in the strictest sense.

A selective structure, illustrated in Figure 5, causes the execution of a particular block of code or set of instructions depending on the truth of the selective condition. In the diagram the diamond represents the selective mechanism. The IF THEN ELSE structure of PL/I is a selective structure. When the IF statement is true, all instructions within the THEN block are executed. When the IF statement is false, all instructions within the ELSE block are executed.

The IF THEN ELSE structure may be considered linear since each instruction is executed or not executed in the same order as it is coded. The program flow of control does not change directions but continues forward either bypassing instructions or executing them as they are encountered.

The third structure, a repetitive block, is shown in Figures 6 and 7. This structure represents the repeated execution of a block of code. The diamonds in the diagrams

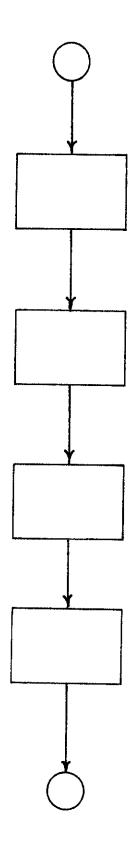


Fig. 4--A sequencing structure

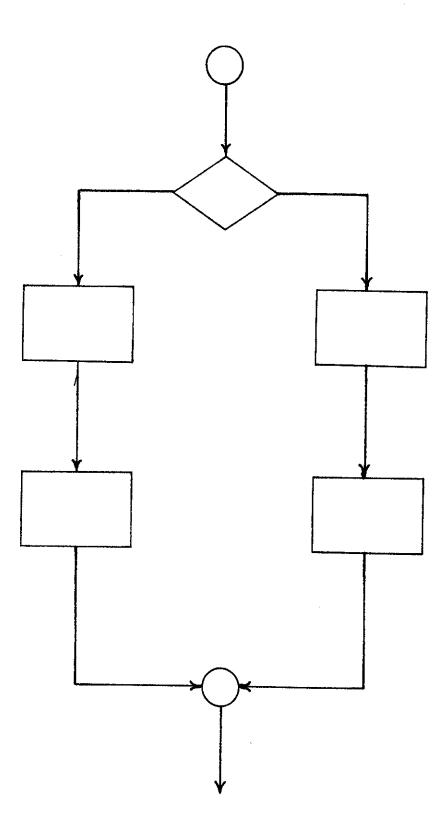


Fig. 5--A selective structure

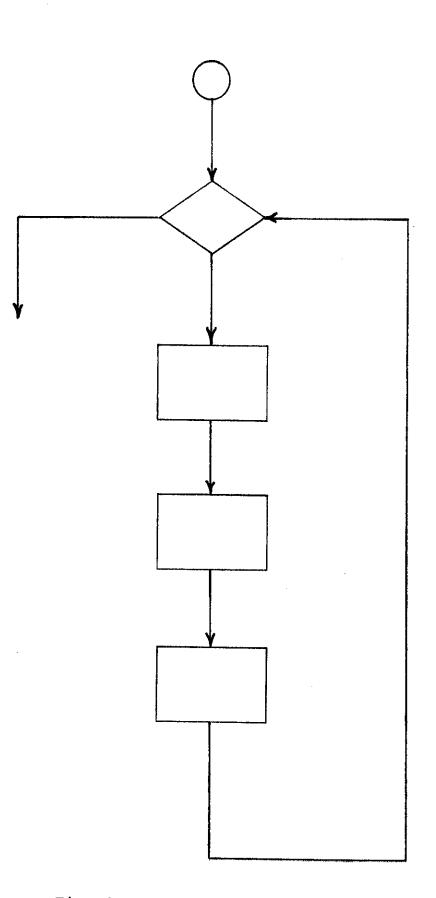


Fig. 6--A DO WHILE structure



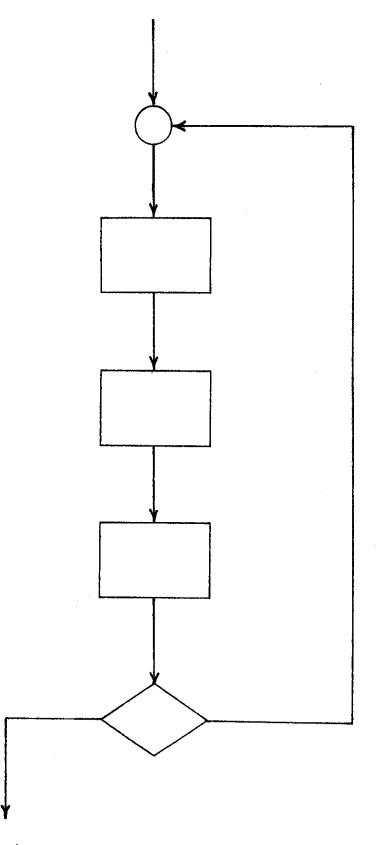


Fig. 7--A DO UNTIL structure

represent the selective mechanisms which determine the number of repetitions. There are two forms of this structure. The first one, Figure 6, tests the looping condition before loop execution. This is implemented in PL/I with the DO WHILE statement. The second form, Figure 7, implemented in ALGOL with the DO UNTIL statement and in FORTRAN with the DO statement, tests the looping condition after each execution of the loop. When the looping condition is satisfied, the repetitive structure passes program control to the instruction immediately following the loop.

Any program logic can be performed by some combination of these structures. They may be sequenced one after another or nested in any combination. As an example, Figure 8 shows the nesting of several structures. A repetitive block is contained within a selective block which is itself nested within another selective block. Notice that the one entry and one exit feature is maintained within the logic structure.

A simple program which uses only the structures suggested by Bohm and Jacopini is diagrammed in Figure 9. The program includes combinations of all three structures and can be decomposed into the various structures used. Each structure maintains a single entry and single exit point; this insures a certain degree of consistency in program design.

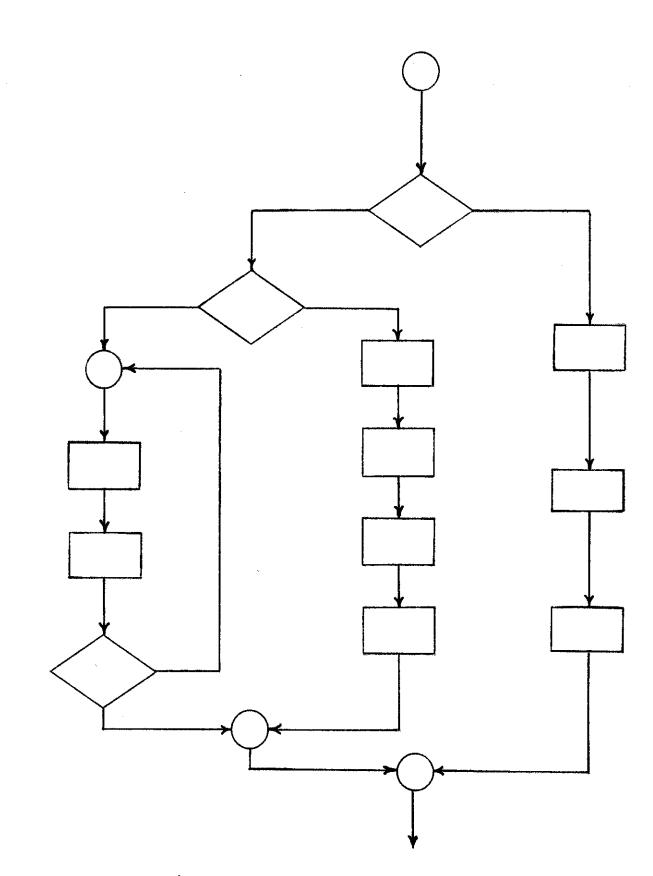


Fig. 8--Nested Control structures

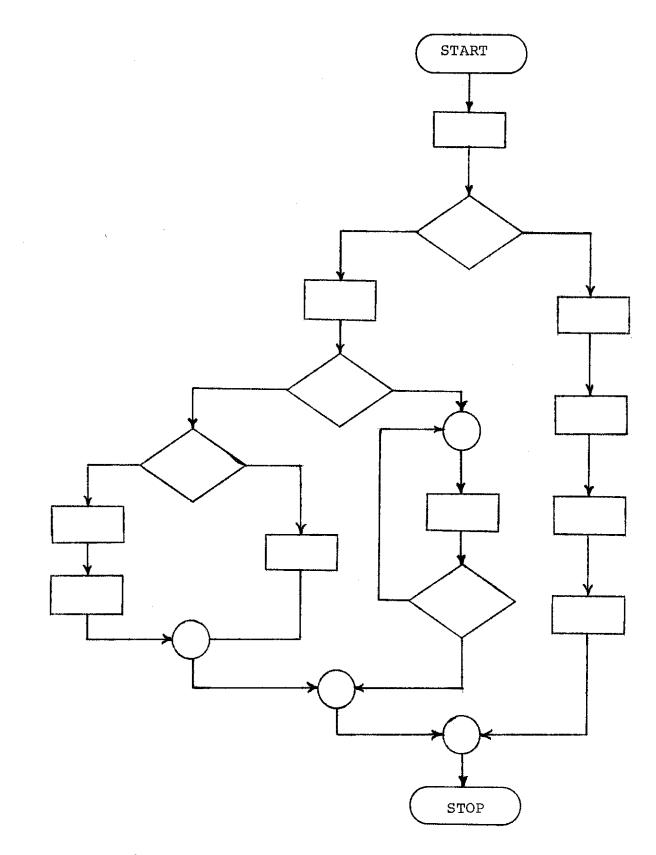


Fig. 9--Decomposition of program structures

Figure 10 represents a program containing control structures other than the three just described. In the figure blocks one through four appear to form a DO UNTIL structure; however, the loop condition may possibly be overridden by decision block two which provides a branch from within the loop. In blocks nine, eleven, and twelve a similar loop situation exists in that there are two decision blocks and two exits from the loop. In the second situation, however, the logic is further complicated by the fact that different paths are taken depending upon the point of departure from the loop.

Figure 10 is a reasonably short program, yet its unusual control structures may prove difficult to debug. Also problems may arise in understanding the purpose of the code because of the rather complicated logic structure. It appears that the programmer has developed control structures to accomplish the program logic, but he has made no attempt to develop a relatively uncomplicated or readable program design. By adhering to the use of the three simple and sufficient control structures, the programmer can define the logic of any program in a manner that will require only simple control structures which may be combined in larger decomposable structures.

Donaldson (4, p. 53), however, describes two occasions where the strict use of only the three mentioned structures will result in inefficiency.

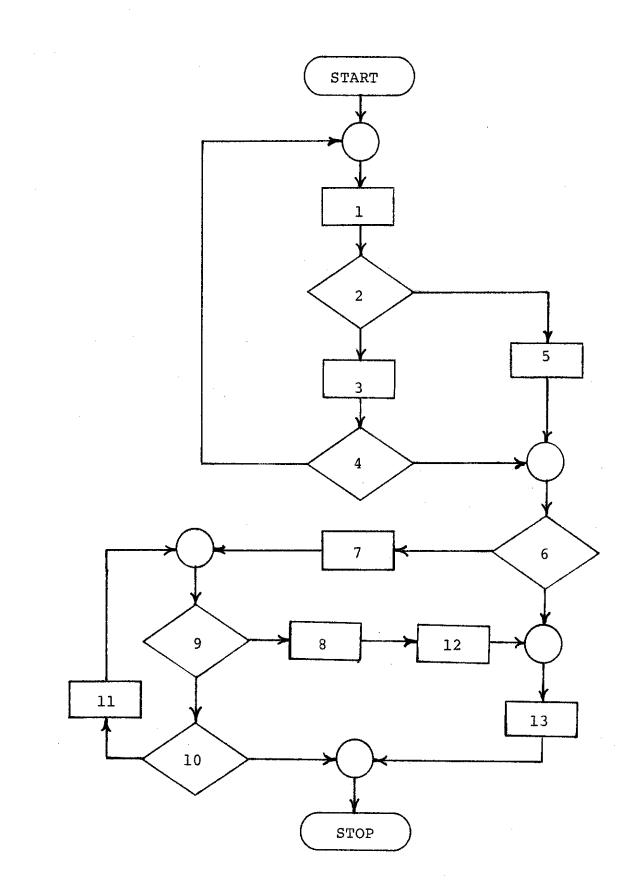


Fig. 10--Improper use of control structures

The first situation occurs when a multivalued selective structure, as shown in Figure 11, is required. Such a situation arises when only one of the three operations is to be performed, depending on whether a variable is less than, equal to, or greater than zero. A control structure implemented for this use is the FORTRAN computed GOTO statement.

To implement a multivalued selective structure using only the three structures of Bohm and Jacopini, a selective structure must be nested within another selective structure, as in Figure 12. Compared to the computed GOTO, this nested form of the selective structure may be grossly inefficient and less readable since it requires an unnecessary ordering of decision statements.

The second instance of inefficiency occurs when the abnormal termination of a repetitive structure is required. This situation is illustrated in Figure 13. Although this structure violates the single entry and single exit rule, Donaldson (4, p. 53) suggests that such a structure may save considerable time and space. Since this structure contains two possible loop-terminating conditions, its use should be properly flagged.

The three basic control structures can usually be approximated in most higher languages. They are directly implemented in PL/I with the IF THEN ELSE and DO constructs, in ALGOL with the IF THEN ELSE and the FOR constructs and

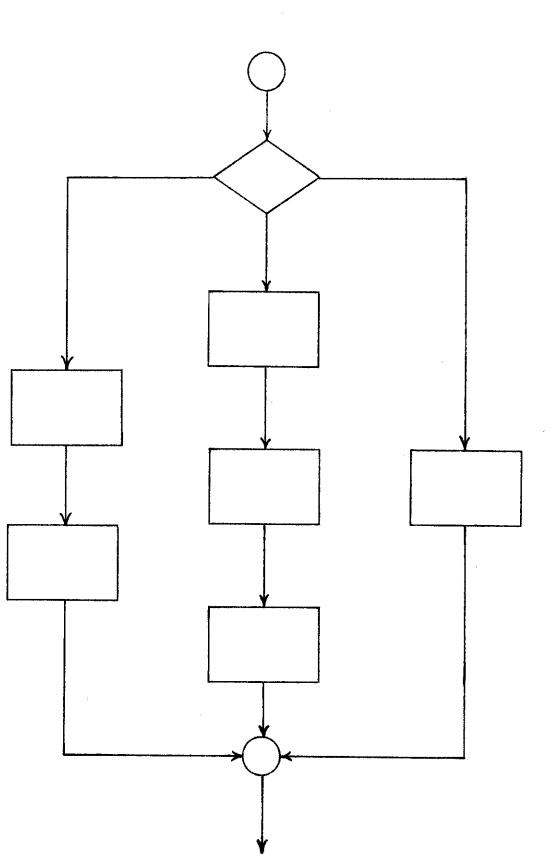


Fig. 11--Multivalued selective structure

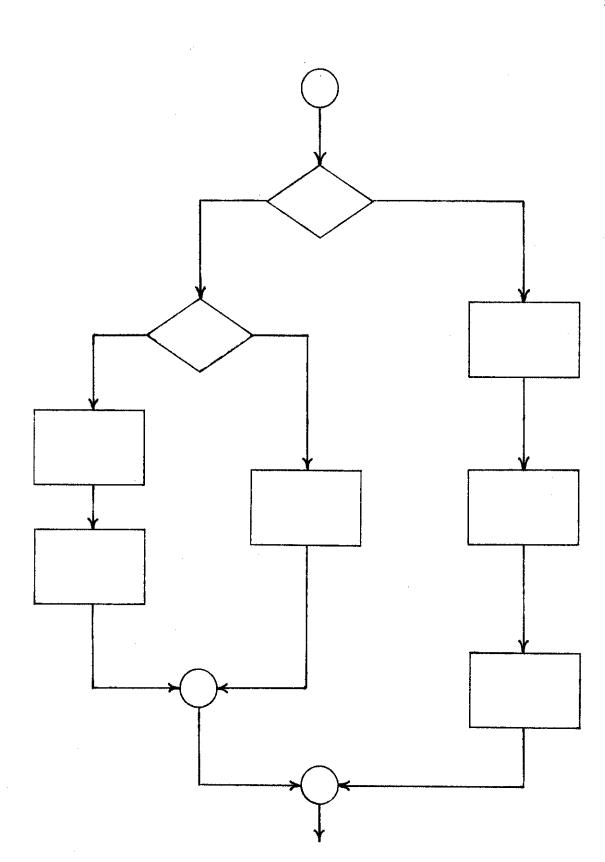


Fig. 12--Multivalued selective structure implemented by nesting simple selective structures.

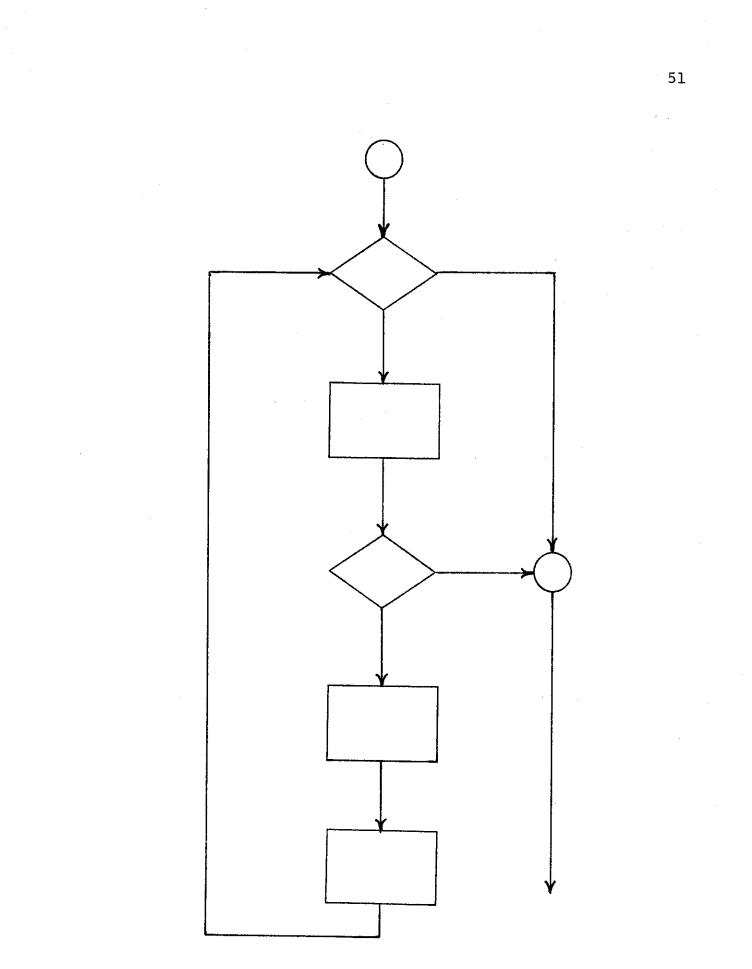


Fig. 13--Abnormal termination of a repetitive structure.

in COBOL with the IF THEN ELSE and the PERFORM constructs. FORTRAN has an IF statement which can approximate a selective structure and a DO construct for repetition (13, p. 111).

Assembly languages do not possess IF statements or DO constructs, but these structures can be approximated if macro processing is available. The use of macros to implement these constructs will make the structure of an assembly language program more visible since macro instructions tend to stand out among other assembly language instructions. Without macros, however, assembly language programming is limited to a few conditional and unconditional branching mechanisms which can only simulate selective and repetitive structures with a great deal of awkwardness, and which tend to obscure program structure.

Significance of Simple Control Structures

The major advantage of limiting the programmer to the three sufficient control structures is that it forces him to design more carefully (1, p. 146). With simplified logic he must take care in deciding how to code iterative and selective procedures and make sure that termination conditions are correct. Forcing the programmer to be careful will increase the likelihood of a correct program.

The programmer need not be concerned with a vast repertoire of program structures if he is limited to three.

He need only decide how to combine these three to satisfy his problem. With such simple logic his programs will more likely be understood. Diagrams, such as Figure 7, can be used to illustrate program logic and program decomposition.

Since each of the three structures maintains a single entry and exit point, the program lends itself to segmentation and modularity. Such programs are more readily verified since testing can be done on program segments rather than on the program as an entity.

If only the three structures are used, program notation can be greatly simplified and easily understood. A notation such as the following:

```
DO
.
.
End DO
```

and

IF . . . .

THEN END THEN ELSE END ELSE

will suffice to define most program logic. Programs will not contain complicated or vague structuring mechanisms which tend to obstruct readability and clarity. If the single

entry and exit rule is maintained while using this notation, the program can be read in a linear fashion making program logic easier to follow.

While these three structures are sufficient to code any programmable logic, it has been pointed out (4) that they may be inefficient in some cases. Limitations on code size or execution time may force the programmer to use other structures to improve code efficiency. When there are not, however, strict limitations on code size and execution time, the use of this simple set of structures should provide overall efficiency, with respect to program readability, accuracy, and maintenance. Although a program may require more time to design and may be less efficient in terms of code usage and execution time, the debugging time will be considerably less and program maintenance considerably easier. A certain amount of assurance in the quality of the program can be implied if the constructs used are simple (1).

The use of a simple but sufficient set of control structures can improve almost every aspect of program design and coding. Program features most likely to be negatively affected by these structures include: time required for program design since the designer is limited by the logic structures available to him, and program length and execution time since the set of structures are sometimes inefficient.

The benefits of using the simple structures seem to outweigh their disadvantages. The author believes that the use of a simple set of structures will improve most programs especially in terms of functional correctness and program readability. The use of additional constructs should be avoided whenever possible, but when they are utilized, care should be taken to insure that their presence has been properly documented. It is much too easy for a program to become unnecessarily complex when a variety of constructs are used indiscriminately by a careless programmer.

# The GOTO Controversy

In regard to the use of other control structures in programming besides the three discussed above, much attention has been directed to the use of GOTO statements. Dijkstra (3) and Mills (9) are just two of the many authors who have directly attacked the use of GOTO constructs in programming. Hopkins (6), on the other hand, suggests that while the GOTO is commonly abused in programming today, it is not necessary or even desirable to discontinue its use in computer languages.

In an effort to define a structured programming technique, the elimination of GOTO constructs is sometimes presented as the only issue (8, p. 51). Indeed, Hopkins (6, p. 55) suggests that the GOTO issue has been greatly over-emphasized. Programmers looking for a simple solution to the problems of programming today have used the GOTO as a scapegoat. While Hopkins (6) agrees that the removal of GOTO's will improve the code of most programmers, he does not consider this sufficient grounds for eliminating the structure from programming.

Removal of the GOTO statement is suggested mostly on the arguments of programming clarity and simplicity of logic. Most studies expound methods of improving GOTO programs by the substitution of other control structures; few, however, discuss the cases where these other structures will themselves produce inefficient code or perhaps some other undesirable side effect. Attention has been focused on the elimination of the GOTO statement and, for the most part, has ignored the problems which programming without the GOTO may introduce.

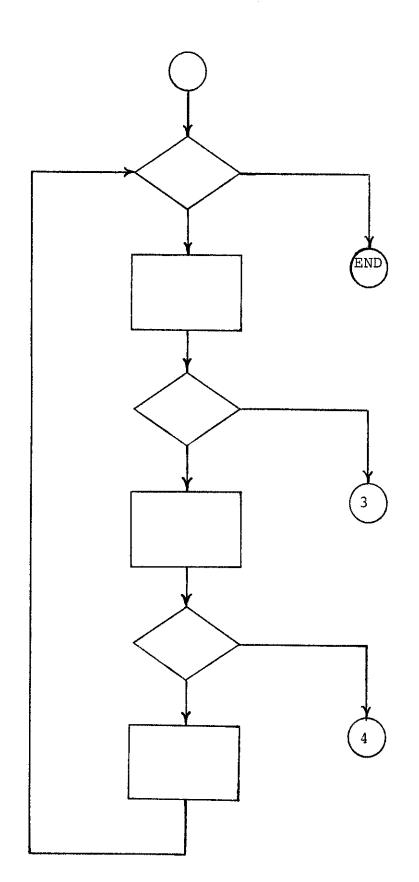
One of the major criticisms of the GOTO structure is its use in multiple exits to different program locations from within a loop. Consider the following FORTRAN code whose structure is diagrammed in Figure 14.

> . DO 1 I=1,50 X = X + 1. Y = Y + 1.

IF (M .EQ. 10) GO TO 3 M = Y \* 2IF (M .EQ. 200) GO TO 4 T = T + 1.1 CONTINUE ... 4 .....

The loop contains three possible exit points, each of which branches to a different location in the program, and consequently, produces a somewhat complicated execution sequence. When the loop has terminated, program control will be at one of three possible locations, and the problem of determining which location may be significant. If the programmer is accustomed to unrestricted use of the GOTO, loops similar to the one just described may be common in his programs.

While it is desirable to avoid this type of logic, eliminating the GOTO will not necessarily prevent multiple exits to locations outside the control of the looping structure. The following PL/I code produces a multiple-exit loop, yet it contains no GOTO structures. Here the programmer has used the RETURN statement just like a GOTO, i.e., to escape the control of the loop and branch to a location outside the range of the loop. Thus, merely



Č,

Fig. 14--A DO loop with multiple exits

eliminating GOTO statements will not prevent awkward control paths or complex logic structures.

Multiple loop exits will not necessarily complicate the logic structure. If the exits all branch to the end of the loop, then the block structure which the loop represents is maintained. In effect, the loop has executed or partially executed some number of times and control passes to the location just past the loop.

On occasion a programmer will discover that a large code segment is not functionally correct, and with the addition of a GOTO statement the program can be made to execute correctly. In this case the insertion of a GOTO may save a considerable amount of time, especially if major code alteration is required to correct the mistake without using the GOTO; moreover, if a functional program were required in a short time, the programmer may be forced to use the GOTO since major code revision is usually timeconsuming. If used with discretion then, the GOTO may be an appropriate means of solving an awkward logic problem. Also since some programs are used only once or are used only by the programmer himself, the need for simplicity or clarity in a program may not be felt; the programmer may see no justification for coding without the GOTO. Although every program can be written without the use of GOTO statements, not every program need be so written.

The program which must maintain strict limits with regard to code efficiency or length may be possible only with the use of GOTO statements. While this situation should be avoided whenever possible, the fact that it does occur suggests that, from a practical point of view, the GOTO construct may be quite useful.

Some of the reasons for eliminating the GOTO have already been mentioned. They include: destruction of program linearity when GOTO constructs are freely used; increase in program complexity due to GOTO branches; and the decrease of program modularity, clarity, and readability due to the presents of GOTO constructs.

The strongest argument against the GOTO statement is the fact that there are so many ways to use it incorrectly. The presence of a GOTO is more likely to be the result of poor programming techniques than it is a sound logic structure. Since the GOTO is not needed and since it is

so apt to be misused, there is good reason to desire its removal.

It is suggested by Wulf (14, p. 68) that efficiency should not be used as an excuse for GOTO programming. Efficiency of program code should be achieved through a highly optimizing compiler; the programmer should not have to resort to introducing GOTO constructs for the sake of improving run time. Code optimization during compilation is more effective if the programmer has used a blockstructured program design and has avoided the use of GOTO structures which unnecessarily disrupt program modularity.

Programming with GOTO's necessitates the use of labels, a factor which adds to program complexity since a label referenced in a program instruction requires a search through the source code to locate the label and interpret the referencing instruction. One of the basic reasons for developing a structured programming style is to achieve a certain degree of program simplicity. Removing the GOTO's will reduce the need for labels, and hopefully, simplify program logic by reducing the number of label references.

The author is of the opinion that programming without the GOTO in most cases will improve the quality of a program; however, to completely eliminate the GOTO seems to be unjustified at this time. It should be remembered

that there are circumstances where the GOTO is useful even though it may produce awkward code or undesirable program logic. Although it lends itself to misuse, the GOTO is not unique in this respect. Many features of a language may be subject to improper or inefficient use, yet it would be impractical to remove all of them.

## Subroutines

Another control structure which deserves special attention is the subroutine call. Like the GOTO statement the subroutine call causes an unconditional branch. The branch passes control from the main program to an external set of instructions, referred to as the subroutine or subprogram, which are then executed. Once the subroutine has been executed, program control returns to some location in the main program.

The use of the subroutine call does not disrupt the structure of a repetitive, selective, or sequential construct within the main program when the subroutine satisfies two conditions. First, if the subroutine returns control to the statement immediately following the subroutine call, then linearity has been preserved. Three sequential statements will have been executed: these statements are the one preceding the call, the subroutine call, and the statement following the subroutine call. The second

condition requires that the subroutine itself consist of some combination of the three simple control structures.

Some subroutines allow multiple entry and exit points in addition to multiple return locations. The legality of their use in terms of structured programming control structures seems to be unclear at this time since most authors, when discussing control structures of structured programming, do not mention subroutines specifically. Program modules is a frequently-used term which may or may not include subroutines. Mills (10), when discussing control structures, uses the term segments. He suggests (10) that program segments should have only one entry and one exit point, but it is unclear whether or not he is applying this rule to subroutines as well.

It seems reasonable to assume that the concepts of control structures discussed in this chapter would apply directly to subroutines. The subroutine is a block of code that is represented in a program by a single instruction, the subroutine call. The subroutine, when inserted in place of the call, should produce a logic structure conforming to the principles of structured programming.

### Conclusions

The theory of control structures plays a key role in the development of a structured programming technique;

however, control structures alone will not provide the solution to the present software dilemma. All aspects of programming, including data structures, must be investigated to develop the most effective approaches to program design and constructions.

# CHAPTER BIBLIOGRAPHY

- Benson, Jeoffrey, "Structured Programming Techniques," <u>Record of the 1973 IEEE Symposium on Computer</u> <u>Software Reliability</u>, New York (April, 1973), 143-147.
- Bohm, C. and G. Jacopini, "Flow Diagrams, Turing Machines and Languages with Only Two Formation Rules," <u>Communications of the ACM</u>, IX (May 1966), 366-371.
- Dijkstra, E. W., "GOTO Statement Considered Harmful," <u>Communications of the ACM</u>, XI (March, 1968), <u>147-148.</u>
- Donaldson, James, "Structured Programming," Datamation, XIX (December, 1973), 53-54.
- Fisher, David, "A Survey of Control Structures in Programming Languages," <u>SIGPLAN Notices</u>, (November, 1972), 1-13.
- 6. Hopkins, Martin, "A Case for the GOTO," <u>SIGPLAN</u> Notices, (November, 1972), 59-62.
- Leavenworth, B. M., "Programming With(out) the GOTO," SIGPLAN Notices, (November, 1972), 54-58.
- McCracken, Daniel, "Revolution in Programming: An Overview," <u>Datamation</u>, XIX (December, 1973), 50-52.
- 9. Mills, H. D., <u>How to Write Correct Programs and Know</u> <u>It</u>, FSC 73-5998, IBM, Gaithersburg, Maryland, (December, 1972).
- 10. , "On the Development of Large Reliable Programs," Record of the 1973 IEEE Symposium on Computer Software Reliability, New York (May, 1973), 155-159.
- 11. , <u>Mathematical Foundations for Structured</u> <u>Programming</u>, IBM FSD Report FSC 72-6012, IBM Gaithersburg, Maryland, (February, 1972).

- 12. Peterson, W. W., et al., "On the Capabilities of While, Repeat, and Exit Statements," <u>Communications of</u> <u>the ACM</u>, XVI (August, 1973), 503-512.
- Tenny, Ted, "Structured Programming in FORTRAN," <u>Datamation</u>, XX (July, 1974), 110-115.
- 14. Wulf, William, "A Case Against the GOTO," <u>SIGPLAN</u> <u>Notices</u>, (November, 1972), 63-69.

### CHAPTER IV

# CONTROL STRUCTURES IMPLEMENTED WITH MACROS FOR ASSEMBLY LANGUAGE PROGRAMMING

In IBM 360 assembly language programming (7) there are no instructions which will provide for the documentation and execution of selective and repetitive control structures described in the previous chapters. Like most assembly languages the IBM 360 language contains only a set of conditional and unconditional branching instructions of which the most sophisticated include the BXLE, BSH, and BCT instructions (6, pp. 66-67). With the use of labels and these branching instructions available to him, the programmer may simulate in ALC the selective and repetitive structures he needs. The additional instructions required for implementing these structures, however, may significantly increase the complexity of the code and reduce the clarity of the program. Because there are no adequate control structures available in ALC, the programmer is forced to improvise looping and selective structures each time they are required in a program.

A set of IBM 360 assembly language macros to be used as selective and repetitive control structures were developed in this study. These macros should provide at

least two important benefits to the ALC programmer. First, the burden of formulating such control structures when they are required is removed from the programmer; he needs only supply the necessary symbolic parameters for the macro which will generate the desired control structure.

The second important benefit which the macros provide is an increase in program clarity. It is readily apparent to anyone reading the code that the macro prototype statements represent looping and selective structures. The mere presence of the macro names within the code indicates the extent of control of each structure. If the NOGEN print option is used, the program source listing is significantly simplified since macro-generated code is not shown. The NOGEN option causes the suppression of print-out of all macro-expanded code. The source listing includes only ALC instructions and macro instructions as they were coded by the programmer. When the NOGEN print option is omitted and the source listing contains the expanded macro code, the program, of course, looks much like a typical assembly language program, since all of the branching instructions and labels necessary to implement the macros are shown. The Appendices contain examples of programs with unexpanded code and programs with expanded code. The differences will be readily apparent to the reader.

### Repetitive Control Structure Macros

To implement a looping or repetitive control structure in ALC two macros, DO and ENDDO, have been developed. The DO macro generates the code for testing the looping condition and the code which either continues the execution of the loop or branches to terminate the loop. The ENDDO macro besides providing a label to which a branch will occur when the loop is terminated, signifies the extent of control of the DO macro.

The DO macro is similar to the DO statement in PL/I. It may be specified with just an increment and range such as DO I = 1 TO 10 BY 3, with just a WHILE condition such as DO WHILE (X LE 2) or with both a looping range and WHILE condition such as DO I = 1 TO 100 BY 3 WHILE (Y GT 30). These options are specified by means of keyword parameters attached to the macro prototype statement.

The looping conditions may be specified by literals when parameters are equated to numeric values, in registers when parameters are equated to register numbers, or by locations when parameters are equated to labels. Numeric values may not be used to specify locations. Once the DO loop parameters are determined, they are maintained in a stack generated by the macro; the programmer need not be concerned with destroying looping variables initially specified in registers or locations. He is free to use registers and locations originally containing looping

parameters as he wishes without affecting the state of the DO loop parameters. Table I contains an explanation of each DO macro parameter.

### TABLE I

# KEYWORD PARAMETERS FOR THE DO MACRO

Keyword Parameter	Meaning	Characteristics of Value
LABEL	Label to identify corresponding ENDDO macro.	Any valid ALC label.
LOWNUM	Value of the lower range of a DO loop.	Integer.
BYNUM	Value of increment of a DO loop range.	Integer.
HGHNUM	Value of the upper range of a DO loop.	Integer.
LOWREG	Register containing the lower range of a DO loop.	Integer from 0 to 15.
BYREG	Register containing the incre- ment of a DO loop range.	Integer from 0 to 15.
HGHREG	Register containing the upper range of a DO loop.	Integer from 0 to 15.
LOWLOC	Location containing the lower range of a DO loop.	Any valid ALC label.
BYLOC	Location containing the incre- menet of a DO loop range.	Any valid ALC label.
HGHLOC	Location containing the upper range of a DO loop.	Any valid ALC label.
DOLOOP	Indicates the presence of a DO loop.	Any non-null value
WHILE	Indicates the presence of a DO WHILE condition.	Any non-null value.

ţ,

TABLE I--Continued

Keyword Parameter	Meaning	Characteristics of Value
WOP	Logical operator for a DO WHILE condition.	LT,LE,EQ,GE, GT, or NE.
WREGA	Register containing the left operand of a DO WHILE condition.	Integer from 0 to 15.
WREGB	Register containing the right operand of a DO WHILE condition.	Integer from 0 to 15.
WLOCA	Location containing the left operand of a DO WHILE condition.	Any valid ALC label.
WLOCB	Location containing the right operand of a DO WHILE condition.	Any valid ALC label.
WLITA	Numerical value representing the left operand of a DO WHILE condition.	Integer.
WLITB	Numerical value representing the right operand of a DO WHILE condition.	Integer.

The ENDDO macro utilizes only one keyword parameter LABEL. This parameter should be equated to the value of the LABEL parameter of the DO macro associated with the ENDDO macro.

Examples of the DO macro are shown in Figure 15. Macro A represents a simple DO loop. The lower range is specified in register three (LOWREG=3), the upper range is a numerical

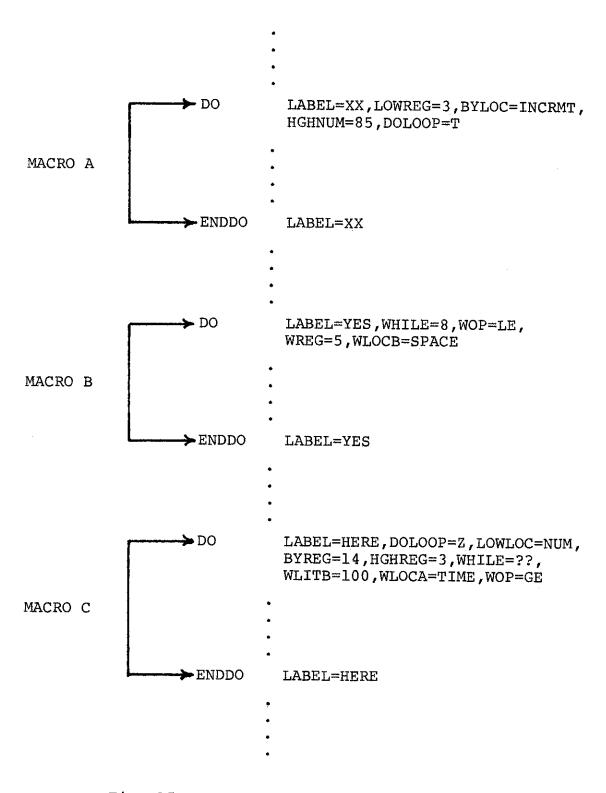
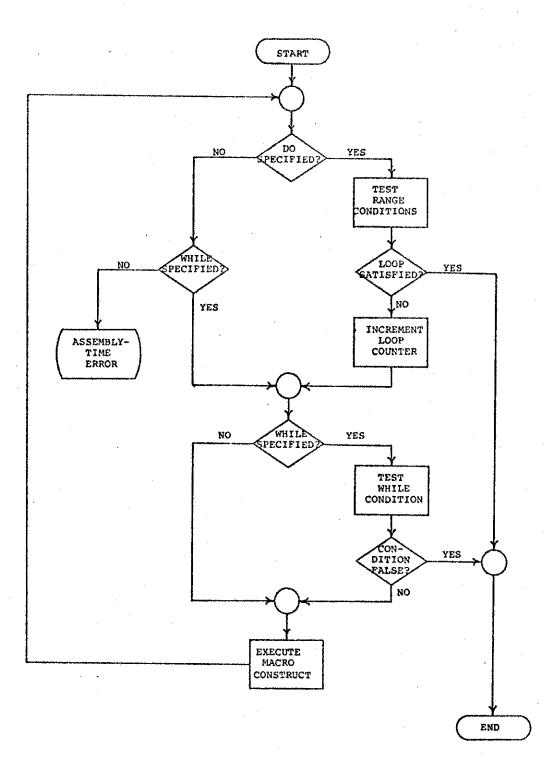


Fig. 15--Examples of DO macro parameters

eighty-five (HGHNUM=85), and the increment for the loop is contained in location INCRMT (BYLOC=INCRMT). Note that the DOLOOP parameter has been made non-null. This is required when a DO loop is specified. Macro B represents a simple DO WHILE loop. The loop will be executed as long as the contents of register five (WREGA=5), is less than or equal to (WOP=LE), the contents of location SPACE (SLOCB=SPACE). The WHILE parameter must be made non-null when a DO WHILE loop is specified. Macro C represents a combination of a DO loop and DO WHILE condition. In this macro both the DOLOOP and WHILE parameters must be made non-null. The lower limit of the DO loop is contained in location NUM, (LOWLOC=NUM), the upper limit is in register three (HGHREG=3), and the increment is in register fourteen (BYREG=14). The WHILE condition is satisfied when the contents of location TIME (WLOCA-TIME), is greater than or equal to (WOP=GE), a numerical one hundred (WLITB=100). When both a DO loop and DO WHILE condition are specified, each is tested before the loop is executed. The basic logic of a DO and ENDDO construct is shown in Figure 16.

Since a stack is used to maintain DO loop parameters, the DO macro permits nesting. A DO macro and its ENDDO macro may be entirely contained within the bounds of a second DO and ENDDO macro. Figure 17 illustrates nesting of DO macros to the second level. Notice that the LABEL parameters of the DO and ENDDO macros must be properly



## Fig. 16--Flowchart of a DO and ENDDO construct

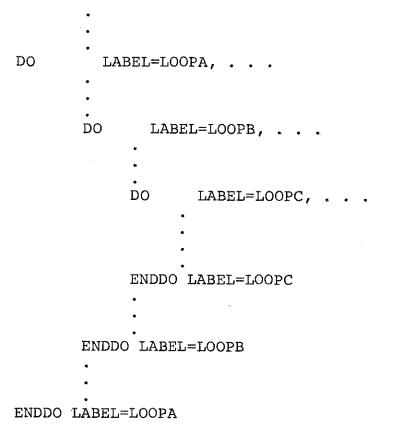


Fig. 17--DO macros nested to the second level

matched, i.e., at any given level of nesting, the LABEL parameters of the DO and ENDDO macros are equated to identical values. (A unique LABEL parameter value should be specified for each DO construct.)

Figure 18 illustrates the use of the stack in executing the nested macros of Figure 17. As each DO macro is encountered its parameters are stacked in a last-in first-out stack. Before executing a particular DO loop, its parameters are unstacked and tested. If the loop is to be executed, then the parameters are restacked; otherwise, the parameters are not restacked, and the loop is bypassed. Thus, each

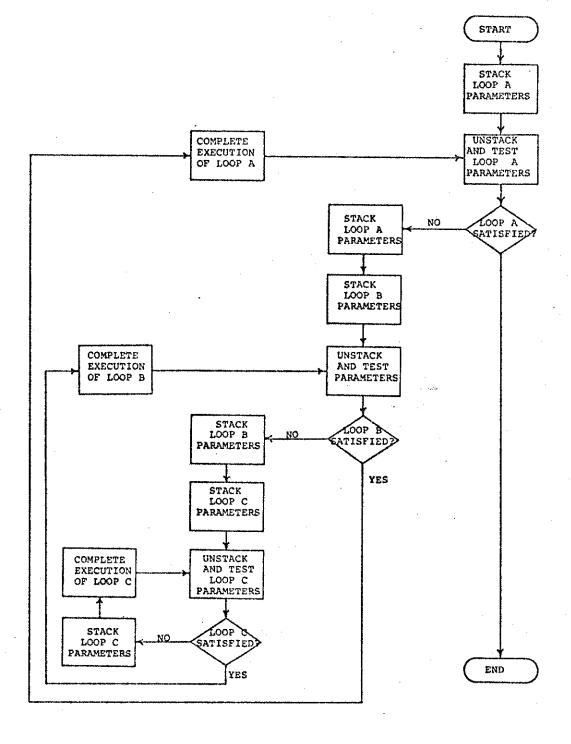


Fig. 18--Flowchart of stack logic for code in Figure 17

time LOOP A is executed the DO macro of LOOP B will be encountered, its parameters stacked, unstacked, and tested. In a similar manner, with each execution of LOOP B, the DO macro of LOOP C will be encountered, and its parameters will be stacked, unstacked, and tested.

Program instructions falling between a DO macro and its corresponding ENDDO macro represent the body of the code affected by the looping construct. This code may include any acceptable ALC instruction including macro instructions; however, the loop should not include branching instructions which transfer program control to a point outside the range of the DO macro. If the programmer wishes to terminate the looping construct abnormally, he should use the LEAVE macro, which is discussed in this chapter.

The DO macro extensively tests parameter specifications at assembly time. When an incorrectly specified parameter is detected, an appropriate error message is printed if the expanded macro code is shown, and the macro is terminated. Before termination of the macro, however, the "work" registers will be restored with their original contents so that program execution may continue. The macro, however, will have essentially been ignored by the program.

In addition to the complete source listings of the DO and ENDDO macros, Appendix A contains an expanded version of each of these macros and sample listings of macrogenerated error messages.

This particular form of the repetitive control structure was selected for implementation because it offers the greatest flexibility in defining looping conditions. А loop may be defined with a simple range condition similar to the FORTRAN DO statement, with a simple comparison condition similar to a PL/I DO WHILE or with a combination of both a range condition and comparison condition similar to the PL/I statement, DO I=...TO...WHILE(.....). Since conditional tests are always executed prior to loop execution, a DO UNTIL structure as illustrated in Figure 7 cannot be generated with the DO and ENDDO macros. The DO UNTIL statement tests looping conditions after the loop is executed; consequently, the loop is always executed at least once even when a condition fails on initial testing. In order to avoid any ambiguity with regard to initial loop execution, a DO UNTIL construct was not implemented. All repetitive loops will be executed only after their looping conditions are tested. If the conditional test fails initially, the loop will not be executed at all.

### The LEAVE Macro

It was pointed out in Chapter III that occasionally it may be expedient to abnormally terminate a looping construct, i.e. branch to the end of the loop from a point within the loop control. Figure 13 is a construct representing an abnormal loop termination.

The DO and ENDDO macros generate instructions that will terminate a loop only when the inclusive code has been completely executed. The programmer might insert a branch instruction within the inclusive code to escape the DO macro, but such a practice is not advised, since the effects of the branch instruction may not be readily understood or perhaps may be overlooked by a programmer unfamiliar with the code.

To provide a loop escape mechanism the LEAVE macro has been developed. It permits a conditional or unconditional branch to the end of the ENDDO macro, and thus, the looping control structure will be abnormally terminated.

Figure 19 illustrates the use of the various LEAVE macro parameters. In the first LEAVE macro LABEL is equated to AAAA in order to associate it with the external DO and ENDDO macros. This LEAVE macro contains a conditional branch since the COND parameter has been made non-null, COND=F. If the number nine (LITA=9) is equal to (OPRATOR=EQ) location Y (LOCB=Y) then the LEAVE condition will be true and a branch to location AAAA, (LABEL=AAAA) will occur. The second LEAVE macro does not specify a condition and will, consequently, produce an unconditional branch to AAAA (LABEL=AAAA) when executed. The third LEAVE macro contains a conditional branch (COND=;), and the branch is executed if the contents of register ten

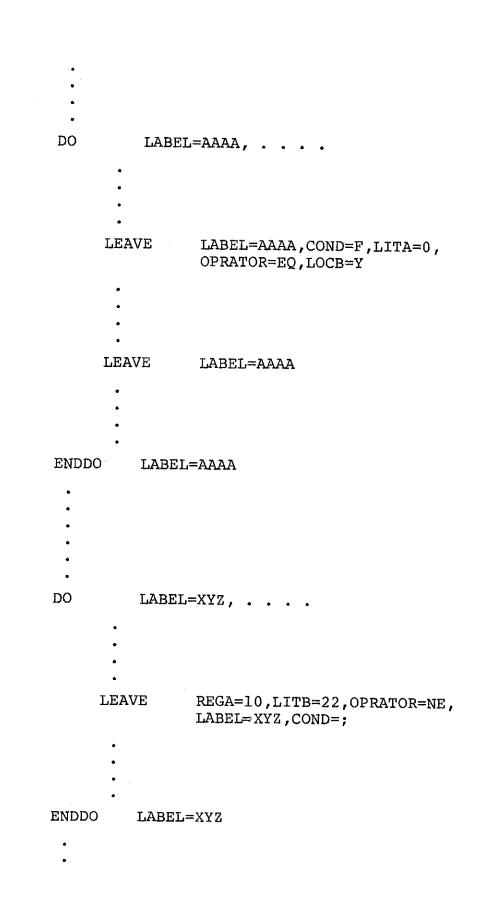


Fig. 19--Examples of LEAVE macro parameters

(REGA=10) are not equal (OPRATOR=NE) to a numeric twenty-two (LITB=22). The LABEL parameter (LABEL=XYZ) specifies the destination of the LEAVE branch. Table II contains a summary of LEAVE parameters, and Figure 20 describes the basic logic of the LEAVE macro at execution time.

### TABLE II

## KEYWORD PARAMETERS FOR THE LEAVE MACRO

Keyword		
Parameter	Meaning	Characteristics of Value
LABEL	Destination of a LEAVE macro producing a branch.	Any valid ALC label.
COND	Indicates the LEAVE branch is conditional.	Any non-null value.
OPRATOR	Logical operator for a condi- tional branch.	LT,LE,EQ,GE, GR, or NE.
REGA	Register containing the left operand of LEAVE condition.	Integer from 0 to 15.
REGB	Register containing the right operand of LEAVE condition.	Integer from 0 to 15.
LOCA	Location containing the left operand of LEAVE condition.	Any valid ALC label.
LOCB	Location containing the right operand of LEAVE condition.	Any valid ALC label.
LITA	Numerical value of the left LEAVE condition operand.	Integer.
LITB	Numerical value of the right operand of LEAVE condition.	Integer.

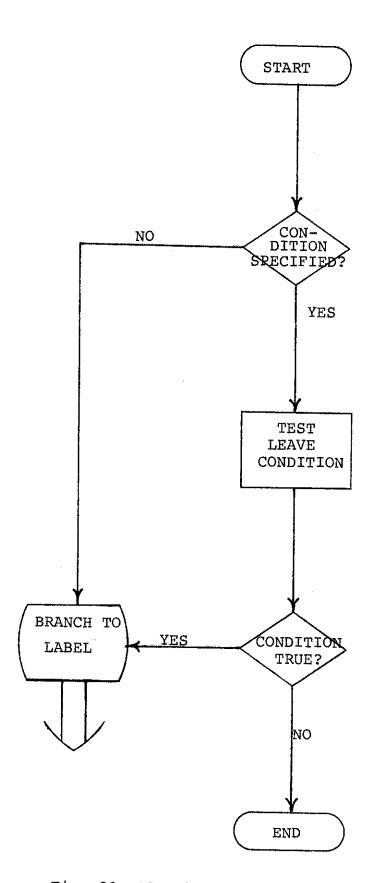


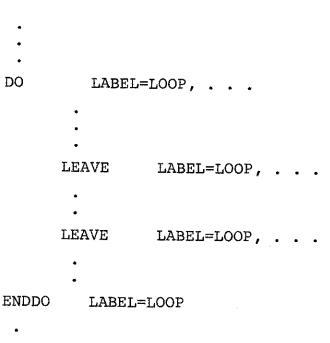
Fig. 20--Flowchart of execution of a LEAVE macro

If the LEAVE macro is used in lieu of a branching instruction, there can be no doubt as to the intentions of the code. The LEAVE statement appearing in the code will readily indicate the presence of a possible abnormal loop exit. Since the LEAVE macro may be specified with condition parameters, the programmer need not include the additional instructions to test his branching condition. He need merely specify the branching conditions by means of the keyword parameters; thus, the development of the LEAVE macro provides the programmer with a convenient and simple method for abnormal loop termination.

While it is possible to specify a LEAVE macro which will branch to a point other than just beyond the control of the looping macro, this practice should be avoided as explained in Chapter III since it may significantly complicate the program's structure.

Figures 21 and 22 illustrate the proper and improper use of the LEAVE macro. Notice that when coded properly the DO and ENDDO LABEL parameters are identical to each other and to any inclusive LEAVE macro LABEL parameters.

The LEAVE macro also tests parameter values at assembly time, and when improperly specified, the LEAVE macro will be ignored at execution time. Like the DO macro not all parameters of the LEAVE macro can be verified at assembly time; macro specification errors may, consequently, cause system-generated errors at execution time.



•

Fig. 21--Properly coded LEAVE macros

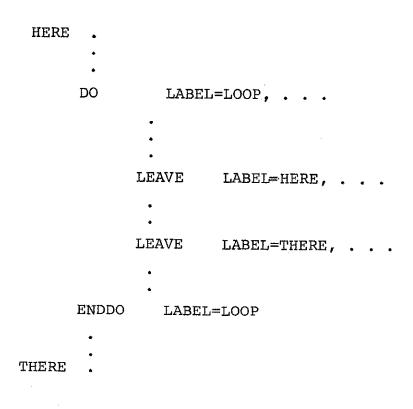


Fig. 22--Improperly coded LEAVE macros

Appendix B contains a source listing of the LEAVE macro, an example of the expanded macro code, and samples of macro-generated error messages.

### Selective Control Structure Macros

A selective control structure, the CASE macro, for ALC has been developed in this study. The CASE macro provides a multiple-selective structure for the programmer so that alternative sets of program instructions may be executed depending on the truth of the conditional parameters specified with each CASE macro.

Each CASE macro must be followed by an ENDCASE macro. All program instructions placed between the CASE and ENDCASE macros represent the code which will be executed if the CASE condition is true.

The CASE macro generates code to test the macro condition which is specified with keyword parameters similar to those which are used with the LEAVE macro. If the condition is true, instructions immediately following the CASE macro are then executed. If the condition is false, a forward branch to the paired ENDCASE macro will occur. The basic logic of a CASE macro is flowcharted in Figure 23.

The ENDCASE macro may be specified in one of two ways. The first way, OPTION=1, will cause a branch around any successive CASE macros if the preceding CASE condition were

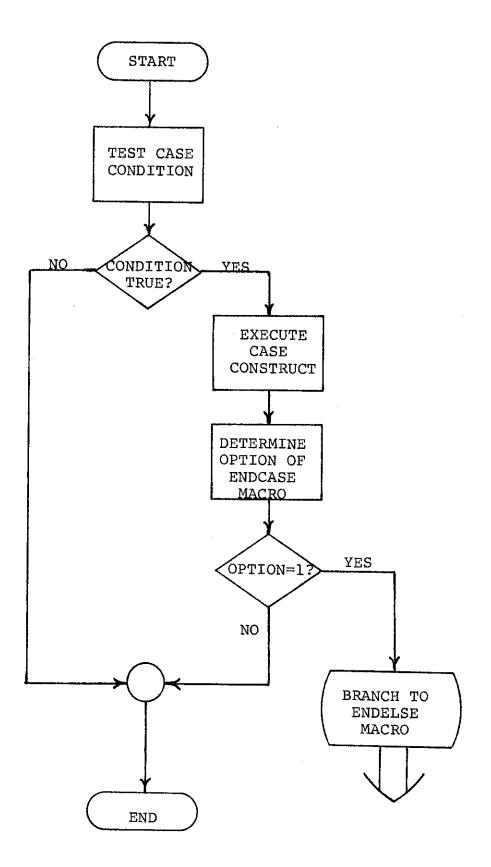


Fig. 23--Flowchart of the execution of a CASE and ENDCASE construct.

.

true. In other words, when several successive CASE and ENDCASE macros are coded with OPTION=1, the first encountered CASE construct whose condition is true will be executed. All subsequent CASE macros will be bypassed. If OPTION=2 is specified with successive CASE macros, each of the macro constructs with true conditions will be executed.

In order to provide an alternative set of instructions to be executed when all of the immediately preceding CASE conditions are false, the ELSE and ENDELSE macros have been developed.

The ELSE macro is merely a prototype statement to signify the beginning of the alternative code. The ENDELSE macro provides a label for branching purposes when the instructions included between the ELSE and ENDELSE macros are not to be executed. The ENDELSE prototype statement indicates the termination of both the selective CASE macros and the ELSE macro.

An example of several CASE and ENDCASE macros using OPTION=1 is shown in Figure 24. The conditional parameters for a CASE macro are identical to those of the LEAVE macro with the exception of the COND parameter. The CASE macro is always conditional so a COND parameter is not required. The ENDCASE macros contain three parameters. One of the parameters is the OPTION keyword which in all cases has been equated to one. This indicates that as soon as one of the

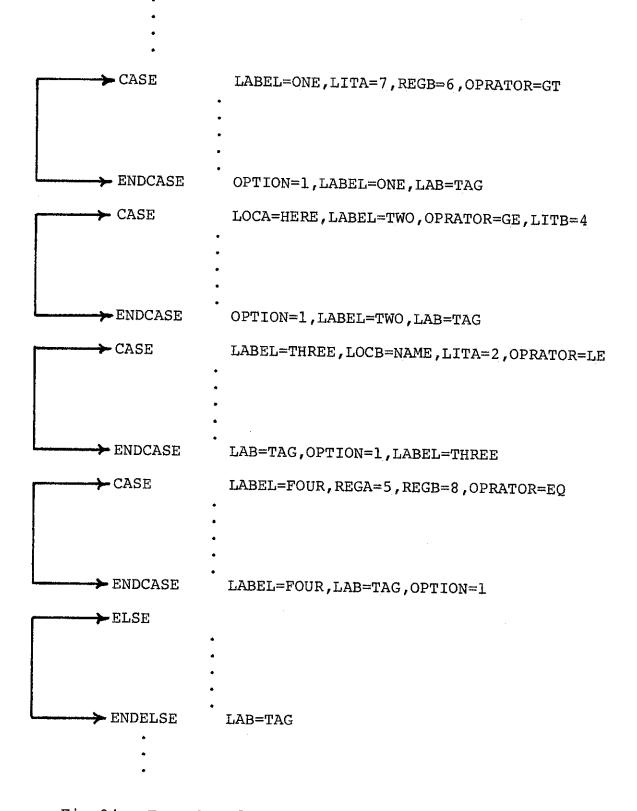


Fig 24.--Example of CASE macros using OPTION=1

CASE conditions is true and its inclusive code is executed, the remaining CASE constructs along with the ELSE construct will be bypassed. In other words at most only one of the four constructs will be executed, namely the first one encountered whose condition is true. If all of the CASE conditions are untrue, the ELSE construct will be executed. The LABEL parameters associate each ENDCASE macro with its corresponding CASE macro. The LAB parameters specify the location of the ENDELSE macro associated with the CASE macros. Notice that each ENDCASE macro has its LAB parameter equated to TAG. This is necessary since a branch to the same ENDELSE macro will occur if any one of the CASE constructs is executed.

Figure 25 is a set of CASE and ENDCASE macros similar to those in Figure 24 except for the specification of the OPTION parameters. In Figure 25 each CASE construct has OPTION=2 specified. Since the second option has been specified, each CASE construct will be tested and executed if its condition is true. There is no ELSE construct included since forward branching does not occur when the second option is used.

The programmer is at liberty to use any combination of OPTION=1 and OPTION=2 CASE constructs that he wishes. Since the CASE and ENDCASE macros never produce backward branching, program linearity is not destroyed when the two options are used in successive CASE constructs. It is

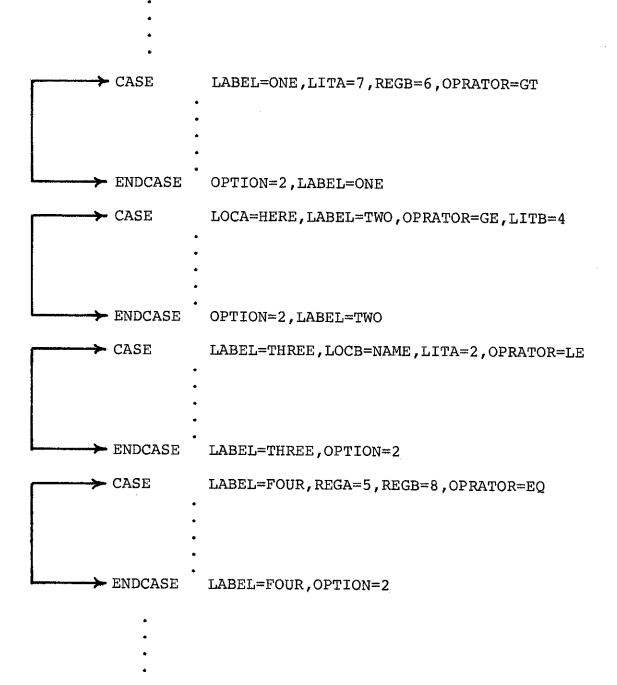


Fig. 25--Example of CASE macros using OPTION=2

important to remember that when OPTION=1 is specified, an ELSE construct must also be included. Tables III and IV summarize the various parameters which may be specified with the CASE and ENDCASE macros.

Both the CASE and ENDCASE macros contain a number of error detection instructions. For the most part, macro keyword parameters are checked to insure that they have been coded correctly. If a CASE macro has been incorrectly specified, its condition is set to false. When ENDCASE macros have been incorrectly specified, the OPTION parameter is set to one and the macro is accordingly expanded.

The source listings for the CASE, ENDCASE, ELSE, and ENDELSE macros are in Appendix C, which also contains an expanded version of each of the macros and samples of macro-generated error messages.

The CASE macro as a selective control structure was implemented because of its versatility with respect to path selection. Any number of CASE constructs may be successively defined; thus, there is no limit to the number of possible paths from which the programmer may choose. To the contrary the PL/I IF THEN ELSE statement, which is a form of the selective control structure, limits the programmer to one of two possible paths unless nesting is used. The CASE macro also allows the programmer to execute more than one of the tested paths if he so desires. This is possible when OPTION=2 is specified. The CASE macro is

### TABLE III

## KEYWORD PARAMETERS FOR THE CASE MACRO

Keyword Parameter	Meaning	Characteristics of Value
LABEL	Label to identify corresponding ENDCASE macro.	Any valid ALC label.
OPRATOR	Logical operator for the CASE condition	LT,LE,EQ,GE, GT,or NE.
REGA	Register containing the left operand of the CASE condition.	Integer from 0 to 15.
LOCA	Location containing the left operand of the CASE condition.	Any valid ALC label.
LOCB	Location containing the right operand of the CASE condition.	Any valid ALC label.
LITA	Numerical value of the left operand of the CASE condition.	Integer.
LITB	Numerical value of the right operand of the CASE condition.	Integer.

## TABLE IV

## KEYWORD PARAMETERS FOR THE ENDCASE MACRO

Keyword Parameter	Meaning	Characteristics of Value
OPTION	Numerical value specifying type of CASE construct.	Integer value of 1 or 2.
LABEL	Label linking ENDCASE macro with corresponding CASE macro	Any valid ALC label.
LAB	Label linking ENDCASE macro with corresponding ENDELSE macro.	Any valid ALC label.

.

especially appropriate for structured programming since it produces no backward branches. Each CASE construct is tested sequentially as it is encountered within the code producing a linear flow of control.

Appendix D contains a simple program which illustrates the combined use of the CASE and nested DO macros. A flowchart of the program is also included to illustrate the control structure of the program.

The DO, LEAVE, CASE, and ENDCASE macros contain a number of assembly-time error messages. All pertain to the specifications of macro parameters. Some of these messages indicate illegal parameter values while others identify missing parameters or duplicate parameters. It is possible to detect these error conditions at assembly-time since macro parameter values must be specified before program assembly begins. Other error conditions, relating to particular ALC instructions which are included within a macro definition, will produce system-generated messages at assembly time.

The macros described in this chapter do not contain any execution-time error tests since any error condition occurring during execution will produce a system-generated error message. At execution time each macro has already been expanded into a set of ALC instructions, and these instructions with other program instructions are subject to all of the error-detecting capabilities of the particular system being used.

### CHAPTER BIBLIOGRAPHY

- 1. Chapin, Ned, <u>360/370</u> Programming in Assembly Language, New York, McGraw-Hill Book Co., <u>1968</u>.
- Dijkstra, E. W., "Structured Programming," <u>Software</u> <u>Engineering</u>, Report on a Conference Sponsored by the NATO Science Committee, Rome, Italy, October, 1969, pp. 84-88.
- 3. Hannula, Reino, <u>Computers and Programming</u>: <u>A System/</u> <u>360-370</u> <u>Assembler Language Approach</u>, Boston, <u>Ma.</u>, <u>Houghton Mifflin Co.</u>, 1974.
- Kent, William, "Assembler-Language Macroprogramming: A Tutorial Oriented Toward the IBM 360," <u>Computing</u> Surveys, (December, 1969), 183-196.
- 5. <u>IBM System/360</u> Operating System Assembler Language, IBM Form No. GC28-6514-8.
- 6. IBM System/360 Operating System Principles of Operation, IBM Form No. GA22-6821-8.

#### CHAPTER V

### SUMMARY, CRITIQUE, AND CONCLUSIONS

#### Summary

Within the last seven years a growing number of programmers have expounded various concepts and methodologies which they believe will improve software design and will increase the productivity, reliability, maintainability, and extendability of programs. These various concepts and techniques have been collectively, and in part, referred to as "structured programming."

The purpose of this study was to present a survey of the major concepts, contributors and techniques of structured programming, and to develop a set of IBM 360 assembly language macros which would facilitate the application of structured programming control structures to assembly language programming.

The first significant contribution to structured programming was made by E. W. Dijkstra when he suggested (3) that the GOTO statement may prove harmful to a program's structure if the construct is used indiscriminately by the programmer. Dijkstra suggested that avoiding the use of the GOTO structure may significantly improve the logic of most programs.

Since the time of Dijkstra's paper (3) a controversy has developed concerning the use of the GOTO construct. The question today is whether or not the GOTO statement should be eliminated from programming (7). "GOTO-less" programming is sometimes used synonymously with structured programming, although the GOTO issue is by no means the only concept involved in the development of a structured programming method. Other concepts being discussed include such issues as the proper methods for subroutine entry and exit (7), and the conditions permitting abnormal program termination (interrupts).

A second significant contribution to structured programming made by Dijkstra is his development of the 'THE' Multiprogramming System (2) which illustrates a technique for system software design. This technique includes the use of what Dijkstra termed "abstract resources" in system design. These resources divided the system into layers or levels of abstraction (both control and data structures) which then were applicable to the definition of various programmable modules.

Dijkstra's work has had a profound impact on the development of structured programming, and most of the literature on structured programming, at least in part, reflects his philosophies.

Another important contributor to structured programming is Harlan Mills of the IBM Corporation. Mills (5) and his associates at IBM have developed the Chief Programmer Team method for the production of large programs. This method incorporates both a managerial and organizational approach to program design.

The Chief Programmer Team defines in a very specific manner the role of a chief programmer, a backup programmer, and a programming librarian in a programming team. The method also employs a well-defined system of program documentation during program development.

Mills (5) describes his "top-down" approach to program design. This approach includes the use of modular programming, specific programming control structures, and strict rules governing the entry and exit points of program modules. Data structures in the top-down approach are important only to the extent that they affect module interfacing.

The concepts of the Chief Programmer Team and top-down programming have contributed a possible solution to the problem of defining a structured programming method for the development of large programs.

An article by Bohm and Jacopini (1) is the basis for structured programming control structure theory. They showed that the logic of any program can be defined with the use of only three types of control structures. These

structures include a sequential, a repetitive, and a selective construct. The use of only these three control structures eliminates not only the necessity of GOTO statements, but also any other control structures which may complicate program logic.

Other programming techniques which are associated with structured programming include: modular programming, open programming, compartmentalization, information hiding, extensive program documentation, the use of indentation and spacing in program code, and the use of meaningful labels and variable names in program coding.

At this time there is no generally accepted definition of structured programming. The literature suggests various approaches to improving programming techniques, and it remains to be seen just which concepts and techniques will be included in a formal definition of structured programming.

In order to apply some of the structured programming principles expounded in the literature, the author developed a set of IBM 360 assembly language macros, described in this thesis, to provide the assembly language programmer the necessary and sufficient control structures defined by Bohm and Jacopini (1).

The IBM 360 assembly language macros implemented include a DO macro for repetitive constructs and a CASE macro for selective control structures. A LEAVE macro was also developed to provide an escape mechanism for abnormal

termination of a repetitive control structure. No macro was developed for sequential program control since IBM 360 assembly language instructions are executed sequentially when there are no instructions to the contrary.

These macros may be used to approximate structured programming constructs in assembly language. It should be remembered, however, that assembly language programs are inherently more tedious to code and interpret than are higher languages; consequently, structured programming techniques may not be as easily applied to assembly languages as they are to higher languages.

The presence of these macros within a program does not imply that the program is well structured. Only when the programmer takes care to apply the principles of structured programming throughout his code and uses the macros in the prescribed fashion will the control structures of his program reflect the type of structuring which is characteristic of structured programming.

In addition, a completely well-structured program would presumably observe certain principles of structuring restricting the access of program modules to data. It is not clear that the data hierarchy concepts of Dijkstra (2) will survive the test of time; in any case, the macros developed in this thesis have not considered the problems of data structures.

### Critique

This investigation uncovered only one other attempt (4) to implement similar control structures using programmer-defined ALC macros. Kessler's (4) macros included a DO and CASE macro but no LEAVE statement.

The most significant difference in the author's DO macro and Kessler's is the specification of looping conditions through keyword parameters. Kessler's (4) DO macro is somewhat more flexible in that an UNTIL option is available, but it is also more restrictive since some conditional parameters must be specified in prescribed registers. The parameters Kessler (4) uses are fewer in number but substantially more complicated than those of this author.

Kessler's (4) CASE macro has a completely different implementation than the CASE macro of the author. Referring to Figure 26, an example of Kessler's CASE macro, if register  $R_x$  conatins a 5, 6, or 7, then Code A is executed. If register  $R_x$  contains a 3, then Code B is executed. If  $R_x$  does not contain a 3, 5, 6, or 7, then neither Code A or Code B will be executed. Kessler's (4) CASE macro is significantly simplier than that in this thesis, but because it lacks conditional execution parameters, the programmer will be required to include necessary conditional testing procedures in his program. For instance, using the example CASENTRY R<sub>X</sub> CASE 5,6,7 Code A CASE 3 Code B

#### ENDCASE

Fig. 26--An example of Kessler's CASE macro

of Figure 26, the programmer would test a condition within his program code. Depending upon the truth of the test, he would load a certain value into register  $R_x$  which would cause the appropriate CASE to be executed. Since the programmer is required to encode the testing procedure, Kessler's (4) CASE macro is not limited to the simple testing conditions of the CASE macro implemented in this thesis. It appears then that the CASE macro of Kessler is more flexible in terms of defining test conditions, but Kessler's macro will require more programmer-generated code than the CASE macro described in this thesis.

### Conclusions

Besides developing an adequate set of structured program control structures at the assembly language level, there appears to be a need to impose some new discipline of data structuring and access, especially at the assembly language level of programming. These are only weakly enforced by most input-output system calls. Multics (6) is the outstanding exception with regard to file security, but Multics per se does not enforce a modularity of user program structure.

It is hoped that after further investigation a set of principles will eventually be developed which will form the basis of a structured programming method. Not only would such a method improve the design and production of computer software and programs, but it would also initiate some semblance of uniformity in programming techniques and program structure. If all programs were written using the same basic set of programming principles, perhaps the interpretation of program logic would become standardized.

At the present time it appears that any structured programming method, if it is to be totally accepted, will have to be of a very general nature, encompassing only a few flexible principles, since the software and programs of today involve such a wide variety of programming problems and applications.

#### CHAPTER BIBLIOGRAPHY

- Bohm, C. and G. Jacopini, "Flow Diagrams, Turing Machines and Languages with Only Two Formation Rules," <u>Communications of the ACM</u>, IX (May, 1966), 366-371.
- Dijkstra, E. W., "The Structure of the 'THE' Multiprogramming System," <u>Communications of the ACM</u>, XI (May, 1968), 341-356.
- 3. \_\_\_\_\_, "GOTO Statement Considered Harmful," <u>Communications of the ACM</u>, XI (March, 1968), 147-148.
- Kessler, M. M., <u>\*CONCEPT\* Report 4</u>, <u>OS/360</u> Assembly Language Block Structured Programming Macros, IBM.
- 5. Mills, H. D., <u>Chief</u> <u>Programmer Teams</u>: <u>Principles</u> <u>and Procedures</u>, <u>Report No. FSC 71-5108</u>, <u>IBM Federal</u> <u>Systems Division</u>, <u>Gaithersburg</u>, <u>Maryland</u>.
- 6. Organick, Elliot I., The Multics System: An Examination of Its Structure, Cambridge, Ma., The MIT Press, 1972.
- 7. Special Interest Group on Programming Languages, <u>Special Issue on Control Structures in Programming</u> <u>Languages</u>, New York, New York, Association for Computing Machinery. Vol. VII (November, 1972).

## APPENDIX A

## THE DO AND ENDDO MACROS

# The Source Listing for the DO Macro

**\*** 

#### STMT SOURCE STATEMENT

2 3 4 5 6 7 8	*	-	THE FOLLOWING MACRO EXPANDS TO PRODUCE AN ITERATIVE OD STATEMENT AND/OR AN ITERATIVE DO WHILE CONDITION. A DD LODP WILL BE GENERATED WHEN THE PARAMETERS (SDD- LODP' IS NON-NULL, AND WHEN THE LODP PARAMETERS HAVE BEEN SPECIFIED. THE LODP PARAMETERS INCLUDE AN IN- CREMENT, A LOWER LODP LIMIT AND AN UPPER LOOP LIMIT. A DO WHILE CONDITION WILL BE GENERATED WHEN THE PA- RAMETER (SWHILE' IS NON-NULL, AND WHEN THE CONDITION PARAMETERS HAVE BEEN SPECIFIED. THE WHILE CONDITION PARAMETERS INCLUDE A LEFT DPERAND, A LOGICAL OPERA- TOR, AND A RIGHT OPERAND.
13			
14		MACRO	
15		DD	&LABEL=,&LOWNUM=,&BYNUM=,&HGHNUM=,&LOWREG=,&DOLGOP=,       X         &BYREG=,&HGHREG=,&LOWLOC=,&BYLOC=,&HGHLOC=,&WHILE=,       X         &WOP=,&WREGA=,&WREGB=,&WLOCA=,&WLITA=,&WLITA=,       X
16		LCLA	SLAB -
	εlab	SETA	a sy sudx
18	- س	AGO	• SK1P1
20			
21			INITIALLY THE 'DOLODP' PARAMETER IS TESTED TO DETER-
22			MINE THE PRESENCE OF A DO LOOP CONDITION. IF IT IS
23			NULLE A BRANCH TO THE *WHILE* LABEL WILL TEST FOR
24			THE PPESENCE OF A DO WHILE CONDITION. TF THE 'DO-
25 26			LOOP: PARAMETER IS NOT NULL, THEN THE CONTENTS OF
27			REGISTERS 8 THROUGH 11 ARE IMMEDIATELY SAVED; THESE REGISTERS ARE USED AS IMORK REGISTERS! BY THE MACRO
28			GENERATING THE DO LOOP CODE.
29	×		
30			
	SKIPL	AIF	("&DCLDOP" EQ '').WHILE
32 33		ST . ST	8, SAVER
33 34		ST	9.SAVE9 10,SAVE10
35		ST	11, SAVELL
36		AGO	• SKIP2
37	*		
38			• • •
39 40			IF A DD LOOP HAS BEEN SPECIFIED, THE FOLLOWING SEC-
40			TION OF CODE SELECTS THE LOWER LIMIT OF THE OD LOOP RANGE. THIS LIMIT MAY BE SPECIFIED BY A LITERAL
42			RANGE, THIS LIMIT MAY BE SPECIFIED BY A LITERAL (LOWNUM), BY THE CONTENTS OF A REGISTER (LOWPEG) OR
43	*		BY THE CONTENTS OF A LOCATION (LOWLDCH, A NUMBER OF
44			COMPARISONS ARE MADE TO VEPIEV THAT THE SPECIFIED
45			PARAMETER HAS BEEN CODED WITH AN ACCEPTABLE VALUE.
46 47			IF THE PARAMETER IS VALID, THE LOWER LIMIT IS LOADED INTO PEGISTER LO; OTHERWISE, AN ERROR MESSAGE IS

48	*		PRIME AND THE MACOO TEOMANATES
49			PRINTED AND THE MACRO TERMINATES.
50			
	SKIP2	ATE	( &LOWNUM + EQ ++ ).A
52		AIF	(*CHGHNUM* EQ **).Al
53		ATE	(*GBYNUM* EQ ++).A1
54		ATE	(* ELOWNUM* LIT * EHGHNUM + ). POS
55		ATE	("&LOWNUM" EQ "&HGHNUM") EPR16
56		AGG	• NEG
	• P.O.S	AIF	('&BYNUM'(1,1) EQ '-').ERR17
58		AG O	• 41
	n NEG	ATE	(*&BYNUM* GT *0*).ERR18
	• A 1	LA	10,0
61		4	10,=F*&LOWNUM*
62		AGO	• C
	• A	AIF	('&LDWREG' EQ '').8
64		AIF	(T'ELOWREG NE 'N'I.ERR24
65		ALF	('&LOWREG' GT '15'). ERR19
66		LR	10, &LOWREG
67		AGD	• C1
	• B	AIF	('&LOWLOC' ED '').ERR1
69 70		l LCO	10, alowloc
	•C	AGO	
	•C	AIF ATF	(*ELOWREG' NE 'ILERR9
73	6 C I	ALE ASO	('&LOWLOC' NE '').ERR9 .C2
74	*	A D ()	0 U Z
75			
76			THE FOLLOWING SECTION OF CODE SELECTS THE UPPER LIM-
77			IT OF THE DO LOOP RANGE. IT MAY BE SPECIFIED IN THE
78			SAME FASHION AS THE LOWER LIMIT; THE CORRESPONDING
79			PARAMETERS ARE HGHNUM, HGHREG AND HGHLDC. THE CODED
80	*		PARAMETER IS ALSO CHECKED FOR VALIDITY, AND THE UP-
81	*		PER LIMIT IS LOADED INTO REGISTER 11. AN APPROPRI-
82	<b></b>	1	ATE ERFOR MESSAGE WILL BE GENERATED FOR ILLEGAL PA-
83			RAMETER VALUES AND THE MACPO WILL TEPMINATE.
84			
85			
	•C2	AIF	('SHGHNUM' EQ ''),D
87		LA	11,0
88		A .	11, = F'&HGHNUM'
89 90	- D	AGO Alf	
91	•0	ALF	('&HGHREG' EQ '')/E
92		ATE	(T'&HGHREG NE "N').ERR25 ('&HGHREG' GT '15').ERR20
93		LR	11, SHGHREG
94		AGO	eF1
95	•E	AIF	('SHGHLOC' EQ '').ERR2
96		L	11. CHGHLOC
97		AGO	sF2
98	۰F	AIF	("&HGHREG" NE "").ERR10
99	•F1	AIF	(*EHGHLOC' NE **).ERRIO
100		AGO	•F2
101			· · · · ·
102			
103			THIS NEXT SECTION OF CODE SELECTS THE DO LOOP IN-
104			CREMENT FOR THE SPECIFIED RANGE. THE INCREMENT MAY
105 106			BE SPECIFTED BY A LITERAL (BYNUM), BY FEGISTER
105			(BYREG) OR BY LOCATION (BYLCC). THE INCREMENT IS
108			LOADED INTO REGISTER 9. ILLEGAL PARAMETER VALUES
109			WILL TERMINATE THE MACRO.
ĩĩó			
111		AIF	(168YNU4+ EQ:++1+G
112		AIF	('6BYNUM' EQ '0').ERR12
113		LA	9+0
114		Д	9,=F1&BYNUM+

115 116 .G 117 118 119 120	AG () AT F AT F AT F L R AG ()	•J {'&BYPEG' EQ '').H {T'&BYPEG NE 'N'}.ERR26 {'&BYREG' GT 'IS'}.ERR23 9,&BYREG •J1
121 .H 122	AIF L	(*&8YLOC* EQ ** ). ERR3 9: &8YLOC
123 124 • J 125 • J1 126 127 *	AGO AIF AIF AGO	oJ2 ('&BYPEG' NE ''}oERRII ('&BYLDC' NE ''}oERRII ₀J2
128 ** 129 * 130 ** 131 * 132 * 133 * 134 *		HAVING LOADED THE INCREMENT, THE LOWER RANGE LIMIT AND THE UPPER RANGE LIMIT INTO REGISTERS 9,10 AND 11 RESPECTIVELY, THESE REGISTERS ARE STORED IN A STACK. REGISTERS B THROUGH 11 ARE THEN LOADED WITH THEIR ORIGINAL CONTENTS.
135 * 136 .J2 137 138 139 140 141 142 143	L ST LA ST LA ST	8; POINTER 9; STACK(8) 8;4(8) 10; STACK(8) 8;4(8) 11; STACK(8) 8;4(6) 8; POINTER
144 145 146 147 148 149 * 150 *	L L L Agg	8;SAVE8 9;SAVE9 10;SAVE10 11;SAVE11 • SKIP3
151 * 152 * 153 * 154 * 155 * 156 * 157 *		THE FOLLOWING STATEMENTS REPRESENT THE BEGINNING OF THE ACTUAL GENERATED DO LODP& WHEN THE LODP HAS BEEM EXECUTED, A BRANCH BACK TO 'QUABEL' WILL INIT- IATE RETESTING OF THE LOOP CONDITIONS AND POSSIBLE REPEATED EXECUTION OF THE LOOP. TO TEST THE LOOP THE WORKING REGISTERS (8 THROUGH 11) ARE STORED, AND THE TEST PARAMETERS ARE LOADED.
159 * 160 SKIP3 161 ELABEL 162 163 164 165 166 167 168 169	ANDP ST ST ST ST L S L	8,SAVE8 9,SAVE9 10,SAVE9 11,SAVE10 11,SAVE11 8,PDINTER 8,=F:4: 11,STACK(8) 8,=F:4:
170 171 172 173 - 174 *	L S L ST AGO	10, STACK(8) 8,==+4 9, STACK(8) 8, POTNTER • SKIP4
175 * 176 * 177 * 178 * 179 * 180 *		THE CONTENTS OF REGISTER 9 APE COMPARED AGAINST ZERO TO DETERMINE WHETHER INCREMENTING IS IN THE POSITIVE OR NEGATIVE DIRECTION: THIS WILL DIRECTLY AFFECT THE COMPARISION OF REGISTERS TO AND IL WHICH ARE USED TO DETERMINE WHEN THE LOOP IS SATISFIED. REGISTER TO

IS COMPARED TO REGISTER 11 AND IF IT IS GREATER THAN 181 \* 182 10 REGISTER 11 FOR A POSITIVE INCREMENT OR LESS THAN REGISTER 11 FOR A NEGATIVE INCREMENT, THE PROGRAM 183 BRANCHES OUT OF THE DO LOOP. IF THE LOOP IS NOT 184 185 10 SATISFIED, CONTROL PASSES TO THE NEXT SECTION OF 186 \* CODE. 187 \* 188 \* • SKTP4 189 ¢ 9,ZER0 190 BL X&LAB 191 CR. 10,11 192 ₿Н YELAB 193 в ZELAB C٩ 194 XELAB 10.11 195 8L YELAB 196 AGO SKIP5 197 \* 198 \* 199 \* WHEN THE DO LOOP IS NOT SATISFIED, THE FOLLOWING SECTION OF CODE ADDS THE INCREMENT TO REGISTER 10 200 \* 201 \* (WHERE THE LOWER RANGE LIMIT WAS INITIALLY LOADED) 202 \* AND STORES THE INCREMENT AND RANGE VALUES BACK INTO 203 xic THE STACK. 204 \* 205 \* 206 . SKTP5 ANOP Ą٩ 10,9 207 ZELAB 208 8, POINTER Ł 209 ST 9, STACK(8) 210 8,4(8) LΑ 211 ST 10, STACK(8) 212 LΑ 8,4(8) 213 ST 11, STACK(8) 214 Ľ٨ 8,4(8) 215 ST 8, POINTER 216 AGO ₀SKIP6 217 \* 218 \* \* 219 THE ORIGINAL CONTENTS OF THE WORKING REGISTERS ARE 220 \* RESTOPED AND THE PROGRAM BRANCHES TO THE BEGINNING 221 \* OF THE DO WHILE CODE. 222 \* 223 \* 224 .SKIP6 ٤ 8,SAVE8 225 L 9, SAVE9 226 Ľ 10, SAVE10 227 11,SAVE11 ٤ 228 В WELAB 229 AGO SKTP7 230 \* 231 \* 232 \* WHEN THE DO LOOP IS SATISFIED, A BRANCH FROM EITHER 233 \* STATEMENT 204 OR 207 TO THE FOLLOWING CODE WILL THEN 234 \* BE MADE. THE WORKING REGISTERS WILL BE LOADED WITH 235 \* THEIR ORIGINAL VALUES (THE DO LOOP PARAMETERS WILL 236 \* NOT BE STACKED) AND THE BPANCH TO "ASLABEL" WILL 237 \* TERMINATE THE LOOP. 238 \* 239 \* 240 . SKTP7 ANOP 241 YELAB L 8,SAVE8 9, SAVE9 242 Ł 243 L 10, SAVE10 244

L

8

EQU

245

246 WELAB

11, SAVE11

AGLABEL

247	AGO	• SK198
248 *	400	· • 201.40
249 *		
250 *		, THE FOLLOWING CODE TESTS FOR THE PRESENCE OF A DO
251 * 252 *		WHILE CONDITION. IF A OD LOOP WAS SPECTFIED, BUT A
253 *		DO WHILE CONDITION WAS NOT, A BRANCH TO THE END OF THE MACRO OCCURS. IF A DO WHILE CONDITION HAS BEEN
254 *		SPECIFIED, THEN THE MACPO PROCEEDS TO GENERATE CODE
255 *		TO IMPLEMENT IT. IF NEITHER A DO LOOP MOR DO WHILE
256 *		CONDITION HAS BEEN SPECIFIED, AN ERROR MESSAGE IS
257 * · 258 *		PRINTED AND THE MACRO TERMINATES.
259 *		
260 • SKIP8	ATE	('EWHILE' EQ ''), END
261	AGO	• •
262 •WHILE 263 &LABEL	AT E	(*&WHILE* EQ **).ER\$4
265 GLADEL 264	EQU ASD	* • R
265 *	140.02	
266 *		
267 <b>*</b> 268 *		TO IMPLEMENT THE DO WHILE CONDITION REGISTERS 10 AND
269 *		11 APE USED AS 'WORK REGISTERS': THEPEFORE, THEIR Contents are initially saved.
270 *		CONTRACTOR AND
271 *		
272 •R 273	ST ST	10, SAVE10
274	AGO	11,SAVE11 • SKIP9
275 *		
276 *		
277 * 278 *		THE FOLLOWING INSTRUCTIONS LOAD THE LEFT OPERAND OF
279 *		THE OD WHILE CONDITION INTO REGISTER 10. THIS DPER- AND MAY BE SPECIFTED IN A REGISTER (WREGA), BY A
280 *		LOCATION (WLDCA) OR BY A LITERAL (WLITA). IF THE
281 *		SPECIFIED OPERAND IS NOT A VALID VALUE (SELE-
282 * 283 *		DEFINING NUMERIC) WHEN A LITERAL IS CODED, THEN AN
284 *		ERPOP MESSAGE WILL BE PRINTED AND THE MACRO WILL TERMINATE.
285 *		
286 * 287 • Skip9		
288	AIF AIF	(18WREGA! EQ !!!).K (T!&WPEGA NE !N!).ERR27
289	AIF	(*&WREGA' GT *15') ERR21
290	LR	10, GWREGA
291 292 •K	AGO AT F	
293	L	(*&WLOCA* EQ **).L 10,&WLOCA
294	Ă30	• M1
295 «L	AIF	("GWLTTA" EQ "").ERR5
296 297	41 F 14	('GWLITB' NE '').ERR15 10,0
298	A	LJ+D 10,=F*&WLITA*
299	AGO	• 12
300 .M	AIF	('EWLOCA' NE ''). ERR13
301 • M1 302	AIF AGO	(*&WLITA* NE **).ERR13
303 *	RUU	
304 *		
305 * 306 *		THE FOLLOWING STATEMENTS LOAD THE PIGHT OPERAND OF
307 *		THE DO WHILE CONDITION INTO REGISTER 11 AND COMPARE IT TO REGISTER 10. THIS OPERAND MAY ALSO BE SPECT-
308 *		FIED IN A REGISTER (WREGB), BY A LOCATION (WLCCB)
309 *		- THE BY A LITERAL (WLITS). IF THE SPECTETED DOGDAND
310 * 311 *		IS AN INVALID VALUE WHEN A LITERAL IS CODED. AND A
312 *		ERROP MESSAGE WILL BE PRINTED AND THE MACRO EXPAN- SION WILL BE TERMINATED.

	313 *		
	314 *		
	315 .M2	AIF	("&WREGB" EQ "").N
	316	ATE	(T'EWPEGB NE 'N').ERR28
	317	ATE	('&WREGR' GT '15'), ERR22
	318	[ ?	11, EWPEGB
	319	AGO	۰P
	320 •N	AIF	(*&WLOC8* EQ **).0
	321	Ľ	11, EWLOCB
	322	ĂGO	. 21
	323 .0	AIF	("EWLITB" EQ "").ERR6
	324	LA	11,0
	325	Ā	11, = F* & WLITB*
	326	AGO	• P2
	327 "P	AIF	(" & WLOCB" NE " " ). ERR14
	328 P1	AIF	(*&WLITB* NE **), ERR14
	329 . P2	ĊŔ	10,11
	330	AGO	• SKIP10
	331 *	A0 J	P D// 14 TO
	332 *		· ·
	333 *		WHEN BOTH OPERANDS OF THE DO WHILE CONDITION HAVE
	334 *		REEN LOADED THIS YEAR CODE COOKENT OF FOR THE
	335 ×		BEEN LOADED, THIS NEXT CODE SEGMENT SELECTS THE
	336 *		LOGICAL OPERATOR USED IN THE COMPARSION. IF AN IL-
	337 *		LEGAL OPERATOR HAS BEEN SPECIFIED OR TE ND OPERATOR
	338 *		HAS BEEN SPECIFIED, THE MACRO WILL TERMINATE WITH AN
	339 *		ERROR MESSAGE: DTHERWISE, THE APPROPRIATE COMPARISON
	340 *		WILL BE MADE, AND IF THE OD WHILE CONDITION IS SAT-
			ISFIED, A BRANCH TO TERMINATE THE LOOP WILL OCCUR.
	341 * 342 *		
	343 •SKIP10		
		47 F	(18W0P1 EQ 11).EPP7
	344 345	AIF	('EWNP' EQ 'EQ'ISEDUAL
		ATE	('EWOP' EQ 'LT'I.LESS
ĺ	346	ALF	(ISWOPI EQ ILEI).LESSEQ
	347	AIF	('GWOP' EQ 'GE').GRTEQ
	348	41 F	(*EWAP* EQ *GT* ). GRT
	349	41 F.	('GWOP' EQ 'NE').NOTEQ
	350	AGO	• E0 0.8
	351 EQUAL	BNE	TELAB
	352	AGO	• 3
	353 LESS	BNL	TALAB
	354	AGO	• 3
	355 .LESSEQ	84	TELAB
	356 357 COTEO	AGO	
	357 GRTEQ	BL	TALAB
	358 350 CPT	AGO	
	359 "GRT 360	88H 703	T & L & B
		46 <b>1</b>	
	361 •NOTEQ 362	BE	TELAB
	363 *	ASC	
	364 * 365 *		
			ONCE THE DO WHILE CONDITION HAS BEEN TESTED, THE
	366 * 367 *		FOLLOWING CODE SEGNENT RESTORES THE WORKING PEGIS-
			TERS, 10 & 11, AND IF THE DO WHILE CONDITION IS NOT
	368 *		SATISFIED, BRANCHES TO THE END OF THE MACRO WHERE
	369 *		THE LOOP WILL BE EXECUTED AGAIN. TE THE CONDITION
	370 *		HAS BEEN SATISFIED, A BRANCH TO THE 'ENDD3' MACRO
	371 *		WILL OCCUR AND THE LOOP WILL BE TERMINATED.
	372 *		
	373 *		1.5. 641/61.5
	374 . 9	L	10, SAVE10
	375	L p	11,SAVE11
	376	B	VELAB
	377 TELAB 378	L L	1), SAVE10
	2 f Q	L	11,SAVE11

379		8	A&LABEL
	VELAB	EQU	*
381		AGO	• END
382			· ·
383 384			
385			THIS FINAL SECTION OF MACRO CODE REPRESENTS THE ER- ROR MESSAGES GENERATED OURING MACRO EXPANSION, ER-
386			RORS INDICATE THAT THE MACYO CAN NOT BE EXPANSION. EF-
387			PRODUCE EXECUTEABLE CODE: CONSEQUENTLY, WHEN ONE
388	*		IS DETECTED, A MESSAGE IS PPINTED DUT AND A BRANCH
389			TO THE 'ENDDD' MACRO DECURS. THE WORKING REGISTERS
390			WILL BE RESTORED AND THE 'DD' MACRD WILL NOT BE FUR-
391			THER EXPANDED.
392 3 <b>93</b>			
	ERR1	MNOTE	*, LOWER DO LOOP RANGE NOT SPECIFIED*
395		B	CELABEL
396		ĂG O	e END
397	e ERR2	MNDTE	*, 'UPPER DO LOJP RANGE NOT SPECIFIED'
398		8	CELABEL
399		AGO	• END
	•ERR3	MNOTE	*, 'DD LOOP INCREMENT NOT SPECIFIED'
401 402	•	8	C&LABEL • END
	ERP4	MNOTE	*, NO DO OR DO WHILE CONDITION SET!
404	/	В	A&LABEL
405		AGO	● END
	■ERR5	MNOTE	*, 'NO LEFT DPERAND SPECIFIED ON DD WHILE CONDITION!
407		8	BELABEL
408	• ERR.6	AGN MNOTE	• END
410	¢ CNP.U	8990E	*, *NO RIGHT OPERAND SPECIFIED ON DO WHILE CONDITION! B&LABEL
411		AGO	• END
412	ERR7	MNOTE	*, 'D' WHILE OPERATOR NOT SPECIFIED'
413		В	BELABEL
414		AGD	. END
415	◆ ERR8	MNDTE	*, TAVALID DO WHILE OPERATOR SPECIFIED
417		8 460	BGLABEL • ENO
	.ERR9	MNOTE	*, MOPE THAN ONE LOWER DO LOOP RANGE SPECIFIED!
419		В	CELABEL
420		AGO	• END
	•ERR10	MNOTE	*, MORE THAN ONE UPPER OD LOOP RANGE SPECIFIED!
422 423		8	Calabel
-	.ERR11	AGO MNOTE	.END *,**72°E THAN ONE DD LODP INCREMENT SPECIFIED!
425	• Las in L	в	CALABEL
426		AGN	♣ END
	.ERR12	MNDYE	*, ZERD IS INVALID DO LOOP INCREMENT!
428		<u>р</u>	CALABEL
429	.ERRI3	AGD MNDTÈ	
431	0 6 7 6 1 2	B	*, MORE THAN ONE LEFT DO WHILE OPERAND SPECIFIED! BALABEL
432		ĂGO	• END.
433	eERR14	MNDTE	*, MORE THAN ONE RIGHT OD WHILE OPERAND SPECIFIED!
434		8	BALABEL
435	6001 C	AGO	• END
435	ERR15	MNOTE B	*, DO WHILE CONDITION CONTAINS TWO LITERAL OPERANDS. BELABEL
438		AGO	• END
439	ERRI6	MNOTE	*, OO LOOP HAS SPECIFIED RANGE OF ZERO*
440		AGO -	6 A L
	•ERR17	MNOTE	* * * VEGATIVE INCREMENT INVALID FOR SPECIFIED DO LOOP*
442 443		B AGM	C &L ABEL • END
	•ERR18	AVOTE	*,*POSITIVE INCREMENT INVALID FOR SPECIFIED DD LODP*
			<ul> <li>Section contracts taxacto LOK SECTATED D3 [2004</li> </ul>

445	в	CALABEL
446	AGO	• END
447 .ERR19	MNOTE	*, INVALID NUMBER FOR 'LOWREG' SPECIFIED IN DO!
448	чурте	*'ITUDB4
449	8	CELABEL
450	ĂGO	• END
451 .ERR20	MNOTE	*, INVALID NUMBER FOR **HGHREG** SPECIFIED IN DO!
452	MNOTE	*, LODA:
453	B	CELABEL
454	AGO	• END
455 .ERR21	MNOTE	*, INVALID NUMBER FOR ''WREGA'' SPECIFIED IN DO
456	MNOTE	** MHITE.
457	8 .	BELABEL
458	AGD .	• END
459 .ERR22	NOTE	*, "INVALID NUMBER FOR ""WREGB"" SPECIFIED IN DO
460	MNOTE	*.*WHILE*
461	8	BGLABEL
462	ĂSŋ	• END
463 .ERR23	MNOTE	*, TAVALID NUMBER FOR "BYREG! SPECIFIED IN DO
464	MNOTE	* * LOUD*
465	6	CALABEL
466	ĂGO	• END
467 . ERR24	MNDTE	*, "'LOWREG'' SPECIFIED IN DO LOOP IS NOT A SELF-
468	MNOTE	*, 'DEFINING NUMERIC'
469	В	CCLABEL
470	ÁG 0	END
471 .ERR25	MNOTE	*, *** HGHREG** SPECIFIED IN DO LOOP IS NOT A SELF-*
472	MNDTE	*. DEFINING NUMERIC
473	В	CGLABEL
474	AGO	• END
475 • ERR26	MNOTE	*, ***BYSEG** SPECIFIED IN DO LOOP IS NOT A SELF-*
476	MNOTE	*, 'DEFINING NUMERIC'
477	В	CELABEL
478	AG 0	• ÉND
479 eERR27	MNOTE	*. " WREGA SPECIFIED IN DO WHILE IS NOT A SELF-
480	MNOTE	*, "DEFINING NUMERIC"
481	6	BELABEL
482	AG D	• END
483 • ERR28	MNOTE	*,***WREGB** SPECIFIED IN DO WHILE IS NOT A SELF-*
484	MNOTE	*, DEFINING NUMERIC:
485	в	BELABEL
486	AGO	• END
487 • END	MEND	

# The Source Listing for the ENDDO Macro

SOURCE STATEMENT

STMT

489 *THE ENDDO MACRO PROVIDES THE RETUPN BPANCH FOR AN490 *TTEPATIVE DO LOOP OR DO WHILE CONDITION. IT IS RE-491 *QUIRED WHENEVER A DO MACRO IS USED. AND THE NAME AS-492 *SIGNED TO THE LABEL! PARAMETEP SHOULD BE THE SAME493 *AS THAT GIVEN TO THE LABEL! PARAMETER IN ITS COR-494 *RESPONDING DO MACRO. IF A SPECIFIED OD LOOP IS NOT495 *EXECUTED BECAUSE OF AN ERPOR CONDITION. A BRANCH TO496 *THE ENDDO MACRO WILL OCCUP. THE WORKING REGISTERS497 *WILL BE RESTORED AND THE MACPO TEPMINATED. WHEN A498 *DO MACRO HAS BEEN EXECUTED PPOPERLY, IT WILL RESTORE499 *THE WORKING REGISTERS AND. UPON COMPLETION. BRANCH500 *TO THE END OF THE ENDDO MACRO WHERE THE PROGRAM WILL501 *CONTINUE WITH SEQUENTIAL CODE EXECUTION.503 *		
490 *ITEPATIVE DO LOOP OR DO WHILE CONDITION. IT IS RE-491 *QUIRED WHENEVER A DJ MACRO IS USED. AND THE NAME AS-492 *SIGNED TO THE 'LABEL' PARAMETEP SHOULD BE THE SAME493 *AS THAT GIVEN TO THE 'LABEL' PARAMETER IN ITS COR-494 *RESPONDING DO MACRO. IF A SPECIFIED DO LOOP IS NOT495 *EXECUTED BECAUSE OF AN ERPOR CONDITION. A BRANCH TO496 *THE ENDDO MACRO. WILL OCCUR. THE WORKING REGISTERS497 *WILL BE RESTORED AND THE MACPO TEPMINATED. WHEN A498 *DO MACRO HAS BEEN EXECUTED PROPERLY, IT WILL RESTORE499 *THE WORKING REGISTERS AND. UPON COMPLETION. BRANCH500 *TO THE END OF THE ENDDO MACRO WHERE THE PROGRAM WILL501 *CONTINUE WITH SEQUENTIAL CODE EXECUTION.	489 <b>*</b>	THE ENDDO MACRO PROVIDES THE RETURN BRANCH COD AN
491 #QUIPED WHENEVER A DJ MACRO IS USED, AND THE NAME AS-492 *SIGNED TO THE 'LABEL' PARAMETEP SHOULD BE THE SAME493 *AS THAT GIVEN TO THE 'LABEL' PARAMETER IN ITS COR-494 *RESPONDING DD MACRO. IF A SPECIFIED DO LOOP IS NOT495 *EXECUTED BECAUSE OF AN ERPOR CONDITION, A BRANCH TO496 *THE ENDDO MACRO. WILL OCCUR. THE WORKING REGISTERS497 *WILL BE RESTORED AND THE MACPO TEPMINATED. WHEN A498 *DO MACRO HAS BEEN EXECUTED PROPERLY, IT WILL RESTORE499 *THE WORKING REGISTERS AND, UPON COMPLETION, BRANCH500 *TO THE END OF THE ENDDO MACRO WHERE THE PROGRAM WILL501 *CONTINUE WITH SEQUENTIAL CODE EXECUTION.	490 *	TTERATIVE DO LOOP OR DO WHILE CONDITION IT IS DE
492 *SIGNED TO THE 'LABEL' PARAMETEP SHOULD BE THE SAME493 *AS THAT GIVEN TO THE 'LABEL' PARAMETER IN ITS COR-494 *RESPONDING DO MACRO. IF A SPECIFIED DO LOOP IS NOT495 *EXECUTED BECAUSE OF AN ERPOR CONDITION, A BRANCH TO496 *THE ENDDO MACRO WILL OCCUR. THE WORKING REGISTERS497 *WILL BE RESTORED AND THE MACPO TEPMINATED. WHEN A498 *DO MACRO HAS BEEN EXECUTED PROPERLY, IT WILL RESTORE499 *THE WORKING REGISTERS AND, UPON COMPLETION, BRANCH500 *TO THE END OF THE ENDDO MACRO WHERE THE PROGRAM WILL501 *CONTINUE WITH SEQUENTIAL CODE EXECUTION.	491 ×	QUIPED WHENEVER & DI MACRO IS USED, AND THE NAME AC
493 #AS THAT GIVEN TO THE "LABEL! PARAMETER IN ITS COR-494 #RESPONDING DO MACRO. IF A SPECIFIED DO LOOP IS NOT495 #EXECUTED BECAUSE OF AN ERPOR CONDITION, A BRANCH TO496 #THE ENDDO MACRO WILL OCCUR. THE WORKING REGISTERS497 #WILL BE RESTORED AND THE MACPO TEPMINATED. WHEN A498 #DO MACRO HAS BEEN EXECUTED PPOPERLY, IT WILL RESTORE499 #THE WORKING REGISTERS AND, UPON COMPLETION, BRANCH500 #TO THE END OF THE ENDDO MACRO WHEPE THE PROGRAM WILL501 #CONTINUE WITH SEQUENTIAL CODE EXECUTION.	492 *	SIGNED TO THE FLABEL PARAMETER SHOULD BE THE CAME
49% #RESPONDING DD MACRO. IF A SPECIFIED DD LDDP IS NOT495 *EXECUTED BECAUSE OF AN ERPOR CONDITION, A BRANCH TO496 *THE ENDDD MACRO WILL DCCUR. THE WORKING REGISTERS497 *WILL BE RESTORED AND THE MACPO TEPMINATED. WHEN A498 *DD MACRO HAS BEEN EXECUTED PPOPERLY, IT WILL RESTORE499 *THE WORKING REGISTERS AND, UPON COMPLETION, BRANCH500 *TD THE END OF THE ENDDO MACRO WHERE THE PROGRAM WILL501 *CONTINUE WITH SEQUENTIAL CODE EXECUTION.	493 <b>*</b>	AS THAT GIVEN TO THE MARELE PARAMETED IN TTO COD
495 *EXECUTED BECAUSE OF AN ERPOR CONDITION, A BRANCH TO496 *THE ENDDD MACPD WILL DCCUR. THE WORKING REGISTERS497 *WILL BE RESTORED AND THE MACPD TERMINATED. WHEN A498 *DD MACRD HAS BEEN EXECUTED PROPERLY, IT WILL RESTORE499 *THE WORKING REGISTERS AND, UPDN COMPLETION, BRANCH500 *TD THE END OF THE ENDDD MACRD WHERE THE PROGRAM WILL501 *CONTINUE WITH SEQUENTIAL CODE EXECUTION.	494 #	RESPONDING DO MACRO. TE A SPECIFIED DO LOOP IS NOT
496 #THE ENDDO MACRO WILL OCCUR. THE WORKING REGISTERS497 *WILL BE RESTORED AND THE MACRO TERMINATED. WHEN A498 *DO MACRO HAS BEEN EXECUTED PROPERLY, IT WILL RESTORE499 *THE WORKING REGISTERS AND, UPON COMPLETION, BRANCH500 *TO THE END OF THE ENDDO MACRO WHERE THE PROGRAM WILL501 *CONTINUE WITH SEQUENTIAL CODE EXECUTION.	495 x	EXECUTED BECAUSE OF AN ERROR CONDITION. A REANCH TO
497 *WILL BE RESTORED AND THE MACPO TEPMINATED.WHEN A498 *DD MACRD HAS BEEN EXECUTED PROPERLY, IT WILL RESTORE499 *THE WORKING REGISTERS AND, UPON COMPLETION, BRANCH500 *TD THE END OF THE ENDDO MACRO WHERE THE PROGRAM WILL501 *CONTINUE WITH SEQUENTIAL CODE EXECUTION.	496 *	THE ENDDO MACRO WILL OCCUR. THE WORKING PECTERS
498 *DD MACRO HAS BEEN EXECUTED PROPERLY, IT WILL RESTORE499 *THE WORKING REGISTERS AND, UPON COMPLETION, BRANCH500 *TO THE END OF THE ENDDO MACRO WHERE THE PROGRAM WILL501 *CONTINUE WITH SEQUENTIAL CODE EXECUTION.502 *	497 *	WILL BE RESTORED AND THE MACRO TERMINATED. WHEN A
499 *       THE WORKING REGISTERS AND, UPON COMPLETION, BRANCH         500 *       TO THE END OF THE ENDDO MACRO WHERE THE PROGRAM WILL         501 *       CONTINUE WITH SEQUENTIAL CODE EXECUTION.         502 *	498 ×	DO MACRO HAS BEEN EXECUTED PPOPERTY. IT WILL DESTORE
500 * TO THE END OF THE ENDDO MACRO WHERE THE PROGRAM WILL 501 * CONTINUE WITH SEQUENTIAL CODE EXECUTION. 502 *	499 *	THE WORKING REGISTERS AND. HEAN COMPLETION DO ANCH
501 * CONTINUE WITH SEQUENTIAL CODE EXECUTION.	500 *	TO THE END OF THE ENDOO MACRO WHERE THE DROCDAN WILL
502 *	501 *	CONTINUE WITH SEQUENTIAL CODE EXECUTION.
	502 *	
	503 *	

11-1

504		MACRO	
505		ENDDO	&LABEL=
506		B	<b>&amp;LABEL</b>
507	CELABEL	L	8,SAVE8
508		L	9,SAVE9
509	BELABEL	L	10,SAVE10
510		L	11, SAVE11
511	AGLABEL	EQU	*
512		MEND	•

## Sample Program A Illustrating the DO and ENDDO Macros

	STMT	SOURCE	STATEMENT	·
	962		PRINT	NOCEN
		COTCOSE		NOGEN
		FRISBEE		12, SAVEAREA
		START	0P EN	(DUMPAREA, DUTPUT)
	985		LA .	2,0
	986		LA	3,0
	987		L A	4,0
	988		LA	5,0
	989	*	LA	6,0
	990	-	LA	7,0
	991		LA	8,0
	992		LA	
	993		LA	9,0
			-	10,0
	994		LA	11,0
	995		LA.	14.0
	996		LA	15,0
	997			
	998	÷		
	999	• .	SNAP .	ID=1,DCB=DUMPAREA,PDATA=REGS
1	011	<b>±</b>		
- 1	012	*		
1	.913		00	WHILE=R, WREGA=4, WOP=LT, WLITE=10, LABEL=ONE
	029			
	030		•	A 4+≃F <sup>ℓ</sup> 1 <sup>ℓ</sup>
	031			A 5+=F*1*
	032			A 6, = F 1 1
	033			
	034			· · · · · · · · · · · · · · · · · · ·
				$A = B_{T} = F^{T} L^{T}$
	035			$A \qquad 9_{1} = F^{*} 1^{*}$
-	036			4 LO,=F*1*
	037		ENDDO	LABEL=ONE
	044			· ·
	045	*		
1	046		SNAP '	ID=2,DCB=DUMPAREA,PDATA=REGS
1	058		LA	3,0
1	059		LA	4,0
1	060		LA	5.0
1	061		LA	6,0
	062		LA	7,0
	063		LA	8,0
_	064		LA	9,0
	065		LA	
	066	*	ĻА	10.0
				×
	067	*	00	
	068		. 00	LOWNUM=1, HGHNUM=30, BYNUM=4, DOLOOP=S, LABEL=TWO
	130			A 3,≖F*2*
	131			$A \qquad 4_{s} = \mathbb{P}^{s} 2^{s}$
	132		٢	A 5,====2+
1	133			A 6,=F*2*

1134		4 - 7,=F*2*	
1135		A 8,=F*2*	
1136		A 9,=F!2!	
1137		A 10+=F*2*	
1138	ENDDO	LABEL=THO	
1145 *			
1146 *			
1147	SHAP		
1159	- LA -	ID=3,DCB=DUMPAREA,PDATA=REGS 3,0	
1160			
	LA .	4,0	
1161 ·	LA	5.0	
-1162	LA	6+0	
1163	LA I	7.0	
1164	LA	8,0	
1165	L4	9,0	
1166	LA	10,0	
1167 *	1		
1168 *			
1169	00	LOWNUM=3,BYLOC=Y,HGHREG=4,DOLOOP=7,WHILE=NN,	х
		LABEL=THRFE, WOP=GT, WLITA=7, WREGB=15	
1243 -		A	
1244		A 4,=F'3'	
1245		A 5,=F131	
1246		A 6,=F'3*	
1247		A 7,=F'3'	
1248		A 8,==F*3*	
1249	· · ·	A 9,=F-3	
1250		A 10,==F*3*	
1251	ENDER		
1258 *		EADEL - FONDE	
1259 *		•	
1260	SNAP		
1272 *	3446	ID=4,DCB=DUMPAREA,PDATA=REGS	
1273 *			
1274	EXIT		
1279		DUNDADE1	
** 1 7	CLOSE	DUMPAREA	
1005 64954054	D.C	101/01	
1285 SAVEAPEA		184(0)	
1286 1287 ZEPO	DS DC	, OF	
	DC	15.0.	
1288 SAVE8	DS	1F	
1289 SAVE9	DS DS	1F	
1290 SAVE10	OS	1F	
1291 SAVE11	DS		
1292 POINTER	DC	16401	
1293 STACK	DS	100F	
1294 Y	DC .	1F*1*	
1295 DUMPAREA	DC B	DONAME=TEAM, DSDRG=PS, RECEM=VBA,	x
		MACRF=W+BLKSIZE=882,LRECL=125	•
1346	END		
1347		=F*10*	
1348		= F * ] •	
1349		= F * 3.0*	
1350		=F*4*	
1351		=F*2*	
1352		= F 1 3 1	
1353		= f * 7 *	

REGS AT ENTRY TO SNAP

ID = 001

				4				
				• •			•	+
REGS 0-7	,	00000030	90018398	000000	00	0.00	0000	000
		00000000	00000000	000000	00	000	0000	000

REGS 8-15	00000000	00000000	00000000	00000000	
	50J18820	00018600	00000000	00000000	
SND DE SNAR					

-			••			
REGS	ΑT	ENTRY	Τņ	SNAP		

REGS 0-7	00000240	80018904	00000000	A0000000	
	A6006000	00000004	A0000000	0000000A	
REGS 8-15	0000004	A000000A	A0000000	0000000	`
	50018820	00018000	0000000	00000000	
END OF SNAP				· · · ·	

,		•
		,

REGS AT ENTRY TO SNAP

ID = 003

00000000

ID = 002

				~ ` `
REGS 0-7	000002A0	A0018458	00000000	00000010
	01006000	00000010	00000010	00000010
REGS 8-15	00000010	00000010	00000010	0000000
	50018820	00018000	00000000	00000000
END OF SNAP				

REGS AT ENTRY TO SNAP . ID = 004REGS 0-7 CAS0000 A0018804 00000000 0000000 00000000 00000000 0000000 0000000 REGS 8-15 00000000 00000000 00000000 00000000

00018000

00000000

50018820

END OF SNAP

# Program A with Expanded Macros

STMT SOURCE STATEMENT

962 FRISBEE	ENTER	12, SAVEAREA
963+FPISBEE	ÐS	0H
964+	ENTRY	FRISBEE DECLARE NAME ENTRY
965+	USING	*,12 DECLARE BASE ADDRESSIBILITY.
966+	BALS	15,0 (INITIAL ADDRESSIBILITY).
967+	В	12(,15) BRANCH AROUND ID FIELD
968+	DC	AL1(7), CL7'FRISBEE' ID LENGTH AND ID
969+	BCTR	15,0 (RESET INTITAL ADDRESSIBILITY
970+	BCTR	15.0 ABSOLUTE ENTRY POINTI
971+	STM	14,12,12(13) SAVE REGISTERS
972+	LR	12,15 SETUP BASE REGISTER.
973+	ST	13. SAVEAREA+4 CHAIN BACK
974+	LA	0, SAVEAPEA CHAIN FORWARD
975+	ST	0.8(0.13)
976+	ርሚ	13,0 SET UP SAVE AREA POINTER
977+	USING	SAVEAREA, 13 AND ADDRESSABILITY
978 START	OPEN	
979+	CNOP	
980+START	BAL	1,*+8 LOAD PEGI W/LIST ADDR.
981+	00	ALL(143) OPTION BYTE
982+	00	AL3(DUMPAREA) DCB ADDRESS
983+	SVC	19 ISSUE OPEN SVC
984	LA	2,0
985	LA	3,0
966	LA	4,0
-987	LA	5,0
988	LA	6,0
989	LA	7,0
990	LA	8,0
991	LA	9,0
992	LA	10,0
993	LA	11,0
994	LA	14.0
995 996 *	LA	15,0
997 #		
998	SNAP	
999+	CNOP	ID=1,DCB=DUMPAREA,PDATA=REGS 0,4
1000+	BAL	•
1001+	DC	1. TH80003 BRANCH AROUND PARAM LIST ALI(1) ID NUMBER
1002+	DC	ALI(0)
1003+	DC	ALI(130) DETION FLAGS
1004+	DC	AL1(32) OPTION FLAGS
1005+	DC	A(DUMPAREA) DCB ADDRESS
1006+	DC	A(0) TCB ADDRESS
1007+	9C	A(O) ADDRESS OF SNAP-SHOT LIST
1008+1H80003	DS	OH
1009+	SVC	51
1010 *		
1011 *		
1012	00	WHILE=R,WREGA=4,WOP=LT,WLITB=10,LABEL=ONE
1013+ONE	EQU	<b>\$</b>
1014+	ST	10, SAVE10
1015+	ST	11, SAVEL1
1016+	LR	10,4
1017*	LA	11,0
1018+	A -	11,=F*10*
1019+ 1020+	CR	10,1%
1021+	BNL	14 10 Seveno
1022+	Լ Ա	10, SAVE10 11, SAVE11
	N.O	▲▲2 ほべをにたた

1023+ 8 ٧4 1024+14 L 10, SAVE10 1025+ 11, SAVE11 Ł 1026+ 8 AONE 1027+V4 EQU 魚 1028 3,=F\*1\* Δ 4,==+1+ 1029 A 1030 Ą 5,==\*1\* 1031 6,=F+1+ A 1032 7,==11 A 1033 A 8,==11 1034 A 9,=F11 1035 10.=F\*1\* A 1036 ENDDO LABEL=ONE 1037 +В ONE 1038+CONE 8,SAVE8 L 1039 +9, SAVE9 L 1040+BCNE 10, SAVE10 L 1041+ ι 11, SAVE11 1042+AONE EQU 1043 \* 1044 \* 1045 SVAP ID=2+DCB=DUMPAREA+PDATA=REGS 1046+ CNOP 0,4 1047+ 9AL 1, THB0006 BRANCH AROUND PARAM LIST 1048 +DC AL1(2) ID NUMBER 1049 +DC AL1(0) 1050 +00 ALI(130) OPTION FLAGS 1051 +00 AL1(32) OPTION FLAGS 1052+ DC A(DUMPAREA) DCB ADDRESS 1053+ DC A(O) TCB ADDRESS A(O) ADDRESS OF SNAP-SHOT LIST 1054+ DC 1055+1H80006 DS. 0H 1056+ SVC 51 1057 Ł٨ 3,0 1058 LA 4,0 1059 LΔ 5,0 1060 ĹΔ 6,0 1061 LA 7+0 1062 LA 8+0 1063 LA 9+0 1064 LA 10,0 1065 \* 1066 \* 1067 00 LOWNUM=1,HGHNUM=30,8YNU4=4,DOLOOP=S,LABEL=TWO 1068+ ST 8,SAVE8 1069+ 9 + SAV 89 ST 1070+ ST 10, SAVE10 1071+ ST 11,SAVE11 1072+ L٨ 10,0 1073+ Å٠ 10,=F'1' 1074 +٤.4 11,0 11,=F'30' 1075+ Α 1076+ LA 9:0 1077 +9;=F141 A 1078+ 8 POINTER L 1079 +9, STACK(8) \$T 1080+ LA 8,4(8) 1081 +ST 10, STACK(8) 1082+ LA 8,4(8) 1083\* ST 11, STACK(8) 1084+ LA 8,4(8) 1085+ ST 8, POINTER 1086 +8,SAVE8 ١., 1087 +1. 9.SAVE9 1088+ L 10, SAVE10

1089+	L -	11,SAVELL -
1090+TW0	ST	8, SAVE8
1091+	ST	9+SAVE9
1092+	ST	10,SAVE10
1093+	ST	11, SAVE11
1094+		
	L	8, POINTER
1095+	S	8+=E*4*
1096+	L	11, STACK(8)
1097+	ŝ	
		8,=F*4"
1098+-	L	10, STACK(8)
1099+	5	8,=F*4*
1100+	Ĺ	9, STACK (8)
1101+	ST	8, POINTER
1102+	C	9,ZERO
1103+	BL	X7
1104+	C٩	10,11
1105+	8.4	Y7
1106+	B	27
1107+X7	CR	10,11
1108+	81.	¥7
1109+Z7	AR	10,9
1110+		
	L	8, POINTER
1111+	ST	9,STACK(8)
1112+	LA	8,4(8)
1113+	ST	
		10, STACK(8)
1114*	LA	8,4(8)
1115+	ST	11, STACK(8)
1116+	LA	8,4(3)
1117+	ST	8, POINTER
1118+	L	8,SAVER
1119*	Ľ	9, SAV 69
1120+		
	L	10, SAVE10
1121+	L	11,SAVE11
1122+	8	W7
1123+77	Ĺ	
		8, SAVE8
1124+	L	9, SAVED
1125+	L.	10,SAVE10
1126+	Ĺ	11,SAVE11
1127+	8	ATWO
1128+₩7	EQU	
1129		A 3,=F*2*
1130		A 4,=F*2*
1131		
		A 5,==F+2*
1132		A 6,=F*2*
1133		A 7,=F121
1134		A 8,=F*2*
		· · · · ·
1135		A 9,=F*2*
1136		A 10,=F*2*
1137	ENDDO	LABEL=TWO
1138+	8	TWO
1139+CTWO	L	8,SAVE8
1140+	L .	9, SAVE9
1141+BTWO	L	10. SAVE10
1142+	L	11,SAVE11
1143+ATWO	EQU	*
1144 * .	•	· · · ·
1145 *		
1146	CHAN	
	SVAP	ID=3,DCB=DUMPAREA,PDATA=REGS
1147+	ÇVOP	0,4
1148+	84L	1. THBOOD9 BPANCH AROUND PARAM LIST
1149+	00	ALI(3) TO NUMBER
1150+		
	00	AL1(0)
1151+	DC	ALI(130) OPTION FLAGS
1152+	00	ALI(32) OPTION FLAGS
1153+	DC	ALDUMPAREAL DCB ADDRESS
1154+	DC	A(O) TCB ADDRESS
		APAL PAG HONCOD

\*

1155+	DC	A(0) ADDRESS OF SNAP-SHOT LIST
1156+IH80009	DS	OH
1157+	SVC	51
1158	LA	3,0
1159	LA	4,0
1160	LA	5,0
1161	LA	6,0
1162	LA	7,0
1163	LA	
1164		8,0
	LA	9,0
1165	LA	19,0
1166 *		
1167 *		
1168	03	LOWNUM=3, BYLOC=Y, HGHREG=4, DOLDOP=7, WHILE=NN, X
		LABEL=THREE,WOP=GT,WLITA=7,WREGB=15
1169+	ST	8, \$AVE8
1170+	ST	9, SAVE9
1171*	ST	10, SAVE10
1172+	ST	11;SAVE11
1173+	LA	10,0
1174+	А	10,=F*3*
1175+	LR	11,4
1176+	L	9, Y
1177+	Ē	8, POINTER
1178+	ŝr	9, STACK (8)
1179+	LA	8,4(8)
1180+	ST	10, STACK(8)
1181+	LA	8,4(8)
1182+	ST	
1183+	LA	11,STACK(8)
1184+	ST	8,4(8) 9 DOINTED
1185+	-	8. POINTER
1186+	L	8+SAVE8
1187+	L	9, SAVE9
	L	10, SAVE10
1188+	L	11,SAVEL1
I189+THREE	sr	8, SAVE8
1190+	ST-	9, SAVE9
1191+	sr	10,SAVE10
1192+	ST	11,SAVE11
1193+	L,	8, POINTER
1194+	S -	8,=5*4*
1195+	L	11, STACK(8;
1196+	S	8,=F*4*
1197+	L,	10, STACK (8)
1198+	S	8,==*4*
1199+	Ľ.	9, STACK (8)
1200+	ST	8, POINTEP
1201+	С	9,ZERA
1202+	BL	X1.0
1203+	CR	10,11
1204+	84	Y10
1205+	В	210
1206+X10	CR	13,11
1207+	8L	Y10
1208+210	AR	10, 9
1209+	L	8, POINTER
1210+	ST	9, STACK181
1211+	LA	8,4(8)
12124	ST	LO, STACK(8)
1213+	LA	8,4(8)
1214+	ST	11, STACK(8).
1215 -	LA	8,4(3)
1216*	ST	8, POINTER
1217*	L	8, SAVE8
1218+	L	9, SAVE9
1219+	L	10,SAVEIO

1000		
1220+	Ļ	11, SAVE11
1221+	5	WIO
1222+Y10	L	8,SAVE8
1223+	1	9, SAVE9
1224+	L	10,SAVE10
1225+	L	IL, SAVEII
1226+	8	ATHREE
1227+W10	EQU	*
1228+	ST	10,SAVE10
1229+	ST .	11, SAVELL
1230+ .	LA	10,0
.1231+	A	10,=F*7*
1232+	LR	11,15
1233+	CR	10,11
1234+	BNH	T10
1235+	L	10, SAVE10
1236+	Ĺ,	11, SAVE11
1237+	8	V10
1238+T10	ĩ	10, SAVE10
1239+	L	IL,SAVE11
1240+	8	ATHREE
1241+V10	EQU	*
1242		
1243		A 3,#F*3*
		A 4,=F*3*
1244		A 5,=F'3!
1245		A 6,=F'3'
1246		$A \qquad 7_{1} = F^{\dagger} 3^{\dagger}$
1247		A 8,=F*3*
1248		Δ 9,=F*3*
1249	-	A 10,=F'3'
1250	ENDDO	LABEL=THREE
1251+	В	THREE
1252+CTHREE	L	8,SAVE8
1253+	L -	9, SAVE9
1254+BTHREE	L	10, SAVE10
1255+	L	11,SAVE11
1256+ATHREE	EQU	*
1257 *		
1258 *		
1259	SVAP	ID=4,DCB=DUMPAREA,PDATA=REGS
1260+	CNOP	0,4
. 1261+	84 L	1, TH80012 BRANCH AROUND PARAM LIST
1262+	00	AL1(4) JD NUMBER
1263+	DC	AL1(0)
1264+	DC	ALI(130) OPTION FLAGS
1265+	DC	AL1(32) OPTION FLAGS
1266+	DC	A(DUMPAREA) DCB ADDRESS
1267+	00	A(O) TCB ADDRESS
1268+	DC	A(O) ADDRESS OF SNAP-SHOT LIST
1269+THB0012	DS	OH
1270+	svc	51
1271 *		
1272 *		· · ·
1273	EXIT	
1274+	L	13,4(,13) POP UP SAVE AREA
1275+	โห	14,12,12(13) RESTORE REGISTERS
1276+	4VI	12(13),X'FF* FLAG EXIT
1277+	69	14 RETURN
1278	CLOSE	DUMPAREA
1279+	CVOP	0,4 ALIGN LIST TO FULLWORD
1280+	BAL	1+*+8 LOAD PEGI W/LIST ADDR
1281+	DC	ALI(128) OPTION BYTE
1282+	00 -	AL3(DUMPAREA) OCB ADDRESS
1283+		20 ISSUE CLOSE SVC
	-	un an an an an an Nar Nar Nar Nar Nar Nar Nar An
1007 CANCEDEL	~ ^	* * * * * *

184(0)

1284 SAVEAREA DC

				1
	1285	DS	0 F	
	1286 ZERO	DC	1F*0*	
	1287 SAVE8	DS	1F	
	1288 SAVE9	0 S	16	
	1289 SAVE10		-	
		DS	1F	
	1290 SAVE11	DS	1戶 .	
	1291 POINTER	DC	IE#04	
	1292 STACK	DS	100F	
	1293 Y	00	1 = * 1 *	
	1294 DUMPAREA	DC B	DDNAME=TEAM, DSDRG=PS, RECFM=VBA,	
	•		MACRE=W, BLKSIZE=882, LRECL=125	
			RRGRE-HIDERSELE-GOZIEREGE-IZJ	
	004.44			
	1296+*		DATA CONTROL BLOCK	
	1297+*			
1	1298+DUMPAREA	DÇ	OF O' GRIGIN ON WORD BOUNDARY	
1	1300+*		DIRECT ACCESS DEVICE INTERFACE	
•	1302+	00	BL16'0' FDAD, DVTBL	
	1303+	DC		
	13034	00	A(O) KEYLE, DEVT, TRBAL	
1	1305+*	•	COMMON ACCESS METHOD INTERFACE	
				· · · ·
· 1	1307+	DC	ALI(0) BUENO	
1	1308+	DC .	AL3(1) BUFCB	
1	1309+	00	AL2(0) BUFL	
1	1310+	00	BL2*01000000000000 * DSDRG	
	1311+	DČ	A(1) INBAD	
		00	ALLI KINDAD	
1	1313+*			
1	1.71.3***		FOUNDATION EXTENSION	
	1315+	ÐČ	BL1'0000000' BFTEK, BFLN, HIARCHY	
	1316+	3G	AL3(1) EODAD	
;	1317+	DC	BL1'01010100' RECEM	
1	1318+	00	AL3(0) EXLST	
1	320+*		FOUNDATION BLOCK	
-			Sanaktich DESOK	
1	1322+	DC	CL8 TEAM DONAME	•
	1323+	00		
			BL1'0000010' CFLGS	7
	324+	00	BL1'0000000' IFLG	
1	325+	DC	BL210000000000000 MACR	
1	327+*		BSAM-BPAM-QSAM INTERFACE	
	329+	00	BL1*0000000* REP.1	
1	330+	DC	AL3(1) CHECK, GERR, PERR	
1	1331+	DC	A(1) SYNAD	
	332+	00	HTO' CIND1, CIND2	
	333+	DC	AL2(882) BLKSTZE	
	334+	DC DC	F'O' WCPD, WCPL, OFFSR, OFFSW	
		00	A(1) IOBA	
	336+	DC	ALL(O) NCP	
I	337+	DC	AL3(1) EOBR, EOBAD	
	220.44			
t	,339+*		BSAM-BPAN INTERFACE	
	2/11			
	341+	00	A(1) EDBW	
-		DC	HO DIRCT	
	.343+	00	AL2(125) LRECL	.*
	344+	DC	A(1) CNTRL, NOTE, POINT	
	345	END	•	1. A 2.4
	.346		#F10 <sup>8</sup>	·
	.347		#F•1*	
1	348		=F*30*	

X

1349	= - 41
1350	===121
1351	= [131
1352	*F171

# Sample Program B Illustrating a DO Macro Coded Incorrectly

#### STMT SOURCE STATEMENT

962 FRISBEE 963+FRISBEE	ENTER DS	12,SAVEAREA OH
964+ 965+	UCTNO	FRISBEE DECLARE NAME ENTRY
966+		*,12 DECLARE BASE ADDRESSIBILITY.
967+	BALR	15,0 (INITIAL ADDRESSIBILITY)
968+	B DC	12(,15) BRANCH AROUND ID FIELD
969+		AL1(7), CL7'FRISBEE' ID LENGTH AND ID
970+	BCTR	15,0 (RESET INITIAL ADDRESSIBILITY
971+	BCTR STM	15.0 ABSOLUTE ENTRY POINTS.
972+		14,12,12(13) SAVE REGISTERS
973+	LR	12,15 SETUP BASE REGISTER.
974+	ST	13, SAVEAREA+4 CHAIN BACK
	LA	O, SAVEAPEA CHAIN FORWARD
975+	ST	0,8(0,13)
976+	LR	13.0 SET UP SAVE AREA POINTER
977+ 070 CT+DT	USING	
978 START	OP EN	(OUMPAREA, OUTPUT)
9794	CNOP	0,4 ALIGN LIST TO FULLADRD
980+START	BAL	1,*+8 LOAD REG1 W/LIST ADDR.
981+	0C	ALI(143) OPTION BYTE
982+	00	AL3(DUMPAREA) DCB ADDRESS
983+	SVC	19 ISSUE OPEN SVC
984	LA	2,0
985	LA	3,0
986	LA	4,0
987	LA	5,0
988	LA	6,0
989	LA	7,0
990	LA	8,0
991	LA	9,0
992	LA	19,0
993	LA	11,0
994	LA	14,0
995	LA	15,0
996 * 997 *		
998	SNAP	
9994	CNOP	ID=1,DCB=DUMPAREA,PDATA=REGS
1000+	BAL	0,4 1 THROOOT DRANCH ADDING DADAW (TOT
1001+	DC	I, THBOOO3 BRANCH AROUND PARAM LIST
1002+	00	ALI(I) ID NUMBER ALI(O)
1003+	DC	ALI(130) DPTYON FLAGS
1004+	DC	ALI(32) OPTION FLAGS
1005+	DC	ALDUMPAREAT DCB ADDRESS
1006+	DC	A(0) TCB ADDRESS
1007+	90	A(O) ADDRESS OF SNAP-SHOT LIST
1008+1880003	0s	OH
1009+	SVC	51
1010 *		
1011 ×		
1012	00	WHILE=R; WREGA=4, WOP=LT, LABEL=ONE
1013+ONE	EQU	#

1016+ 1017 1018+ 1019 1020 1021 1022 1023 1024 1025 1026 1027 1028+ 1029*CONE 1030+ 1031+BONE 1032+ 1033+AONE	L	10, SAVE10 11, SAVE11 10,4 *,NO RIGHT OPERAND SPECIFIED ON DO WHILE CONDITION BONE A 3, =F'1* A 4, =F'1* A 5, =F'1* A 6, =F'1* A 6, =F'1* A 9, =F'1* A 10, =F'1* LABEL=ONE ONE 8, SAVE8 9, SAVE9 10, SAVE10 11, SAVE11 *
1038+ 1039+ 1040+ 1041+ 1042+ 1043+ 1044+ 1045+ 1046+IHB0006 1047+ 1048 * 1049 * 1050 1051+ 1052+ 1053+ 1053+ 1055 1056+ 1058+ 1059+ 1059+	CNOP BAL DC DC DC DC DC DC DC DC DC SVC EXIT L M WVI BR CLOSE CNOP BAL DC	AP JD=2,DCB=DUMPAREA,PDATA=REGS 0,4 1,IHB0006 BRANCH ARDUND PARAM LIST AL1(2) ID NUMBER AL1(2) ID NUMBER AL1(3) AL1(130) OPTION FLAGS A(1) OPTION FLAGS A(0) ADDRESS ADDRESS A(0) ADDRESS OF SNAP-SHOT LIST 04 51 4 4 4 4 4 4 4 4 4 4 4 4 4
1061 SAVEAREA ( 1062 ( 1063 ZEFO ( 1064 SAVE8 ( 1065 SAVE9 ( 1066 SAVE10 ( 1067 SAVE11 ( 1068 POINTER (	DC DS DC DS DS DS DC DS DC DS DC DS DC DS DC DS	18A(0) OF 1F'0' 1F 1F 1F 1F 1F 1F 0' 100F DDNAME=TFAM,DSORG=PS,RECFM=VBA, MACRF=W,BLKSIZE=882,LRECL=125
1072+* 1073+* 1074+DUMPAREA ( 1076+*	DC 0	DATA CONTROL BLOCK DF*O* ORIGIN ON WORD BOUNDARY DIRECT ACCESS DEVICE INTERFACE

х

DC	A(O) KEYLE, DEVT, TRBAL
	COMMON ACCESS METHOD INTERFACE
DC DC DC DC DC	AL1(0) 8UFN0 AL3(1) 8UFC8 AL2(0) 8UFL BL2*0100000000000000000000000000000000000
	FOUNDATION EXTENSION
DC DC DC DC	8L1'00000000' BFTEK,BFLN,HIARCHY AL3(1) E0DAD BL1'01010100' RECFM AL3(0) EXLST
	FOUNDATION BLOCK
DC DC DC	CL8'TEAM' DDNAME BL1'00000010' OFLGS BL1'00000000' TELC

	00	UDRAME	
1099+	DC	8L1'00000010' OFLGS	
1100+	DC	BL1 0000000 FIFLG	
1101+	DC	BL2:000000000000000000 MACR	

1078+

1079+ 1081+\*

1083+ 1084+ 1085+ 1086+ 1087+ 1089+\* 1091+ 1092+ 1092+ 1096+\* 1096+\*

1103+\*

ĐC

#### BSAM-BPAM-QSAM INTERFACE

BL16'3' FDAD, DVTBL

1105*	DÇ	BL1*0000000* REP1
1106+	DC	AL3(1) CHECK, GERR, PERR
1107+	00	A(1) SYMAD
1108+	00	H'O' CINDI, CIND2
1109+	DC	AL2(882) BLKSIZE
L110+	DC	F'O' WCPC, WCPL, CEFSR, DEFSW
1111+	DC	A(L) TOBA
1112+	DC	AL1(0) NCP
1113*	DC	AL3(1) EOBR, EOBAD
1115+*		BSAM-BPAM INTERFACE
1117+	DC	A(1) E0BW
1118+	DC	HIOI DIRCT
1119+	DC	AL2(125) LRECL
1120+	DC	A(I) CNTRL, NOTE, POINT
1121	END	CONTRACTS CONTRACTIONS
1122		#F*14

# Sample Program C Illustrating a DO Macro Coded Incorrectly

STMT SOURCE	STATE	1ENT -
962 FR1SBEE 963*FRISBEE 964+	DS	12,SAVEAREA OH F9ISBEE DECLARE NAME ENTRY
965+ 966+ 967+	USING	*,12 DECLARE BASE ADDRESSIBILITY. 15,0 (IMITIAL ADDRESSIBILITY).
968+ 969+	DC BC TR	12(+15) BEANCH AROUND ID FIELD AL1(7),GL7*FRISBEE* ID LENGTH AND ID 15+0 (PESET INITIAL ADDRESSIBILITY
970+ 971+ 972+	BC TR STM LR	15,0 ABSOLUTE ENTRY POINTI. 14,12,12(13) SAVE REGISTERS 12,15 SETUP BASE PEGISTER.

12.3

973+	ST	13, SAVEAREA+4 CHAIN BACK
974+	LA	O, SAVEAREA CHAIN FURWAR)
975+	\$T	0,8(0,13)
976+	EP -	13.0 SET UP SAVE AREA POINTER
977+		SAVEAREA, 13 AND ADJRESSABILITY
978 START	OP EN	(DUMPAREA, OUTPUT)
979+ 980+START	CNOP BAL	0,4 ALIGN LIST TO FULLAJRD 1,*+8 LOAD REGI W/LIST ADDR.
- 981+	DC.	ALI(143) OPTION BYTE
982+	00 -	AL3(DUMPA9EA) DCB ADDRESS
983+	SVC	19 ISSUE OPEN SVC
934	LA	2,0
985	LA	3,0
986	LA	4,0
987 988	LA	5,0
989	LA LA -	6,0 7,0
990	LA	8,0
991	LA	9,0
992	LA	10,0
993 ·	LA	11,0
994	LA	14:0
995	LA	15,0
996 * 997 *		
998	S	NAP ID=1,DCB=DUMPAREA,PDATA=REGS
999+	CNOP	0.4
1000+	BAL	1, IHB0003 BRANCH ARDUND PARAM LIST
1001+	DC	ALI(I) ID NUMBER
1002+	0C	AL1(0)
1003+ 1004+	DC DC	ALI(130) OPTION FLAGS ALI(32) OPTION FLAGS
10050	DC	ACTUSES OFFICE FLAGS
1006+	DČ	A(0) TCB ADDRESS
1007+	DC	A(O) ADDRESS OF SNAP-SHOT LIST
1008+IH80003	DS	OH
1009+ 1010 *	SVC	51
1010 +		
1012	00	LOWNUM=1,LOWREG=5,HGHNUM=30,BYNUM=1,DOLOOP=W, X
		LABEL=TWO
1013+	ST	8,SAVES
1014+	ST	9, SAVE9
1015+ 1016+	ST ST	10,SAVE10
1017+	51 LA	11,SAVE11 10,0
1018+	Ą	10,====1
1019		*, MORE THAN ONE LOWER DO LOOP RANGE SPECIFIED
1020+	8	CTWO
1021 1022		A 3,=F <sup>1</sup> 2!
1023		A 4,∞F12† A 5,=F12†
1024		A 61=F*2*
1025		A 7,=F+2+
1026		A 8,=====2*
1027		Α 9,=F <sup>*</sup> 2*
1028 1029	ENDDO	A 10,=F*2* {ABFL=TWO
1030+	- E NUU-J - B	TW9
	-	
1031+CTW0	L	8, SAVE8
1032+		9, SAVE9
1033+BIWD 1034+	L L	10, SAVE10 11, SAVE11
1035+ATWO	EQU	*
1036 *		

1037 *			
-1038		NAP ID=2, DCB=DUAPAREA, PDATA=REGS	
1039+	CNOP	0,4	
1040+	BAL	1, IH30006 BRANCH AROUND PARAM LIST	
1041+ 1042+	0C DC	ALL(2) ID NUMBER ALL(0)	
1043+	0C	ALI(130) OPTION FLAGS	
1044+	00	ALI(32) OPTION FLAGS	
1045+	DC	A (DUMPAREA) DCB ADDRESS	
1046+	ĐC	A(0) TCB ADDRESS	
1047+	ÐČ	A(O) ADDRESS OF SNAP-SHOT LIST	
1048+IHB0006	DS	01	· · · ·
1049+	SVC	51	
1050 *	-	and the second	and a second second
1051 *	<i></i>		
1052 1053+	EXIT		-
1054+	L LM	13,4(,13) POP UP SAVE AREA	
1055+	MVI	14,12,12(13) RESTORE REGISTERS 12(13),X'FF' FLAG EXIT	
1056+	BR	14 RETURN	
1057	CLOSE		
1058+	CNOP	0,4 ALIGN LIST TO FULLWORD	
1059+	BAL	1,*+8 LOAD FEGI W/LIST ADDR	
1060*	00	ALI(128) OPTION BYTE	
1061+	DC	AL3(DUMPAREA) DCB ADDRESS	
10624	SVC	20 ISSUE CLOSE SVC	
1063 SAVEAPEA	DC	194701	
1065 SAVEAREA	DS	13A(0) OF	
1065 ZERO	DC -	1 == 0 =	
1066 SAVE8	DS	16	
1067 SAVE9	ÐS	1 F	
1068 SAVELO	ÐS	1 F	
1069 SAVE11	DS	16	
1070 POINTER -		1F*0*	
1071 STACK	DS	100F	
1072 DUMPAREA	DCB	DONAME=TEAM, DSORG=PS, RECFM=VBA,	x
		MACRF=W, BLKSIZE=882, LRECL=125	
1074+*		DATA CONTROL BLOCK	
1075+*			
1076+DUMPAREA	DC .	OF'O' ORIGIN ON WORD BOUNDARY	
1078+*		DIRECT ACCESS DEVICE INTERFACE	
		DI LUT AUCUSS DEVICE INTERFACE	
1080+	DC	BL16'O' FDAD, DVTBL	
1081+	DC	A(O) KEYLE, DEVT, TRBAL	
1002.0		· · · · · · · · · · · · · · · · · · ·	
1083+*	_	COMMON ACCESS METHOD INTERFACE	
1085+	DC	ALI(0) BUF10	
1086+	DC	AL3(1) BUFCB	
1087+	DC	AL2(0) BUFL	
1088+	DC	BL2101000000000000 DSDRG	
1089+	00	A(I) IOBAD	
1091+*		FOUNDATION EXTENSION	
1.0.00			
1093+	DC	BL1'0000000 BFTEK, BFLN, HIARCHY	
1094+ 1095+	DC DC	AL3(1) E00AD	
	0C 0C	BL1'01010100' RECFM AE3(0) EXLST	
** / U ·	46	ACREVI CALOI	
1098+*		FOUNDATION BLOCK	,

•

•

.

125

•

1100+	DC	CL8+TEAM+ DONAME
1101+	DC	BL1'00000010' OFLGS
1102+	00	8L1'00000000' IFLG
1103+	DC	8L2.00000000000000 MACR
1105+*		BSAM-BPAM-USAH INTERFACE
1107+	DC	BL1:000000001 RER1
1108+	DC	AL3(1) CHECK, GERR, PERR
1109+	DC	A(1) SYNAD
1110+	DC	H'O' CINDL, CIND2
1111+	DC	AL2(882) BLKSIZE
1112+	0C	FTOT WOPO, WOPL, OFFSR, OFFSW
1113+	DC .	
1114+	DC	ALI(O) NCP.
1115+	DC	AL3(1) EOBR, EOBAD
1117+*		BSAM-BPAM INTERFACE
1119+	DC	A(1) E08W
1120+	DC	H'O' DIRCT
1121+	DC	AL2(125) LPECL
1122+	DC	A(1) CNTRL, NOTE, POINT
1123 ·	END	
1124		=F*1*
1125		=F+2+
1125		=F121

# Sample Program D Illustrating a DO Macro Coded Incorrectly

STHT SOURCE STATEMENT

962 FRISBEE	ENTER	12, SAVEAREA
963+FRISBEE	DS	04
964+	ENTRY	FRISBEE DECLARE NAME ENTRY
965+	USING	*+12 DECLARE BASE ADDRESSIBILITY.
966+	BALR	15,0 (INITIAL ADDRESSIBILITY).
967+	8	121,15) BEANCH AROUND ID FIELD
968+	DC	AL1(7) + CL7* FRJ SBEE* ID LENGTH AND ID
969+	BCTR	15, J (RESET INITIAL ADDRESSIBILITY
970+	BCTR	15,0 ABSOLUTE ENTRY POINT).
971+	STM	14,12,12(13) SAVE REGISTERS
972+	LR	12,15 SETUP BASE REGISTER.
\$73+	ST	13, SAVEAPEA+4 CHAIN BACK
974+	LA	0, SAVEAREA CHAIN FURWARD
975+	ST	0,8(0,13)
976+	LR	13.0 SET UP SAVE AREA POINTER
977+	USING	SAVEAREA, 13 AND ADDRESSABILITY
978 START	OP EN	(OUMPAREA, OUTPUT)
979+	CNOP	0,4 ALIGN-LIST TO FULLWORD
980+START	BAL	1,*+8 LOAD REGI W/LIST ADDR.
981+	DC.	ALI(143) OPTION BYTE
982+	DC	AL3(DUMPAREA) DCB ADDRESS
983+	SVC	19 ISSUE OPEN SVC
984	LA	2,0
<b>9</b> 85	LA	3,0
986	ĽΑ	4,0
987	LA	5,0
988	LA	6,0
989	LA	7,0
990	LA	8,0
991	LA	9,0
992	LA	10,0

993 LA 11 + 0994 LA 14,0 995 LA 15.0 996 \* 997 \* 998 \* 999 \* 1000 SMAP. ID=1,DCB=DUMPAREA,PDATA=REGS 1001+ CNOP 0,4 1002+ BAL 1. [HB0003 BRANCH ARDUND PARAM LIST 1003 +DC AL1(1) ID NUMBER 1004+ DC AL1(0) 1005+ DC AL1(130) OPTION FLAGS 1006+ DC AL1(32) OPTION FLAGS 1007 +DC ALDUMPAREAL DCB ADDRESS 1008+ A(0) TCB ADDPESS DC. 1009 +DC A(0) ADDRESS OF SNAP-SHOT LIST 1010+IH80003 DS 04 1011+ SVC 51 1012 \* 1013 \* 1014 00 LOWNUM=3, BYLOC=Y, HGHREG=24, DOLODP=2, LABEL=THREE, Х WHILE=NN, WOP=GT, WLITA=7, WREGB=15 1015+ SŤ 8.SAVE8 1016+ SΤ 9, SAVE9 ST " 1017+ 10,SAVE10 \* 1018+ ST 11,SAVE11 1019+ LA 10,0 1020+ 10,=F'3+ A 1021 \*,INVALID NUMBER FOR 'HGHREG' SPECIFIED IN DO 1022 \*,L00P 1023+ CTHREE 8 1024 3,=F'3\* A 1025 4,=F+3+ Δ 1026 Δ 5,=F131 1027 A 6,=F+3\* 1028 A 7,=F\*3\* 1029 8,=F+3! Δ. 1030 9.=F+3+ А 1031 10,=F\*3\* А 1032 ENDDO LABEL=THREE 1033+ B -THREE 1034+CTHREE L 8, SAVE8 1035+ t. 9, SAVE9 1036+8THREE L 10, SAVE10 1037+ Ł 11.SAVE11 1038+ATHREE EQU yłe: 1039 \* 1040 \* 1041 SNAP ID=2,DCB=DUMPAREA,PDATA=REGS 1042+ CNDP 0,4 1043+ BAL 1. THBOOOG BRANCH ARDUND PARAM LIST 1044+ ĐC AL1(2) ID NUMBER 1045+ DC AL1(0) 1046+ ЭG ALI(130) OPTION FLAGS 1047+ ALI(32) OPTION FLAGS DC 1048+ A (DUBPAREA) OCB ADDRESS DC 1049+ DC A()) TCB ADDRESS 1050+ A(O) ADDRESS OF SNAP-SHOT LIST 00 1051±1H80006 D S 0H 1052+ SVC 51 1053 \* 1054 \* 1055 EXIT 1056+ L 13,4(,13) OP UP SAVE AREA 1057+ LM 14,12,12(13) RESTORE REGISTERS

YV T 12(LJE, X\*FET FLAG EXIT BR 14 RETURN CLOSE DUMPAREA CNOP 0,4 ALIGN LIST TO FULLWORD BAL 1 \*\*+8 LOAD REGI W/LIST ADDR ALI(128) OPTION BYTE DC ÐC AU3(DUMPAREA) DCB ADDRESS SVC 20 ISSUE CLOSE SVC 1066 SAVEAREA DC 184(0) · DS 0F 16.0. 00 DS 1F DS 1F 1071 SAVE10 0.5 1 F 1072 SAVEL1 1F 0S 1073 POINTER DC. 1 = \* 0\* DS 100F D¢ 16.1. 1076 DUMPAREA DCB DDNAME=TEAM, DSORG=PS, RECFM=VBA, MACRF=W, BLKSIZE=882, LRECL=125 .

1058+

1059+

1060

1061 +

1062+

1063+

1064+

1065+

1067

1068 ZERO

1069 SAVE8

1070 SAVE9

1074 STACK

1075 Y

1121+#

.

1078+* 1079+*		DATA CONTROL BLOCK
1080+DUMPAREA	∿ DÇ	OF OF ORIGIN ON WORD BOUNDARY
1082+*		DIRECT ACCESS DEVICE INTERFACE
1084+	DC-	BL16# 3* FDAD, DVT8L
1035+	DC	A(O) KEYLE, DEVT, TRBAL
1087+*	•	COMMON ACCESS METHOD INTERFACE
1089+	DÇ	ALI(0) BUENO
1090+	90	AL3(1) BUFCB
1091+	9C	AL2(0) BUFL
1092+	DC	BL2*010000000000000 + DSDRG
1093+	DC	A(1) IOBAD
1095+*		FOUNDATION EXTENSION
1097+	DC	BL11000000001 BFTEK, BFLN, HIARCHY
1098+	DC	AL3(1) ENDAD
1099+	DC	BL1+01010100+ BECEM

1098+	DC	AL3(1) EODAD
1099+	DC	BL1*01010100* RECEM
1100+	DC	AL3(0) EXLST
1102+*		FOUNDATION BLOCK

1104+	DC	CL8*TEAM* DDNAME	
1105+	DC .	BL1.00000010. OFLGS	
1106+	DC	BL1+00000000 IFLG	
1107+	DC ,	BE2*00000000001000J0*	MACR

1109+\* BSAM-BPAM-USAM INTERFACE 1111+ ÐC BL1'00000000' RER1 AL3(1) CHECK, GERR, PERR 1112+ ÐC 1113+ DC A(1) SYHAD 1114+ HIO! CIND1: CIND2 00 1115+ DC AL2(882) BLKSIZE F\*O\* WCPO, WCPL, DEESR, OFESW 1116+ DC 1117 +DC A(1) 1084 1118+ 00 ALI(0) MCP 1119+ 90 AL3(1) EOBR, EOBAD

#### BSAM-BPAM INTERFACE

Х

.

. .

· ·

. . . .

. .

### APPENDIX B

# THE LEAVE MACRO

# The Source Listing for the LEAVE Macro

#### STMT SOURCE STATEMENT

514	*		THE LEAVE MACRO PROVIDES A MEANS OF ABNORMALLY TER-
515	*		MINATING A LOOPING STRUCTURE. IT MAY INVIKE A CON-
516	*		DITIONAL OR UNCONDITIONAL BRANCH FROM WITHIN A LOOP.
517	*		CONDITION PARAMETERS, WHICH MAY DE MAY NOT BE IN-
518	<b>A</b> c		CLUDED, APE SPECIFIED IN MUCH THE SAME MANNEP AS A
519	ж.		DO WHILE CONDITION. THE 'COND' PARAMETER IS MADE
520	*		NON-MULL WHEN A CONDITIONAL LEAVE IS DESIDED, AND
521			THE 'LABEL' PARAMETER SPECIFIES THE DESTINATION OF
522		~	THE BRANCH WHICH SHOULD NORMALLY BE TO THE END OF
523			THE LOOPING STRUCTURE. LEAVE STATEMENTS SHOULD NOT
524			BE USED FOR BACKWARD BRANCHING.
525			OC OSED THE DRUCHARD DRANCHINGS
526			
527		MACRO	
528			
220		66486	&LABEL=,&COND=,&OPRATOR=,&REGA=,&LOCA=,&LITA=,&REGB=, x
529			&LOCB=, &LITB=
	CI & D	LCLA	ELAB
	εlab	SETA	& SY SNDX
531		AGO	• SKIP
532			
533			
534			INITIALLY THE "COND" PARAMETER IS YESTED TO DETER-
535			MINE IF THE LEAVE STATEMENT IS CONDITIONAL OR UN-
536			CONDITIONAL. IF UNCONDITIONAL, THEN A BRANCH TO
537			'LABEL' WILL OCCUR. 'LABEL' MUST BE SPECIFIED
538			EITHER IN ANOTHER MACRO OR IN A PROGRAM INSTRUC-
539			TIONO
540			
541			
	●SKIP	AIF	(*&COND* NE **).A
543		B	AGLABEL .
544		AGO	• STOP
545			and the second
546			
547			WHEN THE LEAVE STATEMENT IS CONDITIONAL, REGISTERS
548			10 AND 11 (HORKING REGISTERS) ARE SAVED. THEY WILL
549			BE USED TO TEST THE LEAVE CONDITION.
550			
551		<i>4</i> <b>-</b>	
552	¢ A	ST	10, SAVE10
553		ST	11, SAVE11 .
554		AGO	• 8
555			· · · · · · · · · · · · · · · · · · ·
556			
557			THE FOLLOWING SECTION OF CODE DETERMINES THE LEFT
558			OPERAND OF THE LEAVE CONDITION. IT MAY BE SPEC-
559			IFTED BY A REGISTER (REGAL, A LOCATION (LOCA) OR BY
560			A LITERAL (LITA). THE OPERAND IS LOADED INTO REG-
561	<b>2</b> 4		ISTER 10. ILLEGALLY SPECIFIED OPERANDS WILL GEN-

562			
	*		ERATE A MACRO ERROR CONDITION AND THE LEAVE MACRO
563			WILL NOT BE EXECUTED.
564	-		
565			
	o o B	AIF	(*&REGA* EQ **).C
567		ATF	(T'GREGA NE 'N').ERR1
568		AIF	("&RFGA" GT *15").ERR2
569		LR ·	10, EREGA
570		AGN	•E
	• C	ATE	(*&LOCA' EQ **).D
572		L	10, 6LOCA
573		AGO	•F1
	• D	ALE	("ELITA" EQ "").ERR3
575		ATF	("ELITB" NE ""), ERR4
	+RETURN	LA ·	10,0
577		Å.	10,=F*&LITA*
578		AGO	• E2
579		AIF	("&LOCA" NE ""I.ERR5
	•EL	ATF -	(*&LITA* NE **).ERRS
581		AGባ -	• E2
582			
583		•	
584			THE RIGHT OPERAND IS DETERMINED IN THE NEXT SECTION
585		• •	OF CODE. IT TOO MAY BE SPECIFIED BY REGISTER
586			[REGB], BY LOCATION (LOCB) OR BY A LITERAL (ITTRL
587		• 、	THE RIGHT OPERAND IS LOADED INTO REGISTER 11 AND
588		, , ,	COMPARED TO REGISTER 10 (LEFT OPERANDIA TE THE
589			RIGHT OPERAND HAS BEEN ILLEGALLY SPECIFTED . AN
590		•	ERROR MESSAGE WILL BE PRINTED AND THE MACRO WILL NOT
591			BE EXECUTED.
592			
593			
	•E2	ATE	(*&PEG8* EQ **).F
595		AIF	T'EREGE NE 'N'I.ERR6
596		AIF	(*GREGB* GT *15*1.ERR7
597		LR	11,GREGB
598	r	AGO	
599 600	• F	AIF	(*&LOCB* EQ **).G
601		L	11, SLCC8
001		100	*F1
602	. 6	AGO	ALCETTOA DO ALL DOAS
602 603	• G	AIF	(*&LITB* EQ **).ERR8
603	• G	AIF La	11,0
603 604	• G	AIF LA A	11,0 11,=F*&LIT8*
603 604 605		AIF La A Ago	11,0 11,=F*&LITB* •H2
603 604 605 606	۰H	AIF La Ago AIF	11,0 11,=F*&LITB* •H2 (*&L^CB* NE **).ERR9
603 604 605 606 607	•H •H1	AIF LA Agg AIF AIF	11,0 11,=F*&LITB* •H2 (*&L^CB* NE **).ERR9 (*&LITB* NE **).ERR9
603 604 605 606 607 608	•H •H1	AIF LA Agg AIF AIF CR	11,0 11,=F*&LITB* •H2 (*&L^CB* NE **).ERR9 (*&LITB* NE **).ERR9 10,11
603 604 605 606 607	•H •H1 •H2	AIF LA Agg AIF AIF	11,0 11,=F*&LITB* •H2 (*&L^CB* NE **).ERR9 (*&LITB* NE **).ERR9
603 604 605 606 607 608 609	•H •H1 •H2	AIF LA Agg AIF AIF CR	11,0 11,=F*&LITB* •H2 (*&L^CB* NE **).ERR9 (*&LITB* NE **).ERR9 10,11
603 604 605 606 607 608 609 610	•H •H1 •H2 *	AIF LA Agg AIF AIF CR	11,0 11,=F*&LITB* •H2 (*&L^CB* NE **).ERR9 (*&LITB* NE **).ERR9 10,11 •1
603 604 605 606 607 608 609 610 611	•H •H1 •H2 ≭ *	AIF LA Agg AIF AIF CR	11,0 11,=F*&LITB* •H2 (*&L^CB* NE **).ERR9 (*&LTB* NE **).ERR9 10,11 •I THE FOLLOWING CODE SEGMENT DETERMINES THE LEAVE
603 604 605 606 607 608 609 610 611 612	•H •H1 •H2 ≭ * *	AIF LA Agg AIF AIF CR	11,0 11,=F*&LITB* •H2 (*&LCB* NE **).ERR9 (*&LITB* NE **).ERR9 10,11 •I THE FOLLOWING CODE SEGMENT DETERMINES THE LEAVE CONDITION OPERATOR AND GENERATES THE APPROPRIATE
603 604 605 606 607 608 609 610 611 612 613	•H •H1 •H2 * * * *	AIF LA Agg AIF AIF CR	11,0 11,=F*&LITB* .H2 (*&L^CB* NE **).ERR9 (*&LTB* NE **).ERR9 10,11 .T THE FOLLOWING CODE SEGMENT DETERMINES THE LEAVE CONDITION OPERATOR AND GENERATES THE APPROPRIATE BRANCH INSTRUCTION. A MISSING OR ILLEGAL OPERATOR
603 604 605 606 607 608 609 610 611 612 613 614	•H •H1 •H2 * * * *	AIF LA Agg AIF AIF CR	11,0 11,=F*&LITB* +H2 (*&LTB* NE **).ERR9 (*&LITB* NE **).ERR9 10,11 .T THE FOLLOWING CODE SEGMENT DETERMINES THE LEAVE CONOTION OPERATOR AND GENERATES THE APPROPRIATE BRANCH INSTRUCTION. A MISSING OR ILLEGAL OPERATOR WILL GENERATE AN ERROR MESSAGE AND THE MACRO WILL
603 604 605 606 607 608 609 610 611 612 613 614 615	•H •H1 •H2 * * * * *	AIF LA Agg AIF AIF CR	11,0 11,=F*&LITB* .H2 (*&L^CB* NE **).ERR9 (*&LTB* NE **).ERR9 10,11 .T THE FOLLOWING CODE SEGMENT DETERMINES THE LEAVE CONDITION OPERATOR AND GENERATES THE APPROPRIATE BRANCH INSTRUCTION. A MISSING OR ILLEGAL OPERATOR
603 604 605 606 607 608 609 610 611 612 613 614 615 616	•H •H1 •H2 * * * * * *	AIF LA Agg AIF AIF CR	11,0 11,=F*&LITB* +H2 (*&LTB* NE **).ERR9 (*&LITB* NE **).ERR9 10,11 .T THE FOLLOWING CODE SEGMENT DETERMINES THE LEAVE CONOTTION OPERATOR AND GENERATES THE APPROPRIATE BRANCH INSTRUCTION. A MISSING OR ILLEGAL OPERATOR WILL GENERATE AN ERROR MESSAGE AND THE MACRO WILL
603 604 605 606 607 608 609 610 611 612 613 614 615 616 617 618 619	•H •H1 •H2 * * * * * *	AIF LA Agg AIF AIF CR	11,0 11,=F*&LITB* +H2 (*&LTCB* NE **).ERR9 (*&LITB* NE **).ERR9 10,11 •I THE FOLLOWING CODE SEGMENT DETERMINES THE LEAVE CONDITION OPERATOR AND GENERATES THE APPROPRIATE BRANCH INSTRUCTION. A MISSING OR ILLEGAL OPERATOR WILL GENERATE AN ERROR MESSAGE AND THE MACRO WILL NOT BE EXECUTED.
603 604 605 606 607 608 609 610 611 612 613 614 615 616 617 618	•H •H1 •H2 * * * * * *	AIF LA AGO AIF AIF CR AGO	11,0 11,=F*&LITB* +H2 (*&LTCB* NE **).ERR9 (*&LITB* NE **).ERR9 10,11 •I THE FOLLOWING CODE SEGMENT DETERMINES THE LEAVE CONDITION OPERATOR AND GENERATES THE APPROPRIATE BRANCH INSTRUCTION. A MISSING OR ILLEGAL OPERATOR WILL GENERATE AN ERROR MESSAGE AND THE MACRO WILL NOT BE EXECUTED. (*&OPRATOR* EQ **).ERR10
603 604 605 606 607 608 609 610 611 612 613 614 615 616 617 618 619 620 621	•H •H1 •H2 * * * * * *	AIF LA AGO AIF AIF CR AGO AIF AIF AIF	11,0 11,=F*&LITB* H2 (*&LTCB* NE **).ERR9 (*&LITB* NE **).ERR9 10,11 THE FOLLOWING CODE SEGMENT DETERMINES THE LEAVE CONDITION OPERATOR AND GENERATES THE APPROPRIATE BRANCH INSTRUCTION. A MISSING OR ILLEGAL OPERATOR WILL GENERATE AN ERROR MESSAGE AND THE MACRO WILL NOT BE EXECUTED. (*&OPRATOR* EQ **).ERRIO (*&OPRATOR* EQ **).ERRIO (*&OPRATOR* EQ **).ERRIO
603 604 605 606 607 608 609 610 611 612 613 614 615 616 617 618 620 621 622	•H •H1 •H2 * * * * * *	AIF LA AGO AIF CR AGO F AIF AIF AIF	11,0 11,=F*&LITB* H2 (*&LTCB* NE **).ERR9 (*&LITB* NE **).ERR9 10,11 THE FOLLOWING CODE SEGMENT DETERMINES THE LEAVE CONDITION OPERATOR AND GENERATES THE APPROPRIATE BRANCH INSTRUCTION. A MISSING OR ILLEGAL OPERATOR WILL GENERATE AN ERROR MESSAGE AND THE MACRO WILL NOT BE EXECUTED. (*&OPRATOR* EQ **1.ERR10 (*&OPRATOR* EQ **1.ERR10 (*&OPRATOR* EQ **1.ERR10 (*&OPRATOR* EQ **1.ERR10 (*&OPRATOR* EQ **1.ERR10 (*&OPRATOR* EQ **1.ERR10 (*&OPRATOR* EQ **1.ERR10
603 604 605 606 607 608 609 610 611 612 613 614 615 616 617 618 620 621 622 623	•H •H1 •H2 * * * * * *	AIF LA AGO AIF AIF CR AGO F AIF AIF AIF AIF	11,0 11,=F*&LITB* H2 (*&LTCB* NE **).ERR9 (*&LITB* NE **).ERR9 10,11 .T THE FOLLOWING CODE SEGMENT DETERMINES THE LEAVE CONDITION OPERATOR AND GENERATES THE APPROPRIATE BRANCH INSTRUCTION. A MISSING OR ILLEGAL OPERATOR WILL GENERATE AN ERROR MESSAGE AND THE MACRO WILL NOT BE EXECUTED. (*&OPRATOR* EQ **1.ERR10 (*&OPRATOR* EQ **1.ERR10 (*&OPRATOR* EQ **1.ERR10 (*&OPRATOR* EQ *LE*).LESS (*&OPRATOR* EQ *LE*).LESS (*&OPRATOR* EQ *LE*).LESSE0 (*&OPRATOR* EQ *LE*).LESSE0
603 604 605 606 607 608 609 610 611 612 613 614 615 616 617 618 619 620 621 622 623 624	•H •H1 •H2 * * * * * *	AIF LA AGO AIF AIF CR AGO F AIF AIF AIF AIF	11,0 11,=F*&LITB* H2 (*&LTCB* NE **).ERR9 (*&LITB* NE **).ERR9 10,11 .T THE FOLLOWING CODE SEGMENT DETERMINES THE LEAVE CONDITION OPERATOR AND GENERATES THE APPROPRIATE BRANCH INSTRUCTION. A MISSING OR ILLEGAL OPERATOR WILL GENERATE AN ERROR MESSAGE AND THE MACRO WILL NOT BE EXECUTED. (*&OPRATOR* EQ **).ERR10 (*&OPRATOR* EQ **].ERR10 (*&OPRATOR* EQ **].ERR10 (*&OPR
603 604 605 606 607 608 609 610 611 612 613 614 615 616 617 618 619 620 621 622 623 624 625	•H •H1 •H2 * * * * * *	AIF LA AGO AIF CR AGO F AIF AIF AIF AIF AIF	11,0 11,=F*&LITB* H2 (*&LTB* NE **).ERR9 (*&LITB* NE **).ERR9 10,11 .T THE FOLLOWING CODE SEGMENT DETERMINES THE LEAVE CONDITION OPERATOR AND GENERATES THE APPROPRIATE BRANCH INSTRUCTION. A MISSING OR ILLEGAL OPERATOR WILL GENERATE AN ERROR MESSAGE AND THE MACRO WILL NOT BE EXECUTED. (*&OPRATOR* EQ **1.ERR10 (*&OPRATOR* EQ **1.ERR10 (*&OPRA
603         604         605         606         607         608         609         610         611         612         613         614         615         616         617         618         620         621         622         623         624         625         626	•H •H1 •H2 * * * * * *	AIF LA AGO AIF AIF CR AGO F AIF AIF AIF AIF	11,0 11,=F*&LITB* H2 (*&LTCB* NE **).ERR9 (*&LITB* NE **).ERR9 10,11 .T THE FOLLOWING CODE SEGMENT DETERMINES THE LEAVE CONDITION OPERATOR AND GENERATES THE APPROPRIATE BRANCH INSTRUCTION. A MISSING OR ILLEGAL OPERATOR WILL GENERATE AN ERROR MESSAGE AND THE MACRO WILL NOT BE EXECUTED. (*&OPRATOR* EQ **).ERR10 (*&OPRATOR* EQ **].ERR10 (*&OPRATOR* EQ **].ERR10 (*&OPR

3,1´

628	AGO	• END
629 LESS	BNL	AELAB
630	ΔGn	• END
631 •LESSE 632		AGLAB
633 .GRTEQ	A30	• END
634	IBL. Agg	A GL A B
635 .GRT	ВЛН	• END
636	AGO	ACLAB
637 NOTEQ		
638	- AGD	• END
639 *	~~ /	S L'ID
640 *		
641 *		THE NEXT SET OF INSTRUCTIONS RESTORES PEGISTERS
642 *		10 AND 11 AFTER THE LEAVE CONDITION HAS BEEN TESTED.
643 *		IF THE CONDITION IS TRUE, A BRANCH TO "LABEL" WILL
644 *	. `	OCCUR: OTHERWISE, THE PROGRAM WILL CONTINUE SE-
645 *		QUENTIAL EXECUTION.
646 *		
647 *	•	
648 .END	L	10, SAVE10
649 650	L	11, SAVE11
651 A&LAB	8. i.	AGLABEL
. 652	د. و_	10, SAVEIO
653	AGO	11,SAVE11 • STOP
654 *	205	¢ J + Ur
655 *		
656 ¥		THIS LAST SECTION OF CODE IS THE SET OF ERROR MES-
657 ×		SAGES ANY ONE OF WHICH MAY BE GENERATED IF THE
658 *		MACFO'S SYMBOLIC PARAMETERS HAVE BEEN INCORRECTLY
659 *		SPECIFIED» WHEN AN ERROR MESSAGE IS PRINTED. THE
660 *	•	MACRO TERMINATES (EXCEPT IN THE CASE OF AM SERPA
661 *		MESSAGE), NEVER GENERATING A PROGRAM BRANCH.
662 *		· · · ·
663 * . 664 .ERR1	MNOTE	
665	MNOTE	*, ** REGATE OF LEAVE CONDITION IS NOT A SELF-*
666	B	*, DEFINING NUMERIC A
667	AGO .	• END
668 .ERR2	MNOTE	*****REGA** OF LEAVE CONDITION DOES NOT SPECIFY*
669	MNOTE	*, A VALID REGISTER NUMBER
670	8	AGLAB
671	AGO	• END
672 • ERR3	MNOTE	*, LEFT OPERAND OF LEAVE CONDITION WAS NOT SPECI-
673	MNOTE	*,'FIED*
674 675	B	ASLAB
676 . ERR4	AGO MNOTE	
677	MNDTE	*, TWO LITERALS HAVE BEEN SPECIFIED IN A LEAVE CON-
678	ASO	*, *DITION* •RETUPN
679	AGO	• END
680 .ERR5	STORM	*, TMORE THAN ONE LEFT OPERAND SPECIFIED FOR A LEAVE!
681	MNOTE .	*, CONDITION
682	В	AELAB
683	AGD	• END
684 . ERR6	BTCVM	*, ***REGB** OF LEAVE CONDITION IS NOT A SELF-*
685 686	MNOTE	*, DEFINING NUMERIC
687	B Ag n	A&LAB
688 • ERR7	MNOTE	* END
689	MNOTE	*,***REGB** OF LEAVE CONDITION DOES NOT SPECIFY A* *,*VALID REGISTER NUMBER*
690	B	ACLAB
691	ĂGO -	# END
692 "ERR8	MNOTE	*, NO RIGHT OPERAND HAS BEEN SPECIFIED FOR A LEAVEL
693	MNOTE	*, "CONDITION"

694 695 696 • ERP9 697	8 AGO MNOTE MNOTE	AGLAB #END *,'MORE THAN ONE RIGHT OPERAND SPECIFIED FOR A' *,'LEAVE CONDITION'
698	8	AELAB
699	AGO	• END
700 .ERR10	чуоте	*,***OPRATOR** FOR LEAVE CONDITION HAS NOT BEEN*
701	MNOTE	*, *SPECIFIED*
702	8	AGLAB
703	AGO	• END
704 .ERR11	. MNOTE	*, 'ILLEGAL OPERATOR SPECIFIED FOR A LEAVE CONDITION'
. 705	В	AELAB
706	AGO	• ENO
707 .STOP	MEND	

## Sample Program E Illustrating the LEAVE Macro

STMT	SOURCE	STATEMENT	
962		PRINT	NDGEN
963	FRISBEE	ENTER	12, SAVEAREA
	START	OPEN	(DUMPAREA, OUTPUT)
985		LA	2,0
986		LA	3.0
987	1 ( ) ( ) ( ) ( ) ( ) ( ) ( ) ( ) ( ) (	LA	4,0
938		LA	5,0
989		LA	6,0
990		LA	7,0
991		LA	8.0
992		ΪÅ.	9,0
993		LA	10.0
994		ŪA.	11,0
995		LA.	14+0
996		LA	15,0
997	*	•	
998	*		
999	· · · ·	SNAP -	ID=1,OCB=DUMPAREA,PDATA=REGS
1011		•	· · · · · · ·
1012	* *		
1013		00	LOWNUM=1, BYNUM=1, HGHNUM=100, DOLDOP=A, LABEL=Z
1076			A 4+=F*1*
1077			A 5+=F*1*
1078			LEAVE LABEL=Z,LITA=7,OPRATOR=EQ,REGB=4,COND=F
1091			A 6,=F'1'
1092		-	A 7, ≈F*1.*
1093 1190	*	ENDDO	LABEL=Z
1100			
1102	*	SNAP	
1114	*	SNAP	ID=2, DCB=DUMPAREA, PDATA=REGS
1115			
1116		DO	
1179			LOWNUM=1, BYNUM=1, HGHNUM=100, DOLDOP=X, LABEL=X A 14,=F*1*
1180			
1182			A 15.=F*1*
1183		ENODO	
1190	*		
1191			
1192		SNAP	ID=3,DCB=DUMPAREA,PDATA=REGS
1204	wie in the second s		n
1205	ulat.		
1206		EXIT	

REGS 0-7	000002A0 00000007	800189D8 00000007	00000000 - 00000006	00000000 00000006
REGS 8-15	00000000 50018820	00000000	00000000	00000000
END OF SNAP	50018820	00018824	00000000	00000000

REGS AT ENTRY TO SNAP

ID = 002

REGS 0-7	0000030	90018888	00000000	00000000
	00000000	00000000	00000000	00000000
REGS 8-15	00000000	00000000	00000000	00000000
	50018820	00018824	00000000	00000000
END OF SNAP	·			

REGS AT ENTRY TO SNAP

ID = 001

1211		CLOSE	DUMPAREA
1217	SAVEAREA	00	184(0)
1218		D \$	0F
1219	ZERO	DC	12:01
1228	SAVE8	DS ·	1F
1221	SAVE 9	DS	1F
1222	SAVELO	DS	16
1223	SAVE11	DS	15
1224	POINTER	DC	16+0+
1225	STACK .	DS .	100F
.1226	DUMPAREA	DC B	DDNAME=TEAM, DSORG=PS, RECEM=VBA,
			MACRF=W, BLKSIZE=882, LRECL=125
1277	· · · ·	END	CONTRACTION CONTRACTOR CONTRACTOR
1278			=F * ] •
1279			=F*100*
1280			±₽=4+
1281			=F171
			· •

134

Х

#### REGS AT ENTRY TO SNAP

ID = 003

REGS 0-7	000002A0	A001BAF8	00000000	00000000
	0000007	00000007	0000006	00000006
REGS 8-15	00000000 50018820	00000000 00018824	000000000	00000000

END OF SNAP

## Program E with Expanded Macros

STMT SOURCE STATEMENT

962 FRISBEE	ENTER	12,SAVEAREA
963+FRISBEE	DS	OH CONTRACTOR AND A CONTRACTOR
964+	ENTRY	FRISBEE DECLARE NAME ENTRY
965+	USING	*,12 DECLARE BASE ADDRESSIBILITY.
966+	BALR	15,0 (INITIAL ADDRESSIBILITY).
967+	В	12(,15) BRANCH AROUND ID FIELD
968+	00	AL1(7), CL7'ERISBEE' ID LENGTH AND ID
969+	BC TR	15+0 (RESET INITIAL ADDRESSIBILITY
970+	BCTR	15.0 ABSOLUTE ENTRY POINTS.
971+	STM	14,12,12(13) SAVE REGISTERS
972+	L٩	12,15 SETUP BASE REGISTER.
973+	ST	13+SAVEAPEA+4 CHAIN BACK
974+	LA	Q, SAVFAREA CHAIN FORWARD
975+	ST	0,8(0,13)
976+	LR	13,0 SET UP SAVE AREA POINTER
977+	USING	SAVEAREA, 13 AND ADDRESSABILITY
978 START	OP EN	(OUMPAPEA, OUTPUT)
979+	CAOb	0,4 ALIGN LIST TO FULLWORD
980+START	BAL	1,*+8 LOAD PEGI W/LIST ADDR.
981+	DC	AL1(143) OPTION BYTE
982+	DC	AL3(DUMPAREA) DCB ADDRESS
983+ 984	SVC	19 ISSUE OPEN SVC
985	LA	2,0
986	LA	3,0
987	LA	4,0
988	LA	5+0
989	LA	6,0
990	LA LA	7,0
991	LA LA	8+0
992	LA LA	9,0 10,0
993	LA	11,0
994	LA	14.0
995	LA	15,0
996 *		
997 *		
998	SVAP	ID=1, OCB=OUMPAREA, PDATA=REGS
999+	CNOP	0.4
1000+	BAL	1. THBOOOS BRANCH AROUND PARAM LIST
1001+	00	ALI(1) ID NUMBER
1002+	DC	AL1(0)
1003+	ÐC	ALI(130) OPTION FLAGS
1004+	DC	AL1(32) OPTION FLAGS
1005+	DC	A(DUMPAREA) DCB ADDRESS
1006+	00	A(O) TCB ADDRESS
1007+	DC	A(O) ADDRESS OF SNAP-SHOT LIST

1008+IHB0003	DS	0.14	
1009+	SVC	51	
1010 *			
1011 *			
1012	00	LOWNUM=1,BYNUM=1,HGHNUM=100,DOLDOP=A,LABEL=Z	
1013+	ST	8, SAVE8	
1014+	ST	9, SAVE9	
1015+	ST	10, SAVE10	
1016+	\$T	11.SAVE11	
1017+	LA	10,0	
1018+	Δ	10,=F*1*	
1019+	LA	11,0	
1020+	A	11,=F'100'	
1021+	LA	9,0	
1022+	A	9,=F*1*	
1023+	L	8,POINTER	
1024+*			
1025+	ST	9, STACK(8)	
1026+	LA	8+4(8)	
1027+	ST	10, STACK(8)	
1028+	LA	8,4(8)	
1029+	ST	11, STACK(8)	
1030+	LA	8,4(3)	
1031+	ST	8; POINTER	
1032+	L	8, SAVE8	
1033+	L	9, SAVE9	
1034+ 1035+	L L	10,SAVE10 11,SAVE11	
1036+2	ST	8, SAVE8	
1037+	ST	9+SAVE9	
1038+	ST	10, \$47210	
1039+	ST	11.SAVE11	
1040+	L	8, POINTER	
1041+	ŝ	8, == 7 4	
1042+	ĩ	11, STACK(8)	
1043+	ŝ	8, ====================================	
1044+	Ē.	10, STACK(8)	
1045+	S	8 + = F ! 4 !	
1046+	L	9, STACK (8)	
1047+	ST	8,POINTER	
1048+	С	9,ZERO	
1049+	81	X4 · · ·	
1050+	CR	10,11	
1051+	BH	Y4	
1052+	8	24	
1053+X4	CR	10,11	
1054+	BL	Y4	
1055+24	42	10,9	
1056+	L	8, POINTER	
1057+	ST	9, STACK (8)	
1058+	LA	8,4(8)	
1059+ 1060+	ST LA	10,ST4CK(8)	
1061+	ST	11:STACK[8]	
1062+	LA	8:4(8)	
1063+	ST	8, POINTER	
1064+	L	8, SAVE8	
1065+	ĩ	9+S4VE9	
1066+	Ľ	13, SAVE10	
1067+	ι.	11, SAVE11	
1058+	8	W4	
1069+74	L	8+SAVE8	
1070+	L	9; SAVE9	
1071+	L	10, SAVE10	
1072+	L	11,SAVEL1	
1073+	8	AZ	

	EQU	*
		$A \qquad 4_{y} = F + 1 $
		A 5,=F*1* LEAVE LABEL=Z.LITA=7.0PRATOR=E0.REGB=4.COND=F
	ST	LEAVE LABEL=Z,LITA=7,OPRATOR=EQ,REGB=4,COND=F 10,SAVE10
	ST	11, SAVE11
	LA	10,0
	A	10,= E*7*
	LR	11,4
	CR	10,11
	BNE	A5
	L ·	10, SAVE10
	L	1L, SAVEI1
	Յ Լ	AZ 10, SAVE10
	և Լ	II, SAVEID
	6	· A 6,=F*1*
		A 7,=F*1*
	ENDDO	LABEL=Z
	8	2
	L	8 * SAVE8
	L	9. SAVE9
	L.	10, SAVE10
	L	11.SAVE11
	EQU	•
	SNAP	ID=2,DCB=DUMPAREA,PDATA=REGS
	CNDP	0,4
	BAL - OC	1, THB0007 BRANCH AROUND PARAM LIST
	DC	ALI(2) ID NUMBER ALI(0)
	DC	ALI(130) OPTION FLAGS
	DC	ALI(32) OPTION FLAGS
	00	A(DUMPARFA) DCB ADDRESS
	DC	A(D) TCB ADDRESS
	ĐC _	A(O) ADDRESS OF SNAP-SHOT LIST
ſ	DS	OH
	SVC	51
	DO	LOWNUM=1,BYNUM=1,HGHNUM=100,DOLDOP=X,LABEL=X
	57	9 541/69

1110+	90	A(O) ADDRESS OF SNAP-SHOT LIST
1111+IHB00C7	DS	OH
1112+	SVC	51
1113 *		
1114 *		
1115	DO	L JWNUM=1, BYNUM=1, HGHNUM=100, DOLDOP=X,LA
1116+	ST	8 SAVES
1117+	ST	9, SAVE9
1118 +	ST	10, SAVE10
1119+	ST	11, SAVE11
1120+	LA	10,0
1121+	Ā	10, =F*1*
1122+	LA	11.0
1123+	A	11,=F*100*
1124+	LA	9,0
1125+	A ·	9.=F414
1126+	Ĺ	8. POINTER
1127+*		
1128+	ST	9.STACK(8)
1129+	LA	8,4(8)
1130+	ST	10, STACK(8)
1131+	LA	8,4(8)
1132+	ST	11, STACK(8)
1133+	LA	8,4(8)
1134+	S٣	8, POINTER
1135+	L	8, SAVE8
1136+	L.	9, SAVE9
1137+	Ľ	10, SAVE10
1138+	L	11, SAVE 11
1130.4	<u> </u>	O CAVER

8.SAVE8

9. SAVE9

ST

S٢

1139+X 1140+

.

1074+84 1075 1076 1077 1078 +1079+ 1080+ 1081+ 1082+ 1083+ 1084+ 1085+ 1086+ 1087+ 1088+A5 1089+ 1090 1091 1092 1093+ 1094+CZ 1095+ 1096+8Z 1097+ 1098+AZ 1099 × 1100 \* 1101 1102+ 1103+ 1104+ 1105+ 1106+ 1107+ 1108+ 1109+

137

.

.

	~ ~	
1141+	ST	10, SAVE10
1142+	ST	11,SAVEL1
11434	L	8,POINTEP
1144+	S -	8,=F+4+
11454	L	LI, STACK(B)
1146+	S	8,=F+4+
1147+	L	10, STACK(8)
1148+	S	₿ <sub>₱</sub> ≠₣₽₹4₽
1149+	L	9+STACK [3]
1150*	ST	8, POINTER
1151+ .	C	9, ZERO
1152*	BL	X8
1153+	CR	10+11
1154+	-	
	BH	Y8
1155+	8	28
1156+X8	CR	10,11
1157+	BL	Y8
1158+Z8	AR	10,9
1159+	L	8, POINTER
1160+	ŠΤ	9, STACK (8)
1161+	LA	
		8,4(8)
1162+	ST	10, STACK(8)
1163+	LA	8,4(8)
1164*	ST	11, STACK(8)
1165+	L.A	8,4(8)
1166+	ST	8 POINTER
1167+	L	8,SAVE8
1168+	Ĺ	9. SAVE9
1169+	Ļ	10, SAVE10
1170+	Ł	11, SAVE11
1171+	ß	¥8
1172+Y8	L	8 + SAVE8
1173+	٤	9,SAVE9
1174+	ι	10, SAVE10
1175+	Ĺ	11,SAVE11
11.76+	B	ΛX
1177+48	EQU	*
1178		$\Delta \qquad \underline{14}_{9} = F + \underline{1}_{9}$
1179	_	LEAVE LABEL=X
1180+	8	AX
1181		A 15,=F*1*
1182	ENDDO	LABEL≖X
1183*	в	X
1184+CX	Ĺ	8,SAVE8
1185+	Ĺ	
1186+BX		9, SAVE9
	Ĺ	10,SAVE10
11874	L	LL+SAVELL
1188+AX	EQU	*
1189 *		Υ.
1190 *		
1191	SNAP	ID=3, DCB=DUMPAREA, PDATA=REGS
1192+	CNOP	0,4
1193+	BAL	
	_	1, THBOOIL BRANCH AROUND PARAM LIST
1194+	00	AL1(3) ID NUMBER
1195+	90	AL1(0)
I196+	DC	ALI(130) OPTION FLAGS
1197+	00	ALL(32) OPTION FLAGS
1198+	DC	ALDUMPAREAL DCB ADDRESS
1199+	00	A(9) TCB ADDRESS
1200+	00	ALOJ ADDRESS OF SNAP-SHOT LIST
1201+IH80011	0S	OH
1202+	SVC	51
1203 *	J¥6.	2 k
1204 *		
1205	EXIT	
1206+	L	13,4(,13) POP UP SAVE AREA

1207+ 1208+ 1209+ 1210 1211+ 1212+ 1213+ 1214+ 1215+	LM MVI BR CLOSE CNOP BAL DC DC SVC	14,12,12(13) RESTORE REGISTERS 12(13),X*FF* FLAG EXIT 14 RETURN DUMPAREA 0,4 ALTON LIST TO FULLWORD 1,**8 LOAD REG1 W/LIST ADDR AL1(128) OPTION BYTE AL3(DUMPAREA) DCB ADDRESS 20 ISSUE CLOSE SVC
1216 SAVEAREA 1217 1218 ZERD 1219 SAVE8 1220 SAVE9 1221 SAVE10 1222 SAVE11 1223 POINTER 1224 STACK 1225 DUMPAREA	0S 0C 0S 0S 0S 0S 0C 0S	134(0) OF 1F*0* 1F 1F 1F*0* 100F DDNAME=TEAM,DSORG=PS,RECFM=VBA, X MACRF=W,BLKSIZE=882,LRECL=125
1227+*		DATA CONTROL-BLOCK
1228** 1229+DUMPAREA	DC	OFTOT ERIGIN ON WORD BOUNDARY
1231+*		DIRECT ACCESS DEVICE INTERFACE
1233+	DC .	BL16'0' FDAD, DVTBL
1234+	DC	A(O) KEYLE, DEVT, TRBAL
1236+*		COMMON ACCESS NETHOD INTERFACE
1238+ 1239+ 1240+ 1241+ 1242+	DC DC DC DC DC	AL1(0) BUFNO AL3(1) BUFCB AL2(0) BUFL BL2*010000000000000 DSDRG A(1) IOBAD
1244+*		FOUNDATION EXTENSION
1246+ 1247+ 1248+ 1249+	DC DC DC DC	BL1'00000000' BFTEK, BFLN, HIARCHY AL3(1) EODAD BL1'01010100' RECFM AL3(0) EXLST
1251+*		FOUNDATION BLOCK
1253+ . 1254+ 1255+ 1256+	DC DC DC DC	CLS'TEAM: DDNAME BL1'00000010' DFLGS BL1'000000000' TFLG BL2'000000000000000' MACR
1258+*		BSAM-BPAM-QSAM INTERFACE
1260+ 1261+ 1262+ 1263+ 1264+ 1265+ 1266+ 1267+ 1268+	DC DC DC DC DC DC DC DC DC DC	BL1'00000000' RER1 AL3(1) CHECK, GERR, PERR A(1) SYNAO H'0' GIND1, CIND2 AL2(882) BLKSIZE F'0' WCPO, WCPL, OFFSR, DFFSW A(1) IO3A AL1(0) NCP AL3(1) EOBP, EOBAD
1270+*		BSAM-BPAM INTERFACE

1272+	DC	A(1) EOBW
1273+	00	H.O. DIRCT
1274+	00	AL2(125) LRECL
1275+	DC	A(1) CNTRL, NOTE, POINT
1276	END	
1277		= <b>F</b> * <u>1</u> *
1278		=F*100*
1279		=F*4*
1280		=F+7+

#### Sample Program F Illustrating a LEAVE Macro Coded Incorrectly

STMT SOURCE STATEMENT

962 FRISBEE	ENTER		
963+FRISBEE	ÐS	он	
964+	ENTRY	FRISBEE DECLARE NAME ENTRY	
965+	USING	*,12 DECLARE BASE ADDRESSIBILITY.	
966+	BALR	15,0 (INITIAL ADDRESSIBILITY).	
967+	B	12(,15) BPANCH AROUND ID FYELD	
968+	00	AL1(7),CL7'FRISBEE' ID LENGTH AND ID	
969+	BCTR	15,0 (PESET TNITFAL ADDRESSIBILITY,	
970+	80 TR	15,0 ABSOLUTE ENTRY POINT).	
971+	STM	14,12,12(13) SAVE REGISTERS	
972+	L٩	12,15 SETUP BASE PEGISTER.	and the second
973+	ST .	13, SAVEAPEA+4 CHATN BACK	
974+	LA	O, SAVEAREA CHAIN FURWARD	
975+	ST	0,8(0,13)	
976+	LR	13,0 SET UP SAVE AREA POINTER	
977+	USING	SAVEAREA, 13 AND ADDRESSABILITY	
978 START	OPEN	(DUMPAREA, OUTPUT)	
979+	CNOP	0,4 ALIGN LIST TO FULLHORD	
980+START	BAL	1,*+8 LOAD REGI W/LIST ADDR.	
981+	DC	ALI(143) OPTION BYTE	
982+	DC .	AL3(DUMPAREA) DCB ADDRESS	
983+	SVC	19 ISSUE OPEN SVC	
984	LA	2,0	
985	τ.A	3,0	
986	LA	4,0	
987	Ľ٩	5,0	
988	LA	6,0	
989	LA	7,0	
990	LA	8,0	
991	L.A	9,0	
992	LA	10,0	
993	LA .	11,0	
994	LA	14+0	
995	LA	15,0	
996 *			
997 *		· · · · ·	
998	St	NAP JO=1, DCB=DUMPAREA, PDATA=REGS	
999+	CNOP	0.4	
1000+	BAL	1, JHBOOOB BRANCH AROUND PARAM LIST	
1001+	00	ALI(1) ID NUMBER	
1002+	30	AL1(0)	
1003+	DC	ALI(130) OPTION FLAGS	
1004+	00	ALI(32) OPTION FLAGS	
1005+	00	A (DUMPAREA) DCB ADDRESS	
1006+	DC	A(O) TCB ADDRESS	
1007+	00	A(0) ADDRESS OF SNAP-SHOT LIST	
1008+1H80003	0S -	011	

1009+	SVÇ	51	
1010 <b>*</b>			
1011 *			
1012	00	LOWNUM=1, BYNUM=1, HGHNUM=100, DOLODP=A, LABEL=Z	
1013+	ST	8, SAVE8	
1014+	ST	9 * SAVE9	
1015+	ST	10.SAVE10	
1016+	Sĩ		
1017+		11, SAVE11	
	LA	10,0	
1018+	A	10,=F*1*	
1019+	LA	11,0	
1020+	A	11,=F*100*	· · · · · · · · ·
1051+	LA	9,0	
1022+	A	9,=Ftlt	e de la companya de l
1023+	L	8,POINTER	
1024+	ST	9, STACK (8)	
1025+	LA	8,4(8)	
1026+	ST	10, STACK(8)	
1027+	LA	8,4(8)	
1028+	ST		
		11,ST4CK(8)	
1029+	LA	8,4(8)	
1030+	ST	8, POINTER	
1031+	Ļ	8, SAVE8	
1032+	L	9+SAVE9	
1033+	L	10, SAVE 10	
1034+	L.	11, SAVE11 ·	
1035+2	ST	8, SAVE8	
1036+	ST	9, SAVE9	
1037+	ST	10, SAVE10	
1038+	ST -	11, SAVE11	
1039+	L	8+POINTER	
1040+	S		
1041+		8, =F141	
	L	11, STACK(8)	
1042+	S	8,=F141	
1043+	L	10, STACK(8)	
1044+	S	8,=F*4*	
1045+	L.	9,STACK(8)	
1046+	ST	8,POINTER	
1047+	С	9, ZERO	
1048+	BL	Χ4	
1049+	CR	10,11	
1050+	BH	Y4	
1051+	В	Z4	
1052+X4	CR	10,11	
1053+	81	Y4	
1054+24	A R		
1055+			
1055+	L cr	8, POINTED	
	ST	9, STACK ( 8)	
1057+	LA	8,4(8)	
1059+	ST	10,STACK(8)	
1059+	LA	8,4(8)	
1060+	ST	11,STACK(8)	
1061+	LA	8,4(8)	
1062+	ST	8 POINTER	
1063+	L	8,SAVE8	
1064+	Ē	9, SAV 59	
1065+	Ľ.	10, SAVE10	
1066+	Ĺ	11, SAVELD	
1067+	8	W4	
1068+Y4	o L	8+ SAVE8	
10694			
1070+	L	9, SAVE9	
1071+	Ĺ	10, SAVE10	
	L	11,SAVE11	
1072+	B	AZ	•
1073+W4	EQU	*	
1074		A 4,=F°∑°	

1076 LEAVE LABEL=Z,REGA=-5,OPRATOR=EQ,REGB=4,COND=V 1077+ ST 10, SAVE10 1078+ ST. 11,SAVE11 1079 \*, \*REGA\* OF LEAVE CONDITION IS NOT A SELF-1080 \*, DEFINING NUMERIC 1031+ 8 45 1082 +10+SAVE10 L 1083+ L 11, SAVE11 1084+-8 ΑZ 1085+A5 L 10,SAVE10 1086 +٤ 11, SAVE11 1087 6,=F!1! Α 7.=F\*1\* 1088 A. 1089 ENDDO £ABEL=Z 1090+ B Ζ 1091+CZ L 8,SAVE8 1092+ L 9, SAVE9 1093+8Z L 10, SAVE10 1094+ 11, SAVE11 ١. 1095+AZ EQU 1 1096 \* 1097 \* 1098 SNAP I D=2, DCB=DUMPAREA, PDATA=REGS 1099+ CNOP 0,4 1100+ 1. THBOOOT BRANCH AROUND PARAM LIST BAL 1101+ DC AL1(2) ID NUMBER -1102+ DC AL1(0) 1103+ ALI(130) OPTION FLAGS DC 1104+ AL1(32) OPTION FLAGS DC 1105+ 00 A(DUMPAREA) DCB ADDRESS A(O) TCB ADDPESS 1106+ DC 1107 +00 A(0) ADDPESS OF SNAP-SHOT LIST 1108+1HB0007 0Н **DS** 1109+ SVC 51 1110 \* 1111 \* 1112 EXIT 1113+ 13,4(,13) POP UP SAVE AREA L 1114+ LM 14,12,12(13) RESTORE REGISTERS 1115+ MVT 12(13),X'FF' FLAG EXIT 1116+ BR. 14 RETURN CLOSE 1117 DUMPAREA 1118+ CNOP 0,4' ALIGN LIST TO FULLWORD 1119+ BAL 1,\*+8 LOAD REG1 W/LIST ADDR 1120+ 00 AU1(128) OPTION BYTE 1121+ DC ALB(DUMPAREA) DCB ADDRESS 1122+ SVC 23 ISSUE CLOSE SVC 1123 SAVEAREA DC 18A(0) 1124 DS 0F 1125 ZERO DC 1F\*0\* 1126 SAVE8 DS 1F 1127 SAVE9 IF DS 1128 SAVE10 DS 1 F 1 F 1129 SAVELL DS DC 16101 1130 POINTER 1131 STACK D S 100F 1132 DUMPAREA DCB DDNAME=TEAM, DSDRG=PS, RECFM=VBA. MACRF=W,BLKSIZE=882,LRECL=125 1134+\* DATA CONTROL BLOCK 1135+\* 1136+DUMPAREA DC OFTOT ORIGIN ON WORD BOUNDARY 1138+\* DIRECT ACCESS DEVICE INTERFACE

5,=F+1

A

1075

142

X

1140+ 1141+	DC DC-	BL16'J' FDAD, DVTBL A(0) KEYLE, DEVT, TRBAL
1143**		COMMON ACCESS METHOD INTERFACE
1145+ 1146+ 1147+ 1148+ 1149+	0C DC DC DC DC	AL1(0) BUFNO AL3(1) BUFCB AL2(0) BUFL BL2*010000000000000 DSDRG A(1) TOBAD
1151+*		FOUNDATION EXTENSION
1153+ 1154+ 1155+ 1155+	DC DC DC DC	BL1°0000000° BFTEK,BFLN,HIARCHY AL3(1) EDDAD BL1°01010100° RECFM AL3(0) EXLST
1158+*		FOUNDATION BLOCK
1160+ 1161+ 1162+ 1163+	DC DC DC DC	CL8*TEAM* DDNAME BL1*00000010* OFLGS BL1*00000000* IFLG BL2*00000000000000* MACR
1165+*	~	BSAM-BPAM-QSAM INTERFACE
1167+ 1168+ 1169+ 1170+ 1171+ 1172+ 1173+ 1173+ 1175+	DC DC DC DC DC DC DC DC DC	BL1'00000000' PERI AL3(1) CHECK, GERR, PERR A(1) SYNAD H'O' CIND1, CIND2 AL2(882) BLKSIZE F'O' WCPD, WCPL, OFFSR, OFFSW A(1) IOBA AL1(0) NCP AL3(1) EDBR, EDBAD
1177+*		BSAM-BPAM INTERFACE
1179+ 1180+ 1181+ 1182+ 1183 1184 1185 1186	DC DC DC DC DC END	A(1) EOBW H'O' DIRCT AL2(125) ERECL A(1) CNTPL, NOTE, POINT =F'1' =F'100' =F'4'

Sample Program G Illustrating a LEAVE Macro Coded Incorrectly

#### STMT SOURCE STATEMENT

962 FRISBEE	ENTER	12, SAVEAREA ·
963+FRISBEE	DS	08
964+	EVTRY	FRISBEE DECLARE NAME ENTRY
965+		*,12 DECLARE BASE ADDRESSIBILITY.
966+	BALR	15.0 (INITIAL ADDRESSIBILITY).
967+		12(,15) BPANCH AROUND ID FTELD
968+	DC .	AL1(7), CL7'FRISBEE' ID LENGTH AND ID
969+	BETR	15,0 (RESET INITIAL ADDRESSIBILITY
970*	8C T 9-	15.0 ABSOLUTE ENTRY POINT).
971+	STM	14,12,12(13) SAVE REGISTERS

972+	L٩	12,15 SETUP BASE REGISTER.
973+	ST	13, SAVEAPEA+4 CHAIN BACK
974+	LA	O, SAVEAPEA CHAIN FORWARD
975+	ST	0,8(0,13)
976*	ΓR.	13.0 SET UP SAVE AREA POINTER
977+	USING	SAVEARE4,13 AND ADDRESSABILITY
978 START	OP EN	(DUMPAREA, OUTPUT)
979+	CNOP	0,4 ALIGN LIST TO FULLADRO
980+START	BAL	1,*+8 LOAD REGI W/LIST ADDR.
981+	ĎC	ALL(143) OPTION BYTE
982+	DC	AL3(OUMPAPEA) DCB ADDRESS
983+	SVC	19 ISSUE OPEN SVC
984	LA	2,0
985	LA	3,0
986	LA	4,0
987	LA	5,0
988	LA	6,0
989	LA	7,0
990	LA	8.0
991	LA	9.0
992	LA	10,0
993	LA	
994	_	, 11,0
	LA	14,0
995	L.A	15,0
996 *		
997 *		
998	S	NAP ID=1,DCB=DUMPAREA,PDATA=REGS
999+	CNOP	0,4
1000+	BAL	1, THB9003 BRANCH AROUND PARAM LIST
1001+	00	ALI(1) ID NUMBER
1002+	DC	AL1(0)
1003+	DC	ALI(130) OPTION FLAGS
1004+	DC	ALI(132) OPTION FLAGS
1005+	DC DC	A(DUMPAREA) DCB ADDRESS
1006+	DC	
1007+		A(O) TCB ADDRESS
	00	A(O) ADDRESS OF SNAP-SHOT LIST
1008+1H80003	DS	OH
1009+	SVC	51
1010 *		
1011 *		
1012	03	L DWNUM=1, BYNUM=1, HGHNUM=100, DDLDOP=A, LABEL=Z
1013+	ST	8, SAVE8
1014+	ST	9 • SAVE9
1015+	ST	10, SAVE10
1016+	ST	11, SAVE11
1017+	L.A	12,0
1018+	4	10,=#*1'
1019+	L.A	11,0
1020+	A	11, = F'100'
1021+	LA	
1022+		9,0
	Α.	9,=F11*
1023+	L	8, POINTER
1024+	ST	9, STACK (8)
1025+	LA	8,4(8)
1026+	ST	10,STACK(8)
1027+	LA	8+4(9)
1028+	ST	11,STACK(8)
1029+	LA	8,4(3)
1020+	ST	8 POINTER
1031+	L	8,54758
1032+	Ĺ.	9, 51/69
1033+	Ĺ	10, SAVE10
1034+	Ū.	11,SAVELL ·
1035+Z	ST	8,SAVES
1036+	ST.	9, 547 59
1037+	ST	10, SAVE10
		and a second

1038+	ST	11, SAVE11
1039+	Ľ	8, POINTER
1040+	ŝ	8,=====
1041+	Ĺ	11, STACK(8)
1042+	S	8,=F*4*
1043+	L	10, STACK(8)
1044+	S	8,=F*4*
1045+	L	9, STACK(8)
1046+	ST	8, POINTEP
1047+	С.	9+ZER0
1048*	BL	X4
1049+	CR	10,11
1050+	RH	Y4
1051+	В	24
1052+X4	Č٦	10,11
1053+	BC	Y4
1054+24	AR .	10,9
1055+	L	8, POINTER
1056+	ST	9, ST4CK (8)
1057+	LA	
1058+	ST .	8,4(8) 10,STACK(8)
1059+	LA	8,4(8)
1060+	ST	11,STACK(8)
1061+	1.4	8,4(8)
1062+	ST	8, POINTER
1063+	L.	8, SAVE8
1064+	Ļ	9, SAVE9
1065+	L	10, SAVE10
1066+	L	11, SAVE11
1067+	8	N4
1068+Y4	L	8,SAVE8
1069+	L	9,SAVE9
1070+	L	10,S4VE10
10714	L	11,SAVE11
1072+	В	AZ
1073+144	EQU	*
1074		A 4,=F*1 (
1075		A 5,=F*1*
1076	<u> </u>	LEAVE LABEL=Z, DPRATOR=EQ, REGB=4, COND=K.
1077+	ST	10, SAVE10
1078+	ST	11, SAVE11
1079		*, LEFT OPERAND OF LEAVE CONDITION WAS NOT SPECI-
1080	_	*,FTED
1081+	8	45
1082+	L	10, SAVE10
1083+	t_	11,SAVE11
1084+	B	ΛZ
1085+A5	L	10, SAVE10
1086+	L	11,SAVE11
1087		A 6,=F <sup>t</sup> 1 <sup>t</sup>
1088		A 7,=F*1*
1089	ENDDO	
1090+	8	2
1091+CZ	L	8, SAVE8
1092+	L	9, SAVE9
1093+8Z	L	10,SAVE10
1094+	L	11, SAVE11
1095+AZ	EQU	*
1096 *		
1097 *	•	
1098		NAP I D=2+DCB=DUMPAREA+PDATA=REGS
1099+	CNOP	0,4
1100+	BAL	1, THB0007 BPANCH AROUND PARAM LIST
1101+	D¢	ALI(2) ID NUMBER
1102+	DC	ALI(0)
1103+	00	ALI(130) OPTION FLAGS

1104+ 1105+ 1106+ 1107+ 1108+IHB0007 1109+ 1110 *	DC DC DC DC DS SVC	ALI(32) DPTION FLAGS A(DUMPAREA) DCB ADDRESS A(0) TCB ADDRESS A(0) ADDRESS OF SNAP-SHOT LIST OH 51	
	EXIT L MVI BR CLOSE CNOP BAL DC DC SVC	0,4 ALIGN LIST TO FULLWORD 1,*+8 LOAD REGI W/LIST ADDR	
1123 SAVEAREA 1124 1125 ZERO 1126 SAVE8 1127 SAVE9 1128 SAVE10 1129 SAVE10 1129 SAVE11 1130 POINTER 1131 STACK 1132 DUMPAREA	0 S DC DS DS DS DS DC DS	134(0) 0F 1F*0* 1F 1F 1F 1F 1F 1F 1OOF DDN AME=TEAM, DSORG=PS, RECFM=VBA, MACRF=W, BLKSIZE=882, LRECL=125	x
1134+*		DATA CONTROL BLOCK	
1134+* 1135** 1136+DUMPAREA	DC	OFTOT ORIGIN ON WORD BOUNDARY	
1135**	DC		
1135** 1136+DUMPAREA 1138+* 1140+	DC	OF:0: DRIGIN ON WORD BOUNDARY DIRECT ACCESS DEVICE INTERFACE BL16:0: FDAD, DVTBL	
1135+* 1136+DUMPAREA 1138+*		OFTOT DRIGIN ON WORD BOUNDARY DIRECT ACCESS DEVICE INTERFACE	
1135** 1136+DUMPAREA 1138+* 1140+	DC	OF:0: DRIGIN ON WORD BOUNDARY DIRECT ACCESS DEVICE INTERFACE BL16:0: FDAD, DVTBL	
1135** 1136+DUMPAREA 1138+* 1140+ 1141+ 1143+* 1145+	DC DC	OF:0' DRIGIN ON WORD BOUNDARY DIRECT ACCESS DEVICE INTERFACE BL16'O' FDAD, DVTBL A(O) KEYLE, DEVT, TRBAL COMMON ACCESS METHOD INTERFACE AL1(O) BUENO	
1135** 1136+DUMPAREA 1138+* 1140+ 1141+ 1143+* 1145+ 1145+ 1146+ 1146+ 1147+	DC DC DC DC DC	OF:0' DRIGIN ON WORD BOUNDARY DIRECT ACCESS DEVICE INTERFACE BL16'O' FDAD, DVTBL A(O) KEYLE, DEVT, TRBAL COMMON ACCESS METHOD INTERFACE AL1(O) BUFNO AL3(I) BUFCB AL2(O) BUFL	
1135** 1136+DUMPAREA 1138+* 1140+ 1141+ 1143+* 1145+ 1145+ 1146+	DC DC DC	OF:0: DRIGIN ON WORD BOUNDARY DIRECT ACCESS DEVICE INTERFACE BL16:0: FDAD, DVTBL A(0) KEYLE, DEVT, TRBAL COMMON ACCESS METHOD INTERFACE AL1(0) BUENO AL3(1) BUFCB	
1135** 1136+DUMPAREA 1138+* 1140+ 1141+ 1143+* 1145+ 1145+ 1146+ 1147+ 1148+	DC DC DC DC DC DC DC	OF:0' DRIGIN ON WORD BOUNDARY DIRECT ACCESS DEVICE INTERFACE BL16'O' FDAD, DVTBL A(O) KEYLE, DEVT, TRBAL COMMON ACCESS METHOD INTERFACE AL1(O) BUENO AL3(1) BUENO AL3(1) BUENO AL3(1) BUENO AL3(1) BUENO AL3(1) BUENO AL3(1) BUENO	
1135+* 1136+DUMPAREA 1138+* 1140+ 1141+ 1143+* 1145+ 1145+ 1146+ 1146+ 1148+ 1148+ 1149+ 1151+*	DC DC DC DC DC DC DC	OF:O: ORIGIN ON WORD BOUNDARY DIRECT ACCESS DEVICE INTERFACE BL16:O: FDAD, DVTBL A(O) KEYLE, DEVT, TRBAL COMMON ACCESS METHOD INTERFACE AL1(O) BUFND AL3(I) BUFND AL3(I) BUFCB AL2(O) BUFL BL2:0100000000000000000000000000000000000	
1135+* 1136+DUMPAREA 1138+* 1140+ 1141+ 1143+* 1145+ 1145+ 1146+ 1146+ 1147+ 1148+ 1148+ 1151+* 1153+ 1153+ 1154+	DC DC DC DC DC DC DC DC	OF:O: ORIGIN ON WORD BOUNDARY DIRECT ACCESS DEVICE INTERFACE BL16:O: FDAD, DVTBL A(O) KEYLE, DEVT, TRBAL COMMON ACCESS METHOD INTERFACE AL1(O) BUFND AL3(I) BUFND AL3(I) BUFCB AL2(O) BUFL BL2:0100000000000000000 DSDRG A(I) IOBAD	
1135+* 1136+DUMPAREA 1138+* 1140+ 1140+ 1141+ 1145+ 1145+ 1145+ 1146+ 1146+ 1146+ 1148+ 1148+ 1151+* 1151+*	DC DC DC DC DC DC DC DC DC	OF*O* ORIGIN ON WORD BOUNDARY DIRECT ACCESS DEVICE INTERFACE BL16*O* FDAD, DVTBL A(0) KEYLE, DEVT, TRBAL COMMON ACCESS METHOD INTERFACE AL1(0) BUFND AL3(1) BUFCB AL2(0) BUFL BL2*O100000000000000* DSDRG A(1) TOBAD FOUNDATION EXTENSION BL1*00000000* BFTEK, BFLN, HIARCHY AL3(1) EDDAD BL1*010100* RECEM	
1135+* 1136+DUMPAREA 1138+* 1140+ 1141+ 1143+* 1145+ 1145+ 1145+ 1146+ 1146+ 1148+ 1148+ 1151+* 1151+* 1155+ 1156+	DC DC DC DC DC DC DC DC	OF:O: ORIGIN ON WORD BOUNDARY DIRECT ACCESS DEVICE INTERFACE BL16:O: FDAD, DVTBL A(0) FDAD, DVTBL A(0) KEYLE, DEVT, TRBAL COMMON ACCESS METHOD INTERFACE AL1(0) BUFND AL3(1) BUFCB AL2(0) BUFL BL2:0100000000000000000000000000000000000	· · · · · · · · · · · · · · · · · · ·
1135** 1136+DUMPAREA 1138+* 1140+ 1141+ 1143+* 1145+ 1145+ 1145+ 1146+ 1146+ 1146+ 1148+ 1151+* 1151+* 1155+ 1156+ 1158+*	DC DC DC DC DC DC DC DC DC	OF*O* ORIGIN ON WORD BOUNDARY DIRECT ACCESS DEVICE INTERFACE BL16*O* FDAD, DVTBL A(0)* KEYLE, DEVT, TRBAL COMMON ACCESS METHOD INTERFACE AL1(0) BUFNO AL3(1) BUFNO AL3(1) BUFNO AL3(1) BUFCB AL2(0) BUFL BL2*O1000000000000000000000000000000000000	
1135+* 1136+DUMPAREA 1138+* 1140+ 1141+ 1143+* 1145+ 1145+ 1145+ 1146+ 1148+ 1148+ 1149+ 1151+* 1153+ 1156+ 1158+* 1160+	DC DC DC DC DC DC DC DC DC DC DC	OF*O* ORIGIN ON WORD BOUNDARY DIRECT ACCESS DEVICE INTERFACE BL16*O* FDAD, DVTBL A(0) KEYLE, DEVT, TRBAL COMMON ACCESS METHOD INTERFACE AL1(0) BUFND AL3(1) BUFND AL3(1) BUFCB AL2(0) BUFL BL2*010000000000000000 DSDRG A(1) IOBAD FOUNDATION EXTENSION BL1*00000000* BFTEK, BFLN, HIARCHY AL3(1) EDDAD BL1*010100* RECFM AL3(0) EXLST FOUNDATION BLOCK CL8*TEAM* DDNAME	· · · · · · · · · · · · · · · · · · ·
1135** 1136+DUMPAREA 1138+* 1140+ 1141+ 1143+* 1145+ 1145+ 1145+ 1146+ 1146+ 1146+ 1148+ 1151+* 1151+* 1155+ 1156+ 1158+*	DC DC DC DC DC DC DC DC DC	OF*O* ORIGIN ON WORD BOUNDARY DIRECT ACCESS DEVICE INTERFACE BL16*O* FDAD, DVTBL A(0)* KEYLE, DEVT, TRBAL COMMON ACCESS METHOD INTERFACE AL1(0) BUFNO AL3(1) BUFNO AL3(1) BUFNO AL3(1) BUFCB AL2(0) BUFL BL2*O1000000000000000000000000000000000000	· · · · · · · · · · · · · · · · · · ·
1135+* 1136+DUMPAREA 1138+* 1140+ 1141+ 1143+* 1145+ 1145+ 1145+ 1146+ 1148+ 1148+ 1149+ 1151+* 1153+ 1155+ 1156+ 1158+* 1160+ 1161+	DC DC DC DC DC DC DC DC DC DC DC DC DC	OF*O* ORIGIN ON WORD BOUNDARY DIRECT ACCESS DEVICE INTERFACE BL16*O* FDAD, DVTBL A(0) KEYLE, DEVT, TRBAL COMMON ACCESS METHOD INTERFACE AL1(0) BUFND AL3(1) BUFND AL3(1) BUFCB AL2*O1000000000000000 DSDRG A(1) IOBAD FOUNDATION EXTENSION BL1*0000000* BFTEK, BFLN, HJARCHY AL3(1) EODAD BL1*010100* RECFM AL3(0) EXLST FOUNDATION BLDCK CL8*TEAM* DDNAME BL1*0000010* OFLGS	

1167+	00	BL1*00000000* PER1
1168+	DC	AL3(1) CHECK, GERR, PERR
	00	
1170+	00	H*9* CIMO1, CIND2
1171+	00	ALZ(882) BLKSIZE
1172+	00	F'O' WCPD, WCPL, OFFSR, OFFSW
1173+	DC	A(1) IOBA
1174+	DC	ALI(0) NCP
1175+	DC	AL3(1) EDBR, EOBAD
1177+*		DCALLODAY INTEDCICE
		BSAM-BPAM INTERFACE
	DC	
1179+		A(1) EOBW
1179+	00	A(1) EOBW HFOF DIRCI
1179+ 1180+	00 00	A(1) EOBW H®O® DIRCY AL2(125) LRECL
1179+ 1180+ 1181+ 1182+	00	A(1) EOBW HFOF DIRCI
1179+ 1180+ 1181+ 1182+ 1183	00 00 00	A(1) EDBW HFOF DIRCT AL2(125) LRECL A(1) CNIRL, NOTE, POINT
1179+ 1180+ 1181+	00 00 00	A(1) EOBW H®O® DIRCY AL2(125) LRECL

## Sample Program H Illustrating a LEAVE Macro Coded Incorrectly

#### STMT SOURCE STATEMENT

		•
962 FRISBEE	ENTER	12, SAVEAREA
963+FRISBEE	DS	OH CONTRACTOR CONTRACTOR
964+	ENTRY	FRISBEE DECLARE NAME ENTRY
965+	USING	*,12 DECLARE BASE ADDRESSIBILITY.
966+	BALR	15,0 (INTITAL ADDRESSIBILITY).
967+	8	12(,15) BPANCH AROUND ID FIELD
968+	DC	ALI(7), CL7'FRTSBEE' ID LENGTH AND ID
969+	BCTR	15,0 (RESET INITIAL ADDRESSIBILITY
970+	BCTQ	15.0 ABSOLUTE ENTRY POINTS.
971+	STM	14,12,12(13) SAVE REGISTERS
972+	LR	12,15 SETUP BASE REGISTER.
973+	ST	13, SAVEAREA+4 CHAIN BACK
974+	LA .	0, SAVEAPEA CHAIN FORWARD
975+	ST	0,8(0,13)
976+	LR	13.0 SET UP SAVE AREA POINTER
977+	US ING	SAVEAREA, 13 AND ADDRESSABILITY
978 START	OPEN	I DUMPAREA, OUTPUT
979+	CNOP	0.4 ALIGN LIST TO FULLADED
980+STAR 1	BAL	1:*+8 LOAD "EGI W/LIST ADDR.
981+ -	00	ALI(143) OPTION BYTE
982+	DC	AL3(DUMPAREA) DCB ADDRESS
983+	SVC	19 ISSUE OPEN SVC
984	LA	2+0
985	LA	3,0
986	LA	4.0
987	L.A	5.0
988	LA	6,0
989	LA	7,0
990	LA	8.0
991	ĹΑ	9:0
992	LA	10,0
993	LA	11,0
994	LA	14,0
995	LA	15,0
996 ×		
997 *		
998	SN	AP ID=1+DCB=DUMPAREA+PDATA=REGS
		S TITLE DOWN HILMAN DWINH KEUD

999+	CVOP	0,4
1000+	BAL	I, THBOOD3 BRANCH AROUND PARAM LIST
1001+	DC	ALI(1) ID NUMBER
1002+	00	AL1(0)
1003+	00	AL1(130) OPTION FLAGS
1004+	DC	AL1(32) OPTION FLAGS
1005+	00	A(DUMPAREA) DCB ADDRESS
1006+	DC	A(0) TCB ADDRESS
1007+	DC	A(O) ADDRESS OF SNAP-SHOT LIST
1008+IHB0003	DS	OH
1009+	SVC	51
1010 *	•	
1011 *		
1012	00	LOWNUM=1,BYNUM=1,HGHNUM=100,DOLDOP=A,LABEL=Z
1013+	ST	8, SAVE8
1014+	ST	9, SAVE9
1015+	ST	10, S4VE10
1016+	sr	11, SAVE11
1017+	LA	10,0
1018+	Δ -	10+=======
1019+	LA	11,0
1020+	A	11,=F*190*
1021+	LA	9,0
1022+ 1	A	9,=F11*
1023+	L.	8, POINTER
1024+	ST 🕾	9, STACK(8)
1025+	LA	8,4(8)
1026+	ST .	19, STACK(8)
1027+	LA	8,4(8)
1028+	ST	11,STACK(8)
1029+	LA	8,4(3)
1030+	ST -	8, POINTER
1031+	L	8, S4VE8
1032+	L	9, SAVE9
1033+	L	10, SAVE10
1034+	L -	11, SAVELL ·
1035+Z	ST	8, SAVEB
1036+	ST	9, SAVE9
1037+	ST	10, SAVE10
1038+	ST	11, SAVE11
1039+ 1040+	L S	8, POINTER
1040+	s L	8, =F • 4 •
1042+	S	11,STACK(8) 8,=F'4*
1043+	L	10, STACK(8)
1044*	ŝ	B <sub>9</sub> = F <sup>1</sup> 4 <sup>1</sup>
1045+	Ĺ	9, STACK(8)
1046+	ŝt	8, POINTER
1047+	Ċ	9, ZERD
1048+	BL	X4 ,
1049+	C.R	10,11
1050+	8H	Υ4
1051+	8	Ζ4
1052+X4	CR	10,11
1053+	8L	Υ4
1054+Z4	49	10,9
1055+	L_	8, POINTER
1056+	ST	9+STACK(8)
1057+	LA	8+4(8)
1058+	ST	10, STACK(8)
1059+ 1060+	LA ST	8,4(8) 11. STACK101
1061+	LA	11,STACK(8) 8,4(8)
1062+	ST	8, POINTER
1063+	Ĺ	B, SAVEB
1054+	Ĺ	9 • SAVE9

.

1065+ 10, SAVE10 L 1066+ L 11+SAVE11 1067+ В ₩4 1068+Y4 Ł 8, SAVE8 1069+ L 9+SAVE9 1070+ L 10, SAVE10 1071+ 11, SAVE11 1 1072+ B ΑZ 1073+W4-EQU ×. 1074 . 4 4.=F'11 5,=F\*1\* 1075 Å 1076 LEAVE LABEL=Z, LITA=7, OPRATOR=QQ, REGB=4, COND=T ST 1077+ 10, SAVE10 1078+ ST 11, SAVE11 1079+ LA 10,0 1080+ 10,=F'7' A, 1081+ LR 11,4 1082+ CR 10,11 1083 \*, ILLEGAL OPERATOR SPECIFIED FOR A LEAVE CONDITION 1084+ R Δ5 1085+ L 10, SAVE10 1086+ 11, SAVE11 L 1087 +8 ΑZ 1088+A5 ٤ 10, SAVE10 1089+ . ٤. 11,SAVE11 • A 6,=F\*1\* 1090 7,=F\*1\* 1091 Α LABEL=Z 1092 ENDDO 1093+ 8 Z 1094+CZ 8,SAVE8 Ł 10954 9+SAVE9 L 1096+BZ L 10, SAVE10 1097+ Ł 11, SAVE11 1098+AZ EQU \* 1099 # 1100 \* 1101 SNAP ID=2,DCB=DUMPAREA,PDATA=REGS 1102+ CNOP 0,4 1103+ BAL 1, THBOOOT BRANCH AROUND PARAM LIST 1104+ DC AL1(2) ID NUMBER 1105+ DC AL1(0) 1106+ DC ALI(130) DPTION FLAGS 1107+ DC AL1(32) OPTION FLAGS A (DUMPAREA) DCB ADDRESS 1108+ DÇ 1109+ 00 A()) TCB ADDRESS 1110+ 00 A(O) ADDRESS OF SNAP-SHOT LIST 1111+IH80007 ÐS 0H 1112+ SVC 51 1113 \* 1114 # 1115 EXIT 1116+ L. 13,4(,13) POP UP SAVE AREA 1117 +14,12,12(13) RESTORE REGISTERS LM 1118+ 4V I 12(13), X'FF' FLAG EXIT 1119+ 8R 14 RETURN 1120 CLOSE DUMPAREA 0,4 ALIGN LIST TO FULLWORD 1121+ CNOP 1122+ 54 L 1,\*+8 LOAD REGI W/LIST ADDR ALI(128) OPTION BYTE 1123+ DC. 1124+ 00 AL3(DUMPAREA) DCB ADDRESS 1125+ SVC 20 ISSUE CLOSE SVC 1126 SAVEAREA DC 18A(0) 1127 0\$ 0F 1128 ZERO 16101 DC 1129 SAVE8 05 16 1E 1130 SAVE9 DS 1131 SAVELO DS 16

1133 POINTER D		lF lF*O* 190F DONAME=TEAM,DSORG=PS,RECFM=VBA, CRF=W,BLKSIZE=882,LRECL=125
1137+* 1138+* 1139+DUMPAREA D	DC 0F*	DATA CONTROL BLOCK O' DRIGIN ON WORD BOUNDARY
1141+*		DIRECT ACCESS DEVICE INTERFACE
	-	6'0' FDAD, DVIBL D) KEYLE, DEVT, TRBAL
1146+*		COMMON ACCESS METHOD INTERFACE
1149+ D 1150+ D 1151+ D	DC AL3 DC AL2 DC BL2	L(0) BUFNO S(1) BUFCB 2(0) BUFL 2*010000000000000 DSORG L) IOBAD
1154+*		FOUNDATION EXTENSION
1157+ D 1158+ D	DC ALB DC BL1	L'00000000' BFTEK,BFLN,HIARCHY S(L) E0DAD L'01010100' RECFM S(O) EXLST
1161+*		FOUNDATION BLOCK
1164+ D 1165+ D	C BL1 C BL1	*TEAM* DDNAME *00000010* OFLGS *00000000* IFLG *0000000000100000* MACR
1168+*		BSAM-BPAM-QSAM INTERFACE
1171+     D       1172+     D       1173+     D       1174+     D       1175+     D       1176+     D       1177+     D	DC         AL3           DC         A(1)           DC         H*0           DC         F*0           DC         A(1)           DC         A(2)           DC         F*0           DC         A(1)	*0000000* RERI (1) CHECK, GERR, PERR ) SYNAD * CINDI, CINDZ (882) BLKSIZE * WCPO, WCPL, OFFSR, OFFSW } IOBA (0) NCP (1) EOBR, EOBAD
1180**	· ·	BSAM-BPAM INTERFACE
1183+ D 1184+ D 1185+ D	0C H*C 0C AL2 0C A(1 ND == F*	100* 4*

х

#### APPENDIX C

#### THE CASE, ENDCASE, ELSE, AND ENDELSE MACROS

The Source Listing for the CASE Macro

STMT SOURCE STATEMENT

709 710 711 712	<b>承</b> · · · ·	<b>.</b>		THE CASE MACRO PROVIDES A SELECTIVE CONTROL STPUC- TURE FOR ALC. A CONDITION IS SPECIFIED WHEN CODING THE MACFO, AND DURING PROGRAM EXECUTION, IF THE CONDITION IS TRUE, PROGRAM INSTRUCTIONS FOLLOWING
713 714 715 716	* .			THE CASE STATEMENT (UP TO THE 'ENDCASE' MACRO) WILL BE EXECUTED. IF THE CONDITION IS FALSE, AN IMMED- TATE BRANCH TO THE 'ENDCASE' MACPO WILL OCCUR.
717	*			
718 719		MACRO CASE	61.40	
117		VAJL.	81.00	HEL=, &LITA=, &REGA=, &LOCA=, &OPRATOR=, &LITB=, &REGB=, X B=
720		LCLA		ELAB
721	LLA8	SE TA AG D		CSYSNDX -
723	*	A (5 (2)		• SKIP1
724				
725	*	-		TO TEST THE CASE CONDITION REGISTERS 10 AND 11 ARE
726				USED AS WORK AREAS; CONSEQUENTLY, THE CONTENTS OF
727				THESE REGISTERS ARE INITIALLY SAVED.
728 729				
	SKIPL	ST		10, SAVE10
731	# DIVEL F	ST		11, SAVE11
732		AGO		•SKTP2
733	*			
734	*			
735				THE LEFT OPERAND OF THE CASE CONDITION IS DETER-
736				MINED IN THE FOLLOWING SECTION OF CODE. IT MAY BE
737				SPECIFIED BY A LITERAL (LITA), BY REGISTER (REGA) OP
738 739				BY LOCATION (LOCA). ITS VALUE IS LOADED INTO RE-
740				ISTEP 10. IF THE LEFT OPERAND HAS BEEN ILLEGALLY
741				SPECIFIED, AN ERROR MESSAGE WILL BE OUTPUT, AND THE CASE CONDITION WILL BE ASSUMED FALSE.
742				CASE CONVITENT WILL DE ASSUMED FALSE.
743	*			
744	• SKIP2	AIF		(*6L1TA* EQ **).A
745		AIF		(*&REGA* NE **).ERR1
746		AIF		('&LOCA' NE '').ERR1
747	OFTION	AIF		(*&LITB* NE **).ERR2
749	RETURN	LA		
750		ል ልር ጋ		10,=F'&LTTA' •C
751	• A	ATE		сс (*&РЕСА! ЕQ !!).В
752	/	AIF		("GLOCA" NE "") ERAL
753		ATE		(T'EREGA NE 'N'IGERR3
754		AIF	•	(*EREGA* GT *15*1.ERR4
755		1.2 		10+ GREGA
756		1001		- (

757 .8 758 759 760 * 761 *	AIF L Agd	(*&LOCA* EQ **).ERR5 10,&LOCA .C
762 * 763 * 764 * 765 * 766 * 767 * 768 * 769 *	、 ·	THE NEXT SECTION OF CODE DETERMINES THE RIGHT OPER- AND, AND LOADS IT INTO REGISTER 11. IT MAY BE SPEC- IFIED IN THE SAME MANNER AS THE LEFT OPERAND WITH THE RESPECTIVE PARAMETERS BEING LITB, PEGB AND LOCB. A MISSING OR ILLEGALLY SPECIFIED PARAMETER WILL CAUSE THE GENERATION OF AN ERROR MESSAGE, AND THE CASE CONDITION WILL BE ASSUMED FALSE.
770 * 771 •C 772 773 774 775	AIF AIF LA A	('&LITB' EQ '').D ('&REGB' NE '').ERR6 ('&LDCB' NE '').ERR6 11.0 11.=F'&LITB'
776 777 D 778 779 780 781 782	AGD AIF AIF AIF AIF LR AGD	• F (*&REGB* EQ **)•E (*&L^CB* NE **)•ERR6 (T*&REGB NE *N*)•ERR7 (*&REGB* GT *15*)•ERR8 11,&PEG8 • F
783 .€ 783 .€ 785 786 * 786 * 787 * 788 *	ATF L AGD	('&LOCB' EQ ''} ERR6 11,&LOCB •F THE NEXT SECTION OF CODE COMPARES THE LEFT AND RIGHT
789 * 790 * 791 * 792 * 793 * 794 *		OPERANDS AND GENERATES AN APPROPRIATE BRANCH IN- STRUCTION BASED ON THE VALUE OF THE SPECIFIED LOG- ICAL OPERATOR. IF THE OPERATOR IS ILLEGAL OR MISS- ING FROM THE PARAMETER LIST, AN EPROR MESSAGE WILL BE DUTPUT, AND THE CASE CONDITION WILL BE ASSUMED FALSE.
795 * 796 * 797 •F 798 799 800	CP AIF AIF AIF	10,11 ('67PRATOR' EQ '').ERR9 ('60PRATOR' EQ 'LT').LESS
801 802 803 804 805	AIF AIF AIF AIF A30	(*&OPRATOR* EQ *LE*).LESSEQ (*&OPRATOR* EQ *EQ*).EQUAL (*&OPRATOR* EQ *GE*).GRTEQ (*&OPRATOR* EQ *GT*).GRTER (*&OPRATOR* EQ *NE*).NOTEQ .ERRIO
806 .EQUAL 807 808 809 .LESS 810 811	8 N E 6 4 G O 8 N L 8 4 G O	B&LAB C&LAB • END B&LAB C&LAB • END
812 .LESSEQ 813 814 815 .GRTEQ 816 817	BH B AGO BL B AGO	B&LAB C&LAB • END B&LAB C&LAB • END
818 •GRTER 819 820 821 •NDTEQ 822	BNH B AGO BF B	B&LAB C&LAB • FND B&LAB C&LAB

823		AGO	• END
824			•
825			THE POLYDUP CEN OF THEFTHEFTONS DESTROYED AND
826			THE FOLLOWING SET OF INSTRUCTIONS RESTORES THE DRIG-
827			INAL CONTENTS OF THE WORKING REGISTERS AND CAUSES
828 829			EITHER THE EXECUTION OF PROGRAM INSTRUCTIONS (IF THE CASE CONDITION IS TRUE) OF A BRANCH TO THE FEND-
830			CASE MACRO (IF THE CASE CONDITION IS FALSE).
831			CASE MACRO (IF THE CASE CONDITION IS FALSE).
832			
	. END	ANOP	
	BELAB	L	LO, SAVELO
835		Ĺ	11, SAVE11
836		B	&LABEL
837	CELAB	£	10,SAVE10
838		L	11, SAVE11
839		AGO	• STOP
840			
841			
842			BELOW ARE THE ERROR CONDITIONS WHICH THE MACRO MAY
843			DETECT. WHENEVER AN ERROR CONDITION IS RAISED, A
844			MESSAGE IS PRINTED AND THE CASE CONDITION IS SET TO
845 846			FALSE.
847			
	ERRI	MNOTE	*, MORE THAN ONE LEFT OPERAND SPECIFIED FOR CASE!
849	U QUILLE	MNOTE	*, 'CONDITION'
850		AGO	• END
851	• EPR2	MNOTE	*, MORE THAN ONE LITERAL SPECIFIED IN COMDITION
852		MNOTE	*, * STATEMENT *
853		AGO	• RETURN
854	• ERR3	MNOTE	*, ***REGA** SPECIFIED IS NOT A SELF-DEFINING *
855		MNOTE	*, *NUMERIC*
856		AGN	• 5ND
	• ERR4	MNOTE	*, *** PEGA** DOES NOT SPECIFY A VALID REGISTER*
858 859		MNOTE	*, 'NUMBER'
	• ERRS	AGS MNDTE	• END 1 GET ODERAND WAS REEN COROTETED FOR STORE
861	e CNNJ	втсим	*,'ND LEFT DPERAND HAS BEEN SPECIFIED FOR CASE' *,'CONDITION'
862		AGN	END -
	eR86		*, MORE THAN ONE RIGHT OPERAND SPECIFIED FOR CASE!
864		MNOTE	*, 'CONDITION'
865		AGO	END
866	ERR7	MNOTE	*,***REGB** SPECIFIED IS NOT A SELF-DEFINING*
887		MNOTE	*, NUMERIC:
868	77 M 64 -	4G 1	• END
	• Ed 38	MNOTE	*, ***REGB** DOES NOT SPECIFY A VALID REGISTER*
	<u></u>	MNOTE	*, *NUMBER1
871	• ERR9	AGO	• END
873	e CILLA	MNOTE MNOTE	*, 'OPERATOR FOR CASE CONDITION HAS NOT BEEN! *, 'SPECIFIED!
		AGO	•END
	.ERRIO	MNOTE	*, 'ILLEGAL OPERATOR SPECIFIED FOR A CASE CONDITION'
876		AGO	. END-
877	• STOP	MEND	

# The Source Listing for the ENDCASE Macro

STMT SOURCE STATEMENT

879 \* THE ENDCASE MACRO PROVIDES THE BRANCHING LOCATION 880 \* FOR A CASE MACRO WHOSE CONDITION IS FALSE. IT ALSO

881 882 883 884 885 886 885 886 885 886 885 889 890 891 892 893	* * * * * * * * * * * * * * * * * * * *	• •	MARKS THE END OF THE PRIGRAM INSTRUCTIONS TO BE IN- CLUDED WITHIN THE CASE STATEMENT. THE ENDCASE MACRO MAY BE SPECIFIED IN DNE OF TWO MODES. THE FIRST MODE (OPTION=1) WILL CAUSE A BPANCH TO A SPECIFIED LOCATION 'LAB' IF THE CASE CONDITION WAS TRUE (THIS WILL PERMIT THE PRIGRAMMEP TO BYPASS ADDITIONAL CASE STATEMENTS WHICH FOLLOW IN LINE). THE SECOND MODE (OPTION=2) CAUSES PROGRAM CONTROL TO BE PASSED TO THE MEXT INSTRUCTION (THIS WILL PERMIT THE TESTING OF SUCCESSIVE CASE STATEMENTS). 'LABEL' SHOULD BE SPECIFIED WITH THE SAME VALUE AS THE 'LABEL' PARAM- ETER IN THE CORRESPONDING CASE MACRO.
894 895	, <b>*</b>	MACRO	
<sup>7</sup> 896 897 898 898		ENDCASE AGO	&OPTION=,&LABEL=,&LAB= _SKIP1
900 901 902 903 904 905	* * *		THE FOLLOWING STATEMENTS DETERMINE THE OPTION TO USE. IF THE PARAMETER HAS NOT BEEN SPECIFIED OR IF IT HAS BEEN INCORRECTLY SPECIFIED, A MESSAGE IS PRINTED AND OPTION=1 IS ASSUMED.
906 907 908 909 910 911	• SKIP1	AIF AIF AIF AGO , ,	('EOPTION' EQ '1'}.OPTION1 ('EOPTION' EQ '2').OPTION2 ('EOPTION' EQ '').ERR1 (T'EOPTION NE 'N').ERR2 .ERR3
912 913 914 915 916 917	* *		BELOW ARE THE ERROR MESSAGES GENERATED BY THE MACRO. EACH ONE REFERS TO THE 'OPTION' PARAMETER AND CAUSES IT TO BE SET TO ONE.
918 919 920	• ERR1	MNOTE MNOTE Agn MNOTE	<pre>*,'''OPTION'' NOT SPECIFIED FOR ENDCASE (OPTION' *,'SET TO ONE)' OPTION1 *,'''OPTION'' SPECIFIED IS NOT A SELF-DEFINING' </pre>
922 923 924 925 926 927	• ERR 3	MNOTE AGO MNOTE MNOTE AGO	<pre>*, 'NUMERIC (OPTION SET TO ONE)' OPTION1 *,'''OPTION'' SPECIFIED IS AN TLLEGAL NUMERIC VALUE' *,'(OPTION SET TO ONE)' OPTION1</pre>
928 929 930 931 932	* *		THE FOLLOWING STATEMENTS INCLUDE THE BRANCH AND/OR *LABEL* INSTRUCTIONS GENERATED BY THE MACRO.
934	•OPTION1 •OPTION2 &LABEL		£L4B ★

### The Source Listing for the ELSE Macro

STMT SOURCE STATEMENT

938 \* THE ELSE MACRO IS MERELY A MACRO DEFINITION NAME TO 939 \* DENOTE THE BEGINNING OF THE ALTERNATIVE SET OF IN+

940 * 941 * 942 * 943 * 944 *		STRUCTIONS TO BE EXECUTED WHEN ALL OF THE PROCEED- ING CASE STATEMENTS HAVE FAILED. AN 'ENDELSE' MACPO SHOULD ALWAYS BE SPECIFIED WHEN THE ELSE MACRO IS USED.
945 * 946 947 948	MACRO ELSE MEND	

#### The Source Listing for the ENDELSE Macro

# STMTSOURCE STATEMENT950 \*THE ENDELSE MACRD GENERATES A LABEL TO WHICH A SET951 \*OF CASE STATEMENTS WILL BRANCH WHEN ANY OR ALL HAVE952 \*HAD 'TRUE' CONDITIONS. IT ALSO MARKS THE END OF953 \*THE SET OF INSTRUCTIONS ENCOMPASSED BY THE ELSE

954	*		MACRO.			0,		L L J L	
955	*								
956	*				•				
957	+	MACRO		· •					
958		ENDELSE	£LA8=						
959	8LAB	EQU	* .				-		
960		MEND							

#### Sample Program I Illustrating the CASE, ENDCASE, ELSE, and ENDELSE Macros

STMT	SOURCE	STATEMENT	
962 963 979 985 987 988 989 990 991 992 992 993 994 995	FRISBEE START	OP EN	NDGEN 12, SAVEAREA (DUMPAREA, DUTPUT) 2,0 2,0 3,0 4,0 5,0 6,0 7,0 8,0 9,0 10,0 11,0
996 997 998 999			14,0 15,0
1000 1012 1013	*	SVAP	ID=1,DCB=DUMPAREA,POATA=REGS
1014 1028 1029 1030 1031 1032 1033		CASE	LABEL=A,LITA=O,OPRATOR=NE,REGB=15 A 4,=F*1* A 5,=F*1* A 6,=F*1* A 7,=F*1* A 8,=F*1* A 8,=F*1*

					-	
1034		ENDCAS	F	TTON=1.LABEL=A.LA	8 = H60 F	
1037	*			LIC: IILHDLL-RJLA		
1038						
1039		CASE		BEL=3, REGA=14, OPR	ATOR=FO.LITS=0	
1053			,	4,=F121	ATOM CHIELIÓ O	
1054				5,=E121		
1055				6,=F*2*		
1055				7,=F121		
1057				8,=F*2*		
1058				9,=F*2*		
1059		ENDCAS	c		9-4 <b>5</b> 0 <b>5</b>	
1062	*	ENDUAS	) C	TION=1,LABEL=B,LA	9-nexe	
1063						
1064	т.	CASE		CA-Y 1000-Y 14051	-6 000 4700-1 7	
1077		6436		CA=X,LOC8=Y,LA8EL	=C+OFRATUR=L1	
				4,=F*3*		
1078	•			5,=F'3*		
1079				6,=F131		·
1080				7,=F'3'		
1081				8,===13		•
1082				9,==131	· · · · · · ·	
1083	.4.	ENDCAS	E	TICM=1, LABEL=C, LA	B=HERE	
1086						•
1097	<b>X</b>					
1088		ELSE				
1089	·			4,===4		
1090				5,=F'4'		
1091				6,=F141		
1092				7,==*4*		
1093	1. A.	-	·	8 * = E * 4 *		
1094				9,===4*	•	
1095		ENDELS	Ε	AB=HERE		
1097	<i>ф</i>				·	
1098	*					
1099		SN AP		)=2,DCB=DUMPAREA,P.	DATA=REGS	
1111	*			,		
1112	*				,	
1113		EXIT				
1118		CLOSE		IMPAREA		
1124	SAVEAREA	00		1A ( 0 )		
1125		DS .		<b>.</b>		
1126	ZERO	DC		:+0+		
1127	SAVE8	DS		:		
1128	SAVE9	05		<b>:</b>		
1129	SAVE10	DS		:		
1130	SAVE11	05		:		
	POINTER	DC				
	STACK	DS		IOF		
1133		DC		141 		
1134		00		:+8+	•	
	DUMPAREA	-		NAME=TEAM, DSORG=P.	S-RECEM=VBA-	
			MACR	W, BLKSIZE=882,LRE		15
1186		END		in provide a same to the part of the part	the the state of the state	
1187			=F!(			
1188			=F+]	•		
1189			=F+2			
1190			=F•3			
1191			=F•4			
						`

X

REGS AT ENTRY TO	SNAP			ID = 001
REGS 0-7	00000030	9001888C	00000000	0000000
	00000000	00000000	00000000	00000000
REGS 8-15	00000000	00000000	00000000	0000000
	5001B820	00018908	00000000	00000000
END OF SNAP				

REGS AT ENTRY TO	) SNAP		•••	ID = 002
REGS 0-7	000002A0 00000032	A001899C 00000002	00000000 00000002	00000000
REGS 8-15	00000002 50018820	00000002 00018908	00000000	00000000
END OF SNAP				

# Program I with Expanded Macros

#### STMT SOURCE STATEMENT

962 FRISBEE		
963+FRISBEE		
964*		FRISBEE DECLARE NAME ENTRY *,12 DECLARE BASE ADDRESSIBILITY.
965+		
966+		15,0 (INITIAL ADDRESSIBILITY).
967+	B	
968*	0C	AL1(7), CL7'FRISBEE' ID LENGTH AND ID
969+		15,0 (RESET INITIAL ADDRESSIBILITY
970+		15,0 ABSOLUTE ENTRY POINTI.
971+	STM.	
972*	LR	12,15 SETUP RASE PEGISTER.
973+	ST	13,SAVEANEA+4 CHAIN BACK
974+	LA	O, SAVEAREA CHAIN FORWARD
975+	ST	0,8(0,13)
976+	્ર	13,0 SET UP SAVE AREA POINTER
977+	USING	SAVEAREA, 13 AND ADDRESSABILITY
978 START	<b>JPEN</b>	(DUMPAREA, OUTPUT)
979+	CNOP	0,4 ALIGN LIST TO FULLWORD
980+START	BAL	1,*+8 LOAD REGI W/LIST ADDR.
981+	DC	ALI(143) OPTION BYTE
982+	00-	AL3(DIMPAREA) DCB ADDRESS
983+	SVÇ	19 ISSUE OPEN SVC
984	LA	2,0
985	64	2,0
986	LA	3,0

987		4 <sub>2</sub> 0 5 , 0
988 989	LA LA	6,0
990	LA	7,0
991	LA	8,0
992	LA	9±0
993	LA	10,0
994	LA	11,0
995	LA	14,0
996	LA	15,0
997¥.		·
998 *		
999	SNAP	TD=1,DCB=DUMPAREA,PDATA=REGS
1000+	CNOP	0,4 1,IHB0003 BRANCH ARDUND PARAM LIST
1001+ 1002+	BAL DC	ALI(1) ID NUMBER
1003+	DC	ALI(0)
1004+	DC	ALI(130) OPTION FLAGS
1005+	DC	AL1(32) OPTION FLAGS
1006+	DC DC	A (DUMPAREA) DCB ADDRESS
1007*	00	A(O) TCB ADDRESS
1008+	00	A(0) ADDRESS OF SNAP-SHOT LIST
1009+IHB0003	05	OH
1010+	SVC	51
1011 *		
1012 *	<b>*</b> • • **	
1013	CASE	LABEL=A,LITA=O,DPRATOR=NE,REGB=15
1014+	ST	10, SAVELO
1015+ 1016+	ST LA	11,SAVE11 10,0
1017+	4	10;=F'0'
1018+	LR	11,15
1019+	CR	10,11
1020+	8E	B4
1021+	B	C4
1022+84	L	10, SAVE10
1023+	L	11, SAVE11
1024+	8	Α
1025+04	L	10, SAVE10
1026+ 1027	L .	11, SAVE11
1028		A 4,=F*1* A 5,=F*2*
1029		A = 5 + 2 + 2 + 2 + 2 + 2 + 2 + 2 + 2 + 2 +
1030		A 7,=F 1
1031		A 8,=F*1*
1032		A 9,=F!1*
1033	ENDCAS	• • • • • • • • • • • • • • • •
1034+	B	HERE
1035+4	EQU	*
1036 * 1037 *		
1038	CASE	1 ABEL TR. RECATIA, ODDATOR-EO ( TTD-O
1039+	ST	LABEL=B,REGA=14,DPRATOR=EQ,LITB=0 10,SAVE10
1040+	ST	11, SAVE11
1041+	٤R	10,14
1042+	LA	11,0
1043*	A	11,=F*0*
1044+	CR	10,11
1045+ 1046+	BNE	B6
1045+	B ·	C6
1043+	ւ Լ	10,SAVE10 11,SAVE11
1049+	8 .	B
1050+06	Ľ.	10, SAVE10
1051+	L	11,SAVE11
1052		A 4,==F*2*
1053		A 5,=F*2*

1054 Α . 6,=E\*2\* 1055 7 = = + 2 \* А 8,=E+2+ 1056 Δ 1057 9,=F\*2\* A OPTION=1,LABEL=B,LAB=HERE 1059 ENDCASE 1059\* B HERE 1060+8 EQU ŵ. 1061 \* 1062 \* 1063 CASE LOCA=X,LOCB=Y,LABEL=C,OPRATOR=LT ST 10: SAVE10 1064+ 11, SAVE11 1065+ ST 1066+ L 10.X 1067+ L 11.Y 10,11 1068+ CR 1069+ BNL 88 1070+ в C 8 10, SAVE10 1071+B8 L 1072+ 11, SAVE11 L 1073 +8. C. 10, SAVE10 1074+08 L 1075+ L 11, SAVE11 4,==13! 1076 Α 1077 5,=F'3' 4 1078 6,==131 Δ 1079 Δ 7.=F-3 1080 8,=F'3' A 1081 9,=F\*3\* A 1082 ENOCASE OPTION=1,LABEL=C,LAB=HERE 1083+ B HERE 1084+C EQU \* 1085 \* 1086 \* 1087 ELSE 4,=≓4 1088 A 1089 ٨ 5,=F141 1090 A 6,=F141 7,=F'4' 1091 Α 8,===4 1092 А 1093 9,=F+4+ Δ 1094 ENDELSE LAB=HERE 1095+HERE EQU \* 1096 \* 1097 \* 1098 SNAP ID=2,DCB=DUMPAREA,PDATA=REGS 1099+ CNOP 0+4 1100+ BAL 1, THBOO12 BRANCH AROUND PARAM LIST 1101+ 36 AL1(2) ID NUMBER 1102+ 00 AL1(0) 1103+ 00 ALI(130) OPTION FLAGS 1104+ 00 AL1(32) OPTION FLAGS 1105+ 0C A(DUMPAREA) DCB ADDRESS 1106+ DC A(O) TCB ADDRESS 1107+ DC A(0) ADDRESS OF SNAP-SHOT LIST 1108+IH80012 DS 0H 1109+ SVC 51 1110 \* 1111 \* 1112 EXIT 1113+ 1 13,4(,13) POP UP SAVE AREA 1114+ Ľ٩ 14,12,12(13) RESTORE REGISTERS 1115+ MVT 12(13), X'FF' FLAG EXIT 1116+ 8R 14 RETURN 1117 CLOSE DUMPAREA 1118+ CAUb 0+4 ALIGN LIST TO FULLWORD 11194 BAL 1,\*+8 LOAD REGI W/LIST ADDR 1120+ DC ALI(128) OPTION BYTE

1121+ 1122+	DC SVC	AL3(DUMPAPEA) DCB ADDRESS 20 ISSUE CLOSE SVC
1123 SAVEAREA 124 125 ZERO 126 SAVE8 127 SAVE9 128 SAVE10 129 SAVE11 130 POINTER 1131 STACK 1132 X 1133 Y 1134 DUMPAREA	DS DC DS DS DS DC DC DC DC	18A(0) OF 1F*0* 1F 1F 1F 1F*0* 100F 1F*4* 1F*8* ODNAME=TEAM,DSURG=PS,RECFM=VBA, X MACRF=W,BLKSIZE=882,LRECL=125
1136+* 1137** 1138+DUMPAREA	חר	OATA CONTROL BLOCK OF!O! ORIGIN ON WORD BOUNDARY
1140+*		DIRECT ACCESS DEVICE INTERFACE
· · · ·	05	· · · ·
1142+ 1143*	00 00	BL16*O* FDAD, DVTBL A(0) KEYLE, DEVT, TRBAL
1145+*		COMMON ACCESS METHOD INTERFACE
1147+ 1148+ 1149+ 1150+ 1151+	DC DC DC DC DC	AL1(0) BUFND AL3(1) BUFCB AL2(0) BUFL BL2*010000000000000 * DSDRG A(1) IDBAD
1153**		FOUNDATION EXTENSION
1155+ 1156+ 1157+ 1158+	DC DC DC DC	BL1:0000000: BFTEK,BFLN,HIARCHY AL3(1) EDDAD BL1:01010109: RECFM AL3(0) EXLST
1160+*		FOUNDATION BLOCK
1162+ 1163+ 1164+ 1165+	DC DC DC DC	CL8*TEAM* DDMAME BL1*0000010* CFLGS BL1*000000000100000* MACR BL2*00000000000000000* MACR
1167**		BSAM-BPAM-QSAM INTERFACE
1169+ 1170+ 1171+ 1172+ 1173+ 1174+ 1175+ 1175+ 1176+ 1177*	00 00 00 00 00 00 00 00 00 00	BL1'00000000; RER1 AL3(1) CHECK, GERR, PERR A(1) SYNAD H'0' CIND1, CIND2 AL2(882) BLKSIZE F'0' WCPO, WCPL, OFFSR, OFFSW A(1) IOBA AL1(0) NCP AL3(1) EOBR, EOBAD
1179+*	~~	BSAM-BPAM INTERFACE
1181+ 1182+ 1183+ 1184+	0C 0C 0C 0C	A(1) EDBW H'O' DIRCT AL2(125) LRECL A(1) CNTRL, NOTE, POINT

1185	END		
1186	= f	= 101	
1187	± [	1-	
1188	= f	=121	
1189	== {	= 131	
1190	= {	F141	

#### STHT SOURCE STATEMENT

4		
962 FRISBEE	ENTER	12, SAVEAREA
963+FR1SBEE	0 S	0H (
964+	ENTRY	FRISBEE DECLARE NAME ENTRY
965+	USING	*,12 DECLARE BASE ADDRESSIBILITY.
966+	BALR	15,0 (INITIAL ADDRESSIBILITY).
967+	8	12(,15) BRANCH AROUND ID FIELD
968+	DC	ALI(7), CUT'FRISBEE' ID LENGTH AND ID
969+	BCTR	15,0 (RESET INITIAL ADDRESSIBILITY
	BCTR	
970+ *		15,0 ABSOLUTE ENTRY POINT).
971+	STM	14,12,12(13) SAVE REGISTERS
972+	LR	12,15 SETUP BASE REGISTER.
973+	ST	13, SAVEAREA+4 CHAIN BACK
974+	LA	O,SAVEAREA CHAIN FORWARD
975+	ST	0,810,131
976+	LR	13.0 SFT UP SAVE AREA POINTER
°977+	USING	SAVEAREA, 13 AND ADDRESSABILITY
978 START	OPEN	(DUMPAREA, OUTPUT)
979+	CNOP	0,4 ALIGN LIST TO FULLWARD
980+START	BAL	1,*+8 LOAD REG1 W/LIST ADDR.
981+	DC	ALI(143) OPTION BYTE
932+	DC	AL3(DUMPAREA) DCB ADDRESS
983+	SVC	19 ISSUE OPEN SVC
984	LA	2+0
985	LA ·	3.0
986	LA	4,0
987	LA	5+0
988	LA	6+0
989	LA	7:0
990	LA	8,0
991	LA	9,0
992	LA	10,0
993	LA	11.0
994	LA	14,0
995	LA	15,0
996 *		20170
997 ×		
928	<b>C N</b>	NAP ID=1,0CB=DUMPAREA,PDATA=REGS
999+	CNOP	0.4
1000+	BAL	• •
1001+	DC	LITHSDOOS BRANCH AROUND PARAM LIST
		ALI(1) TO NUMBER
1002+	00	ALI(0)
1003+	20	ALI(130) OPTION FLAGS
1004+	DC	ALL(32) OPTION FLAGS
1005+	00	A (DUMPAREA) OCB ADDRESS
1006+	DC.	A(O) TCB ADDRESS
1007+	00	ALOJ ADDRESS OF SNAP-SHOT LIST
1008+TH80003	05	OH
1009+	SVC	51
1010 *		
1011 *		
1012	CASE	LABEL=A,LITA=0,OPRATOR=NE,REGB=15

1013+	ST	10, SAVE10
1014*		
	ST	11,SAVE11
1015+	LA	10,0
1016+	4	10,=F*0*
		•
1017+	LR	11,15
10184 '	CR	10.11
1019+	BE	
		84
1020+	8	C4
1021+84	Ľ	10, SAVE10
1022+	Ē	
		11,SAVE11
1023+	. 8	A
1024+04	L	10, SAVE10
1025+		
	L	11, SAVE11
1026		
1027		A 5,=F*1*
1028		$A \qquad 6_{s} = F^{s} 1^{s}$
1029		
1030		A 8, ≈F*1.
1031		
		A 9,=F*1*
1032	ENDCA	SE OPTION=4,LABEL=A,LAB=HERE
1033		
1034		*, 'OPTION' SPECIFIED IS AN ILLEGAL NUMERIC VALUE
		*, (OPTION SET TO ONE)
1035+	в	HERE
1036+4	EQU	*
	240	<b>T</b>
1037 *		
1038 *		
1039	CASE	
		LABEL=B,REGA=14,OPRATOR=EQ,LITB=0
1040*	ST	10,SAVE10
1041+	ST	11.SAVE11
1042+	LR	
		10,14
1043+	LA	11,0
1044+	A	11,=F*0*
1045+	CR	
		10,11
1046+	8 N E	B 6
1047+	В	C 6
1048+86		
	L	10, SAVE10
1049+	Ľ	IL, SAVE11
1050+	8	B
1051+66		
	Ĺ	10, SAVE10
1052+	L	11, SAVE11
1053		A $4_{\mu} = F^{*} 2^{*}$
1054		
		A 5,=F!2"
1055		A 6, =F <sup>1</sup> 2 <sup>+</sup>
1056		A 7,=F*2*
1057		
		A 8,=E*2*
1058		A 9,=F*2*
1059	ENDCA	SE OPTION=1, LABEL=8, LAB=HERE
1050+	8	HERE
1061+8	EQU	*
1062 *		
1063 *		· · · ·
	<b></b>	
1064	CASE	LOCA=X,LOCB=Y,LABEL=C,OPRATOR=LT
1065+	ST ,	10, SAVE10
1066+	ST	11, SAVE11
1067+	L	10,X
1068+	- L	11,Y
1069+	ČR	10,11
		• • •
1070+	BNL	88
1071+	8	C 8
1072+88	L	10, SAVE10
1073+		11, SAVE11
1074+	8	C
1075+C8	L.	10, SAVE10
1076+		
	L	11.SAVE11
1077		A 4,⇒F <sup>‡</sup> 3 <sup>#</sup>
1078		A 5 <sub>f</sub> ≈F+3
~		

16Ž

1079 6,==131 A 1080 A 7,=F\*3\* 8,=F+3+ 1081 A 9,=F+3+ 1082 A ENDCASE 1083 OPTION=1,LABEL=C,LAB=HERE 1084+ HERE 8 EQU 1085+6 **1** ٠., 1086 \* 1087 \* 1088 ELSE 1089 ·A · 4,=F+41 5,=F'4' 1090 A . 6;=F!4! 1091 A 1092 7,=F+4+ A 8,=F+4+ 1093 A 1094 9,=F+4+ A 1095 ENDELSE LAB=HERE 1096+HERE EQU \* 1097 \* 1098 \* 1099 \* 1100 \* 1101 SNAP ID=2 DCB=DUMPAREA, PDATA=REGS 1102+ CNOP 0,4 1103+ 1. THB0012 BRANCH AROUND PARAM LIST BAL 1104 +**DC** AL1(2) ID NUMBER 1105 +DC AL1(0) 1106 +DC AL1(130) OPTION FLAGS ALI(32) OPTION FLAGS 1107 +DC. 1108+ DC A(DUMPAREA) DCB ADDRESS 1109+ DC A(0) TCB ADDRESS 1110 +DC A(O) ADDRESS OF SNAP-SHOT LIST 1111+1H80012 DS ØН 1112+ SVC 51 1113 \* 1114 \* 1115 EXIT 1116+ Ł 13,4(,13) POP UP SAVE AREA 1117 +LM 14,12,12(13) RESTORE REGISTERS 1118+ HV I 12(13), X'FF' FLAG EXIT 1119+ 88 14 RETURN 1120 CLOSE DUMPAPEA 1121+ CNOP 0,4 ALIGN LIST TO FULLWORD 1122 +1,\*+8 LOAD REGI W/LIST ADDR BAL 1123 +DC ALI(128) OPTION BYTE 1124+ DC AL3(DUMPAREA) DCB ADDRESS SVC 1125+ 20 ISSUE CLOSE SVC 1126 SAVEAREA DC 18A(0) 1127 DS 0F 1128 ZEPD 16101 DC. 1129 SAVE8 18 DS 16 1130 SAVE9 DS. 1131 SAVE10 1132 SAVE11 0.5 1 F DS. 1F 1133 POINTER 15.0. DC 1134 STACK DS 100F 1135 X DC 16141 16181 1136 Y 00 1137 DUMPAREA DCB DDNAME≈YEAM, DSORG=PS, RECFM=VBA, · MACRF=W, BLKSIZE=882, LRECL=125 1139+\* DATA CONTROL BLOCK 1140+\* 1141+DUMPAREA DC-OFTOT OFTGIN ON WORD BOUNDARY 1143+\* DIRECT ACCESS DEVICE INTERFACE

Х

11454	ÐC	BL16'0' FDAD, DVTBL
1146+	00	A(0) KEYLE, DEVT, TRBAL
		• •
1148+*		COMMON ACCESS METHOD INTERFACE
1150+	00	ALI(O) BUENG
1151*	DC	AL3(1) BUFCB
1152∻ 1153+	DC	AL2(0) BUFL
11544	DC DC	BL2'010000000000000 DSORG
11344		ALLI IOBAD
1156+*		FOUNDATION EXTENSION
1158+	ĐC	
1159+	DC	BL1'00000000 BETEK, BELN, HIARCHY AL3(1) EODAD
1160+	DC	BL1'01010100' RECFM
1161+	0C 0C	AL3(0) EXIST
	00	ACTOP CALLS
1163+*		FOUNDATION BLOCK
1165+	00	CL8*TEAM* DDNAME
1166+	00	BL1'0000010' DFLGS
1167+	00	BL1:0000000: IFLG
1168+	DC	BL2*000000000000000 MACR
1170+*		BSAM-BPAM-QSAM INTERFACE
1170	D.2 .	
1172+ 1173+	0C -	BL1'0000000' RER1
1174+	00	AL3(1) CHECK, GERR, PERR A(1) SYNAD
1175+	00	H'O' CIND1, CIND2
1176+	DC	AL2(882) BLKSIZE
1177+	DC	F'O' WCPO, WCPL, OFFSR, OFFSW
1178+	90	A(1) IOBA
1179+	DC	ALI(0) NCP
1180+	DC	AL3(1) EOBR, EOBAD
1182**		BSAM-BPAM INTERFACE
1184+	DC	A(1) EDBW
1185+	DC	HSOF DIRCT
1186+	DC	AL2(125) LRECL
1187+	DC	A(1) CNTRL, NOTE, POINT
1188	END	/-
1189		=F+0*
1190		=F•11
1191		=F121
1192		= F131
1193		= F * 4 *

## Sample Program K Illustrating a CASE Macro Coded Incorrectly

STMT SOURCE	STATEMENT
962 FPISBEE 963+FRISBEE 964+ 965+ 966+ 966+ 967+ 968+ 969+	ENTER 12, SAVEAREA DS OH ENTRY FRISBEE DECLARE NAME ENTRY USING *,12 DECLARE BASE ADDRESSIBILITY. BALR 15,0 (INITIAL ADDRESSIBILITY). B 12(,15) BPANCH AROUND ID FIELD DC AL1(7), CL7*FRISBEE* ID LENGTH AND ID BCTR 15,0 (RESET INITIAL ADDRESSIBILITY

<i>î</i> ,	15.0 ABSOLUTE ENTRY POINT). 14.12.12(13) SAVE REGISTERS 12.15 SETUP BASE REGISTER. 13.SAVEAPEA+4 CHAIN BACK 0.SAVEAPEA CHAIN FORWARD 0.8(0.13)
NG N P	13.0 SET UP SAVE AREA POINTER SAVEAPEA,13 AND ADDRESSABILITY (DUMPAFEA,OUTPUT) 0.4 ALIGN LIST TO FULLADRO 1.**8 LDAD REGI W/LIST ADDR. AL1(143) OPTION BYTE AL3(DUMPAREA) DCB ADDRESS 19 ISSUE OPEN SVC 2.0
	$3 \cdot 0$ $4 \cdot 0$ $5 \cdot 0$ $6 \cdot 3$ $7 \cdot 0$ $8 \cdot 0$ $9 \cdot 0$ $1 \cdot 0$ 1
	AP ID=1,DCB=DUMPAREA,PDATA=REGS 0,4 1,IHB0003 BRANCH ARDUND PARAM LIST ALI(1) ID NUMBER ALI(1) ALI(130) OPTION FLAGS ALI(32) OPTION FLAGS A(DUMPAREA) DCB ADDRESS A(0) TCB ADDRESS A(0) ADDRESS OF SNAP-SHOT LIST OH 51

1009+	210	51
1010 *		[1] A set of the se
1011 *		
1012	CASE	LABEL=A,LITA=O,DPRATOR=NE,REGB=15
1013+	ST	10, SAVE10
1014+	ST	11, SAVE11
1015+	LA	10,0
1016+	A	10,====01
1017+	LR	11,15
1018+	CR	10,11
1019+	BE	64
1020+	8	C4
1021+84	L	10,SAVEIO
1022+	L,	11,SAVELL
1023+	B	A
1024+04	L,	10, SAVE10
1025+	L	11, SAVELL
1026		A 4,=F*1*
1027		A 5,=F*1*
1028		A 6,=F*1*
1029		A 7,=F*1*
1030		A . 8,=F*1*
1031		A 9,====1*
1032	ENDCA	SE OPTION=1,LABEL=A,LAB=HERE
1033+	8	HERE
1034+A	EQU	<b>*</b> · · · · · · · · · · · · · · · · · · ·
1035 *		

1035 \* 1036 \*

,

970+

971+

972+ 973+

974+

975+

976+

977+

979+

981+

982+

983+

984

985

986

987

288

989

**9**90

991

992

993

994

995

996 \* 997 \* 998

999+

1000+ 1001+

1002+ 1003+ 1004+ 1005+

1006+

1007+

1009+

1008+1HB0003

978 START

980+START

BCTR

STM LR

ST.

L A

ST

LR

US 1NG

OP EN

CNOP

. BAL

DC

DC

LA

L٨

LΑ

LA

LA

LA

Ł٨

LA

LΑ

LΑ

LA

LA

CNOP

BAL DC

00 00 00

DC.

DC

DC

DS

SVC

SVC

1077	
1037 1038+	CASE LABEL=B,PEGA=14,OPRATOR=EQ,LITA=0
1039+	ST 10,SAVE10 ST 11,SAVE11
1040	*, MORE THAN ONE LEFT OPERAND SPECIFIED FOR CASE
1041	*+CONDITION
1042+B6	L 10,SAVE10
1043+	L 11,SAVE11
1044+	B B
1045+C6 1046+	L 10, SAVE10
1047	L 11, SAVE11
1048	A 4,=F!2! A 5,=F!2!
1049	A 6+=F*2*
1050	A 7,=F*2*
1051	A 8,=F*2*
1052 1053	A 9,=F*2*
1054+	ENDCASE OPTION=1,LABEL=B,LAB=HERE
1055+B	B HERE EQU *
1056 *	
1057 *	
1058	CASE LOCA=X,LOCB=Y,LABEL=C,OPRATOR=LT
1059+	ST IO, SAVEIO
1060+ 1061+	ST 11, SAVE11
1062+	
1063+	L 11,Y CŘ 10,11
1064+	BNL B8
1065+	B C8
1066+58	L 10, SAVE 10
1067+	L 11, SAVE11
1068+ 1069+C8	B C L 10.SAVE10
1070+	L 10,SAVE10 L 11,SAVE11
1071	$A \qquad 4_{*} = \mathbb{P}^{*} 3^{*}$
1072	A 5,=F*3*
1073	A 6,=F*3'
1074	A 7,=F*3*
1075 1076	A 8,≂F*3* A 9,≖F*3*
1077	A 9,=F*3* ENDCASE OPTION=1,LABEL=C,LAB=HERE
1078+	B HERE
1079+C	EQU *
1080 *	
1081 * 1082	C1 CC
1033	ELSE A. 4, =F141
1084	A 5,∞F*4*
1085	A 6,=F*4*
1086	A 7,=F*4*
1087	$A = 8_y = F^* 4^{t}$
1088 1089 -	A 9,=E*4* ENDELSE LAB=HERE
1090+HERE	EQU *
1091 *	
1092 *	
1093 *	
1094 * 1095	SNAP I D=2.DCB=DUMPAREA.PDATA=REGS
1096+	CNOP 0,4
1097+	BAL 1, IHBOO12 BRANCH AROUND PARAM LIST
1098+	DC AL1(2) ID NUMBER
1099+ 1100+	DC ALI(0) DC ALI(0) ODTION FLACE
1101+	DC ALL(130) OPTION FLAGS DC AL1(32) OPTION FLAGS
1102+	DC A(DUMPAREA) DCB ADDRESS

1103+ 1104+ 1105+IHB0012 1106+ 1107 *	0C 0C 0S SVC	A(0) TCB ADDRESS A(0) ADDRESS OF SNAP-SHOT LIST OH 51
1108 * 1109 1110+ 1112+ 1112+ 1113+ 1114 1115+ 1115+ 1116+ 1117+ 1118+- 1119+	EXIT L MVI BR CLOSE CNOP BAL DC SVC	13,4(,13) POP UP SAVE AREA 14,12,12(13) RESTORE REGISTERS 12(13),X*FF* FLAG EXIT 14 RETURN DUMPAREA 0,4 ALIGN LIST TO FULLADRD 1.*+8 LOAD REGI W/LIST ADDR AL1(128) OPTION BYTE AL3(DUMPAREA) DCB ADDRESS 20 ISSUE GLOSE SVC
1120 SAVEAREA 1121 1122 ZERD 1123 SAVE8 1124 SAVE9 1125 SAVE10 1126 SAVE10 1126 SAVE11 1127 POINTER 1128 STACK 1129 X 1130 Y 1131 DUMPAREA	DS DC DS DS DS DC DC DC DC DC	18A(0) OF 1F*0* 1F 1F 1F 1F*0* 100F 1F*4* 1F*8* DONAME=TEAM, DSORG=PS, RECFM=VBA*, MACRF=W, BLKSIZE=882, LRECL=125
1133+*		DATA CONTROL BLOCK
1134+* 1135+DUMPAREA	DC	OF OF BRIGIN ON WORD BOUNDARY
1137+*		DIRECT ACCESS DEVICE INTERFACE
1139+	DC	BL16'0' FDAD, DVTBL
1140+	DC -	A(O) KEYLE, DEVT, TRBAL
1144+ 1145+ 1146+ 1147+ 1148+	00 00 00 00 00	COMMON ACCESS METHOD INTERFACE AL1(0) BUFNO AL3(1) BUFCB AL2(0) BUFL BL2*01000000000000 DSDRG A(1) IOBAD
1150+*	20	FOUNDATION EXTENSION
1152+	DC	BL1*00000000* BFTEK, BFLN, HIARCHY
1153+ 1154+	DC DC	AL3(1) EDDAD BL1'01010100' RECEN
1155+	00	AL3(0) EXLST
1157**		FOUNDATION BLOCK
1159+ 1160+	DC DC	CL8'TEAN' DDNAME BL1'00000010' CFLGS
1161+ 1162+	DC DC	8L14000007004 IFLG 8L2400000000000000000 MACR
1164+*		BSAM-BPAM-QSAM INTERFACE
1166+	DC	BL1'0000000' RER1

X

1167+	00	AL3(1) CHECK, GERR, PERR
1168+	DC	
1169+	00	H'O' CINDI, CIND2
1170+	DC	AL2(882) BLKSIZE
1171+	00	FIOI WCPD, WCPL, DEESR, DEESW
1172+	00	A(1) 108A
1173+	00	ALI(0) NCP
1174+	20	
1176+*		BSAM-BPAN INTERFACE
1178+	DC	A(1) E08W
1179+	ÐC	H'O' DIRCT
1180+	00	AL2(125) LRECL
1181+	DC	A(1) CNTRL, NOTE, POINT
1182	END	
1183		=F101
1184		=F11
1135		= = = 2 =
1186		± F13t
1:87		====4

# Sample Program L Illustrating a CASE Macro Coded Incorrectly

.

#### STMT SOURCE STATEMENT

1962 FPISBEE	ENTER	12, SAVEAREA	
963+FRISBEE	DS	OH	
964+		FRISBEE DECLARE NAME ENTRY	
965+	USTNO	*+12 DECLARE BASE ADDRESSIBILITY.	
966+	BALR	15,0 (INITIAL ADDRESSIBILITY).	
967+	8	12(,15) BRANCH AROUND ID FIELD	
96B+	DC	ALIAN CLAEDING AND ID FIELD	
969+	BCTR	AL1(7), CL7'FRISBEE' ID LENGTH AND ID	
970+	BCTR	15,0 (RESET INITIAL ADDRESSIBILITY	
971+	STM	15,0 ABSOLUTE ENTRY POINT),	
972+	LR	14,12,12(13) SAVE REGISTERS	
973+	ST	12,15 SETUP BASE REGISTER.	
974+	LA.	13, SAVEAPEA+4 CHAIN BACK	
975+	ST	0, SAVEAREA CHAIN FORWARD	
976+			
977+		13,0 SET UP SAVE AREA POINTER	
978 START		SAVEAREA, 13 AND ADDRESSABILITY	
979+	OP EN CNOP	(DUMPAFEA, OUTPUT)	
980+START	BAL	0.4 ALIGN LIST TO FULLWORD	
981+	DC	1.*+8 LOAD REGI W/LIST ADOR.	
982+		AL1(143) OPTION BYTE	
9834	DC SV C	AL3(DUMPAREA) DCB ADDRESS	
984	SVC	19 ISSUE OPEN SVC	
985 985	LA	2,0	
	LA	3,0	
986	LA.	4,0	
987	LA	5,0	
988	LA	6,0	
989	LA	7,0	
990	LA	8,0	
991	LA	9,0	
992	L A	10,0	
993	LA .	11,0	
994	LA	14,0	
995	LA	15+0	
996 *			

997 \*

998	S	NAP ID=1, OCB=DUMPAREA, PDA TA=REGS
999+	CNOP	0,4
1000+	BAL	1. THBOOO3 BRANCH AROUND PARAM LIST
1001+	DC	ALI(1) ID NUMBER
1002+	DC	
		ALI(O)
1003+	DC	ALI(130) OPTION FLAGS
1004+	00	ALI(32) OPTION FLAGS
1005+	DC	A(DUMPAREA) DCB ADDRESS
1006+	00	A(O) TCB ADDRESS
1007+	DC	A(0) ADDRESS OF SNAP-SHOT LIST
1008+1HB0003	DS	OH
1009+	svc	
	346	51
1010 *		and the second
1011.*		•
1012	CASE	LABEL=A,LITA=O,OPRATOR=NE,REGB=15
1013+	ST	10, SAVE10
1014+	ST	11, SAVE11
10[5+	LA	10,0
1016+		
	Α	10,=F'0*
1017+	LR	11,15
1018+ '	CP	10,11
1019+	BE	84
1020+	в	C4
1021+84	L	10, SAVE10
1022+	Ĺ	11, SAVE11
1023+	B	A
	-	
1024+C4	L	10, SAVE10
1025+	L	11, SAVE11
1026		A 4,=F*1*
1027		A 5,=F*1*
1028		$A \qquad 6_{t} = F^{\dagger} 1^{\bullet}$
1029		A 7,=F*1*
1030		-
1031		A 9+=F11*
1032	ENDCAS	SE OPTION=1,LABEL=A,LAB=HERE
1033+	B	HERE
1034+A	EQU	* · · · · · · · · · · · · · · · · · · ·
1035 *		
1036 *		
1037	CASE	LABEL=B,REGA=14,OPRATOR=EQ,LITB=0
1038+	ST	10, SAVE10
1039+	ST	11, SAVE11
1040+	LR	10,14
1041+	LA	11,0
1042+	A	11,=F*0*
1043+	ÇR	10,11
1044*	BNE	86
1045+	8	C6
1046+86	L.	10, SAVE10
1047+	L	11,SAVE11
1048+	8	8 .
1049+06	L	10,SAVE10
1050+	L	11, SAVE11
1051		Å 4, ≃F <sup>‡</sup> 2 <sup>↓</sup>
1052		A 5,==F*2*
1053		A 6,=F*2*
		-
1054		A 7,=F*2*
1055		A 8,=F*2*
1056		4 9,=F*2*
1057	ENDCAS	E OPTION=1,LABEL=B,LAB=HERE
1058+	в	HERE
1059+8	EQU	*
1060 *		•
1061 *		
1062	CASE	LOCA=X,LOCB=Y,LABEL=C,OPRATOR=LL
1063+	ST	
	31	10, SAVE10

1054+ ST 11.SAVE11 1065+ L 10+X 1066+ ٤ 11,Y 1067 +CR 10,11 1068 \*, ILLEGAL OPERATOR SPECIFIED FOR A CASE CONDITION 1069+B8 L 10, SAVE10 1070+ ٢ 11, SAVEI1 1071 +8 С 1072+C8 L 10, SAVE10 1073+ L 11,SAVE11 1074 Α 4,=F\*3\* 1075 Á 5.=F'3' 1076 6,=F+3+ Δ 7,=F131 1077 A 1078 8,=F\*3\* A 1079 9,=F131 A 1080 ENDCASE OPTION=1,LABEL=C,LAB=HERE 1081+ В HERE 1082+C EQU \* 1083 \* 1084 \* 1085 EL SE 1086 A 4,=F14\* 1087 A 5,=F\*4\* 1088 6,=F+4+ A 1089 7.===41 A 1090 8,=F 4 A 9.=F+4 1091 Δ. 1092 ENDELSE LAB=HEPE 1093+HERE EQU \* 1094 \* 1095 \* 1096 \* 1097 \* 1098 SNAP ID=2,DCB=DUMPAREA,PDATA=REGS 1099+ CNOP 0,4 1100+ 84L 1. IHBOO12 BRANCH AROUND PARAM LIST 1101+ DC AL1(2) ID NUMBER 1102+ DC AL1(0) 1103+ DC ALI(130) OPTION FLAGS 1104+ DC ALI(32) OPTION FLAGS 1105+ DC ALDUMPAREA) DCB ADDRESS 1106+ 30 A(O) TCB ADDRESS 1107+ DC A(O) ADDRESS OF SNAP-SHOT LIST 1108+IH60012 DS 0H 1109+ SVC 51 1110 \* 1111 \* 1112 EXIT 1113+ L 13,4(,13) POP UP SAVE AREA 1114+ Ľ٩ 14,12,12(13) RESTORE REGISTERS 1115+ MVI 12(13),X'FF' FLAG EXIT 1116+ ₿₹ 14 RETURN 1117 CLOSE DUMPAREA 1118+ CNOP 0,4 ALIGN LIST TO FULLWORD 1119+ BAL 1,\*+8 LOAD REGI W/LIST ADDR 1120+ 0C ALI(128) OPTION BYTE AL3(DUMPAPEA) DCB ADDRESS 1121+ ЭС 1122+ SVC 20 ISSUE CLOSE SVC 1123 SAVEAREA DC 18A(0) 1124 DS 0F 1125 ZERO 00 1F:0! 1126 SAVEB 1 F D S 1127 SAVE9 05 1 F 1128 SAVELO DS 1F 1129 SAVE11 DS 1F

1130 POINTER DC 15.04 1131 STACK 0.5 100F 1132 X DC 1F+4\* 1133 Y 15481 00 1134 DUMPAREA DCB ODNAME=TEAM, DSORG=PS, RECEM=VBA, MACRF=W, BLKSIZE=882, LRECL=125 1136+\* DATA CONTROL BLOCK 1137+\* 1138+DUMPAREA DC OFTOT ORIGIN ON WORD BOUNDARY 1140+\* DIRECT ACCESS DEVICE INTERFACE 1142+ DC BL16'0' FDAD, DVT8L 1143+ DC A(O) KEYLE, DEVT, TRBAL 1145+\* COMMON ACCESS METHOD INTERFACE 1147+ 00 ALI(0) BUFNO 1148+ DC AL3(1) BUFCB 1149+ 00 AL2(0) BUFL 1150+ DC 8L2\*0100000000000000 bSORG 1151 +DC A(1) IOBAD 1153+\* FOUNDATION EXTENSION 1155+ DC BL1'0000000' BFTEK.BFLN.HIARCHY 1156+ DC ALB(1) ENDAD 1157+ 00 811101010100+ RECEM 1158+ DC AL3(0) EXLST 1160+\* FOUNDATION BLOCK 1162+ 0C CL8\*TEAM\* DDNAME 1163+ BL1\*00000010\* OFLGS DC 8L1'000000001 IFLG 1164+ ĐC 1165+ DC. BL2\*0000000000000000 MACR 1167+\* BSAM-BPAM-OSAM INTERFACE 1169+ 00 BL1:00000000 RER1 1170+ DC AU3(1) CHECK, GERR, PERR 1171+ DC A(1) SYNAD. 1172 +DC. H'0' CIND1, CIND2 DC 1173 +AL2(882) BLKSIZE 1174+ ÐĆ F'O' WCPD, WCPL, OFFSR, OFFSW 1175+ DC A(1) IOBA 1176+ DC ALITON NCP AL3(1) ECBR, EOBAD 1177\* 0C 1179+\* BSAM-BPAM INTERFACE 1181+ A(1) EOBW 0C H'O' DIRCT 1182+ DC 1183+ DC AL2(125) LRECL 1184+ DC A(1) CNTRL, NOTE, POINT 1185 END = F . 01 1136 = = 1 + 1 + 1187 1188 =F\*2\* 1189 =F131 1190 =F141

•

171

х

#### APPENDIX D

#### THE USE OF NESTED DO MACROS

# Sample Program M Illustrating Nested DO and ENDDO Macros

STMT	SOURCE	STATEMENT	
979	FRISBEE START	P <sup>®</sup> INT ENTER OPEN	NDGEN 12, SAVEAREA (DUMPAREA, OUTPUT)
985		LA	2 * 0
986		LA	3,0
987		LA ·	4,0
988		LA	5,0
939 990		LA	6,0
991		LA -	7,0
992		LA	8 • 0 9 • 0
993		LA	10,0
994		LA	11,0
-995		LA	
996		LA	15,0
997	*		1940
998			
999		SNAP	ID=1,DCB=DUMPAREA,PDATA=REGS
1011	*		
1012	*		
1013		00	LOWNUM=1, BYNUM=1, HGHNUM=5, DOLDOP=S, LABEL=LODPA
1075			A 4,=F*1*
1076			A 5,=F*1*
1077			Α 6,=F*1*
1078			SNAP ID=2,DCB=DUMPAREA,PDATA=REGS
1090	*		
1091		D0	WHILE=A,WREGA=8,WOP=LT,WLITB=10,LABEL=LDOPB
1107			A 8, #F*1*
$\frac{1108}{1109}$			A 9,=F11*
1116		ENDOD	
****	*	ENDDO	LABEL=LOOPB
1117	*	ENDDO	
1117	*	ENDDO	SNAP ID=3, DCB=DJMPAREA, PDATA=REGS
1129	*	ENDDO	SNAP ID=3,DCB=DJMPAREA,PDATA=REGS A 14,=F+1+
	*		SNAP ID=3,DCB=DJMPAREA,PDATA=REGS A 14,=F*1* A 15,=F*1*
1129 1130		ENDDO	SNAP ID=3,DCB=DJMPAREA,PDATA=REGS A 14,=F+1+
1129 1130 1131	×		SNAP ID=3,DCB=DJMPAREA,PDATA=REGS A 14,=F*1* A 15,=F*1*
1129 1130 1131 1138	×		SNAP ID=3,DCB=DJMPAREA,PDATA=REGS A 14,=F+10 A 15,=F*10 LABEL=LOOPA
1129 1130 1131 1138 1139	×	ENDOO	SNAP ID=3,DCB=DJMPAREA,PDATA=REGS A 14,=F*1* A 15,=F*1*
1129 1130 1131 1138 1139 1140 1153 1154	×	ENDOO	SNAP ID=3,DCB=DJMPAREA,PDATA=REGS A 14,=F*1* A 15,=F*1* LABEL=LOOPA LABEL=TESTA;REGA=4,OPRATOR=E0,REGB=15
1129 1130 1131 1138 1139 1140 1153 1154 1155	×	ENDOO	SNAP ID=3,DCB=DJMPAREA,PDATA=REGS A 14,=F+1+ A 15,=F+1+ LABEL=LOOPA LABEL=TESTA,REGA=4,DPRATOR=EQ,REGB=15 LA 4,0
1129 1130 1131 1138 1139 1140 1153 1154 1155 1156	×	ENDOO	SNAP ID=3,DCB=DJMPAREA,PDATA=REGS A 14,=F*1* A 15,=F*1* LABEL=LOOPA LABEL=TESTA;REGA=4,OPRATOR=EQ;REGB=15 LA 4,0 LA 5,0
1129 1130 1131 1138 1139 1140 1153 1154 1155 1156 1157	* *	ENDOO	SNAP       ID=3,DCB=DJMPAREA,PDATA=REGS         A       14,=F*1*         A       15,=F*1*         LABEL=LOOPA         LABEL=TESTA,REGA=4,OPRATOR=EQ,REGB=15         LA       4,0         LA       5,0         LA       6,0         LA       8,0         LA       9,0
1129 1130 1131 1138 1139 1140 1153 1154 1155 1156 1157 1158	* *	ENDOO	SNAP ID=3,DCB=DJMPAREA,PDATA=REGS A 14,=F+1+ A 15,=F+1+ LABEL=LOOPA LABEL=TESTA,REGA=4,OPRATOR=EQ,REGB=15 LA 4,0 LA 5,0 LA 6,0 LA 8,0 LA 9,0 LA 14,0
1129 1130 1131 1138 1139 1140 1153 1154 1155 1156 1157 1158 1159	* *	ENDOO	SNAP ID=3,DCB=DJMPAREA,PDATA=REGS A 14,=F*1* A 15,=F*1* LABEL=TESTA,REGA=4,DPRATOR=EQ,REGB=15 LA 4,0 LA 5,0 LA 6,0 LA 8,0 LA 9,0 LA 14,0 LA 15,0
1129 1130 1131 1138 1139 1140 1153 1154 1155 1156 1156 1157 1158 1159 1160	* *	ENDDO CASE	SNAP       ID=3,DCB=DJMPAREA,PDATA=REGS         A       14,=F*1*         A       15,=F*1*         LABEL=LOOPA         LABEL=TESTA,REGA=4,DPRATOR=E0,REGB=15         LA       4,0         LA       5,0         LA       6,0         LA       8,0         LA       9,0         LA       15,0         SNAP       ID=4,DCB=DJMPAREA,PDATA=REGS
1129 1130 1131 1138 1139 1140 1153 1154 1155 1156 1157 1158 1159 1160 1172	* *	ENDOO	SNAP ID=3,DCB=DJMPAREA,PDATA=REGS A 14,=F*1* A 15,=F*1* LABEL=TESTA,REGA=4,DPRATOR=EQ,REGB=15 LA 4,0 LA 5,0 LA 6,0 LA 8,0 LA 9,0 LA 14,0 LA 15,0
1129 1130 1131 1138 1139 1140 1153 1154 1155 1156 1156 1157 1158 1159 1160	*	ENDDO CASE	SNAP       ID=3,DCB=DJMPAREA,PDATA=REGS         A       14,=F*1*         A       15,=F*1*         LABEL=LOOPA         LABEL=TESTA,REGA=4,DPRATOR=E0,REGB=15         LA       4,0         LA       5,0         LA       6,0         LA       8,0         LA       9,0         LA       15,0         SNAP       ID=4,DCB=DJMPAREA,PDATA=REGS

1176 1190 1191 1192 1193 1194 1195 1207	L	CASE	LABEL=TESTB,LITA=0, OPRATOR=NE,REGB=12 LA 4,5 LA 5,5 LA 6,5 LA 8,5 LA 9,5 SNAP ID=5,DC3=DUMPAREA,PDATA=REGS
	<b>*</b> .	0.0	
1208 1268		DN	LOWREG=4, BYREG=4, HGHNUM=100, DCLOOP=G, LABEL=EEE
1269			A 14,≃≓*1* A 15,=F*1*
1209		ENDOG	LABEL=EEE
1277	*	ניטטאים	LACCLECTE
1278	<i>~</i>		SNAP ID=6.DCB=DUMPAREA.PDATA=REGS
1290	•	ENDCASE	LABEL=TESTB.OPTION=2
1292	ak	LIDUADE	CHOLCHICSIDSCRITCH-S .
1293			
1294		SNAP	ID=7,DCB=DUMPAREA,PDATA=REGS
1306	*		10. 13009-0000 AKERS 041 A-KE00
1307			
1308		EXIT	
1313		CLOSE	DUMPAREA
1319	SAVEAREA	D <b>C</b>	184(0)
1320		0 S	OF
1321	ZERO	00	1F*O*
1322	SAVE8	DS	16
	SAVE9	DS	1F
	SAVELO	ÐS	14
	SAVE11	DS	1F
•	POINTER	ÐC	1F*O*
	STACK	DS	100F
1328	DUMPAREA	-	DDNAME=TEAM, DSORG=PS, RECFM=V8A,
			RF=W,BLKSIZE=882,LRECL=125
1379		END	
1380		=F*	-
1381		= F •	-
1382		= = = =	•
1383		= F *	• -
1384 1385		====	100*
1393		- T	700.

REGS AT ENTRY TO SNAP

ID = 001

REGS 0-7	00000030 00000000	90018888 00000000	00000000 00000000	00000000
REGS 8-15	00000000 50018820	00000000 00018C20	00000000	00000000

END OF SNAP

173

X

REGS AT ENTRY TO	SNAP			ID = 002
REGS 0-7	0AS00000	40318994 00000001	00000000	00000000
REGS 8-15	00000000 5001B820	00000000 00018C20	00000000	00000000
END OF SNAP				•
REGS AT ENTRY TO	SNAP		•	ID = 003

			•	
REGS 0-7	000002A0	800189F8	00000000	00000000
	00000001	00000001	00000001	00000000
REGS 8-15	A000000A	A000000A	00000000	00000000
	50018820	00018C20	00000000	00000000
END OF SNAP				

FEGS AT ENTRY TO SNAP

ID = 002

00000000

REGS 0-7	00000240	A0018994	00000000	00000000
	0000002	0000002	00000002	00000000
<b>REGS 8-15</b>	0000004	A0000000	00000000	00000000
	50018820	00018620	00000001	00000001
END OF SNAP				

REGS AT ENTRY	TO SNAP			ID = 003
FEGS 0-7	000002A0 0000002	800189F8 0000002	00000000	00000000
REGS 8-15	0000000	A000000A	00000000	00000000

00018C20

00000001

5001B820

END OF SNAP

END OF SNAP		
· · · · · · · · · · · · · · · · · · ·	··· · · · ·	•
		•
REGS AT ENTRY TO SNAP	<i></i>	ID = 003

A0018994 0000000

00000003

A000000A

0000003

00000000

00018C20 0000002 00000001

REGS 0-7	000002A0 00000003	800189F8 00000003	00000000 0000003	000000000
REGS 8-15	0000000A 50018820	0000000A 00018C20	00000000	00000000
END DF SNAP			•	

REGS AT ENTRY TO SNAP

PEGS AT ENTRY TO SNAP

REGS 0-7

REGS 8-15

000002A0

A000000A

50018820

60000033

REGS	0-7	000002A0 00000004	A0018994 00000004	00000000 00000004	00000000
REGS	8-15	00000004 50018823	0000000A 00018C20	00000000 0000003	00000000 00000001

END OF SNAP

REGS AT ENTRY TO	SNAP			ID = 003
REGS 0-7	00000240	800189F8	00000000	0000000
	00000004	00000004	00000004	00000000
REGS 8-15	A000000	A0000000	00000000	00000000
<b>、</b>	50018820	00018020	0000003	00000000

END OF SNAP

ID = 002

ID = 002

00000000

00000000

FEGS AT ENTRY T	O SNAP ==			ID = 002
REGS 0-7	00000240	A0018994	00000000	00000000
	00000005	00000005	00000005	00000000
REGS 8-15	A000000A	A0000000	00000000	00000000
•	50018820	00018020	00000004	00000001
END OF SNAP				

. .

	800189F8	-	000 0	000000
1005	00000005			
		5 00000	005 (	00000000
000A	0000000	A 00000	000	000000
3820	0001BC2	0 00000	004 1	0000000
		3820 0001BC2	3820 0001BC20 00000	3820 0001BC20 0000004

REGS AT ENTRY TO SNAP ID = 005REGS 0-7 000002A0 90018ACC 00000000 00000000 00000005 00000005 00000005 00000000 REGS 8-15 00000005 0000,0005 00000000 00000000 50018820 00018620 00000005 00000001 END OF SNAP

REGS AT ENTRY TO SNAP REGS 0~7 000002A0 A001880C 00000000 00000005 00000005 00000005

RESS 8-15 0000005 0000005 0000000 00000000 50018820 00018C20 00000019 00000014 END OF SNAP

ID = 006

00000000

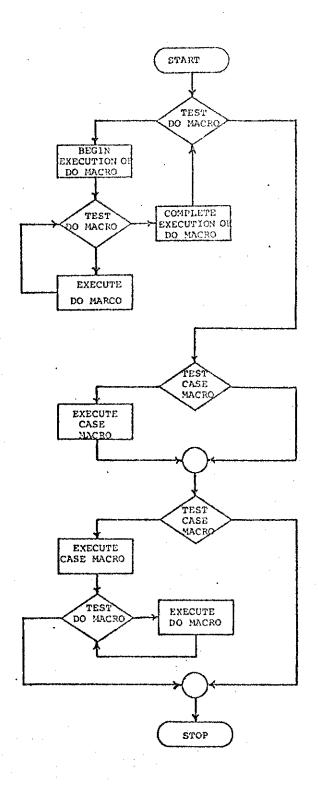


Fig. 27--Flowchart showing structured logic of Program M.

## PEGS AT ENTRY TO SNAP

ID = 007

PEGS 0-7	00000240	A00188F4	00000000	00000000
	00000005	00000005	00000005	00000000
REGS B-15	00000005	00000005	00000000	00000000
	50018820	00018620	0000019	00000000
END OF SNAP				

# Program M with Expanded Macros

### STMT SOURCE STATEMENT

oro chicade	-	
962 FRISBEE	ENTER	12, SAVEAREA
963+FRISBEE	DS	0H
964+	ENTRY	FRISBEE DECLARE NAME ENTRY
965+		*,12 DECLARE BASE AUDRESSIBILITY.
956+	BALR	15,0 (INITIAL ADDRESSIBILITY).
967*	8	12(,15) BRANCH AROUND ID FIELD
968+	DC	AL1(7), CL7'FRISBEE' ID LENGTH AND ID
969+	BCTR	15.0 (RESET INITIAL ADDRESSIBILITY
970+	8CTR	15.0 ABSOLUTE ENTRY POINT).
971+	STM	14,12,12(13) SAVE REGISTERS
972+	LR	12,15 SETUP BASE PEGISTER.
973+	\$T	13,SAVEAREA+4 CHAIN BACK
974+	LA	O, SAVEAREA CHAIN FORWARD
975+	ST	0,8(0,13)
976+	LR	13,0 SET UP SAVE AREA POINTER
977+		SAVEAREA, 13 AND ADDRESSABILITY
978 STARY	OPEN	(DUMPAREA, OUTPUT)
979+	CNOP	0,4 ALIGN LIST TO FULLWORD
980+START	BAL	1.*+8 LOAD PEGI W/LIST ADDR.
981+	DC	ALI(143) OPTION BYTE
982+	00	AL3(DUMPAREA) DCB ADDRESS
983+	SVC	19 ISSUE OPEN SVC
984 985	LA	2,0
986		3,0
987	LA LA	4,0
988	LA	5+0
989	LA	6,0 7,0
990	LA	8:0
991	LA	9.0
992	LA	10,0
993	LA	11.0
994	LA	14,0
995	LA	15.0
996 *		
997 *		
998	SVAP	ID=1,DC8=DUMPAREA;PDATA=REGS
999*	CNOP	0,4
1000+	BAL	1, IHBOOO3 BRANCH ARDUND PARAM LIST
1001+	00	ALI(1) IO NUMBER
1002+	DC	ALI(O)
1003+	DC .	ALI(130) OPTION FLAGS
1004+	DC	ALI(32) OPTION FLAGS
1005+	DC	ALDUMPAREAT DCB ADDRESS
1006+ 1007+	DC DC	A(0) TCB ADDRESS
TAALA	DC	A(O) ADDRESS OF SNAP-SHOT LIST

1009+ St 51 $1011 * CDMUM=1, BYNUM=1, HGMNM=5, DDLCOP=S, LASEL=LCOPA 1011 * CDMUM=1, BYNUM=1, HGMNM=5, DDLCOP=S, LASEL=LCOPA 1014 * ST 9, SAVE9 1014 * ST 10, SAVE10 1016 * ST 11, SAVE11 1017 * LA 10, 0 1018 * A 10, FF11 1019 * LA 1, 0 1020 * A 11, FF5 1021 * LA 9, 0 1022 * LA 9, FOITER 1022 * LA 9, FOITER 1022 * LA 9, CONTER 1023 * LA 9, CONTER 1024 * LA 9, CONTER 1025 * LA 9, CONTER 1026 * ST 11, STACK (a) 1027 * LA 8, CONTER 1028 * ST 11, STACK (a) 1029 * LA 9, CONTER 1031 * L 11, STACK (a) 1029 * LA 9, CONTER 1031 * L 11, STACK (a) 1031 * ST 9, SAVE9 1033 * ST 9, SAVE9 1034 * ST 9, SAVE9 1034 * ST 9, SAVE9 1035 * CONTER 1036 * ST 9, SAVE10 1038 * ST 11, SAVE11 1039 * L 11, STACK (a) 1044 * L 11, STACK (a) 1044 * S 8, FF14 10404 * S 8, FF14 10404 * C 9, ZEND 1044 * L 11, STACK (a) 1045 * L 9, SAVE10 1045 * L 9, SAVE10 1045 * L 9, SAVE10 1046 * ST 9, SAVE9 1046 * ST 8, FOINTER 1047 * C 9, ZEND 1046 * ST 8, FOINTER 1047 * ST 10, SIXCK 101 1050 * ST 11, SAVE11 1050 * ST 9, SAVE9 1050 * ST 9, SAVE9 1050 * ST$	1008+1HB0003	DS	он			
1011 *         COMUNAL, BYNUMAL, KGHNUMAS, ODLCOPAS, LABEL=LCOPA           1013+         ST         R, SAVER           1014+         ST         R, SAVER           1015+         ST         10, SAVE10           1016+         ST         11, SAVE11           1017+         LA         10, 0           1018+         A         10, -FF11           1017+         LA         10, 0           1018+         A         10, -FF11           1017+         LA         10, 0           1021+         LA         9, 0           1022+         A         9, 0F11           1023+         L         8, 07117ER           1024+         LA         9, 0           1025+         ST         11, STACK(8)           1027+         LA         8, 4(81           1027+         LA         8, 4(81           1027+         LA         8, 4(81           1032+         LA         8, 4(81           1033+         L         10, SAVE10           1033+         L         10, SAVE10           1033+         L         10, SAVE10           1033+         L         10, SAVE10 <tr< th=""><th></th><th>216</th><th>51</th><th></th><th></th><th></th></tr<>		216	51			
$ \begin{array}{cccccccccccccccccccccccccccccccccccc$						
$ \begin{aligned}   0   1 + & ST & 0 & SAVE0 \\   0   1 + & ST & 0 & SAVE0 \\   0   1 + & ST & 0 & SAVE0 \\   0   1 + & I & 1 & SAVE1 \\   0   1 + & I & 1 & 0 & . \\   0   1 + & I & 1 & 0 & . \\   0   1 + & I & I & 0 & . \\   0   1 + & I & I & 1 & 0 & . \\   0   1 + & I & I & 1 & 0 & . \\   0   1 + & I & I & 1 & 0 & . \\   0   1 + & I & I & 0 & . \\   0   1 + & I & I & 0 & . \\   0   1 + & I & I & 0 & . \\   0   1 + & I & I & 0 & . \\   0   1 + & I & I & 0 & . \\   0   1 + & I & I & 0 & . \\   0   1 + & I & 0 $		רח	LOWNUM=1+BYN	UM=1.HGHNUM=5.DOL	COP=S,LASEL=	LODPA
$ \begin{aligned} 101+ & ST & 0, SAVE0 \\ 1016+ & ST & 11, SAVE11 \\ 1017+ & LA & 10, SAVE11 \\ 1017+ & LA & 10, SAVE11 \\ 1019+ & LA & 11, SAVE11 \\ 1021+ & LA & 9, STORE \\ 1022+ & A & 9, STORE \\ 1022+ & LA & 9, STORE \\ 1022+ & LA & 9, SAVE11 \\ 1024+ & ST & 9, STACK(8) \\ 1027+ & LA & 8, 4(8) \\ 1030+ & ST & 9, 2014TER \\ 1030+ & L & 9, SAVE11 \\ 1030+ & L & 9, SAVE11 \\ 1033+ & L & 9, SAVE11 \\ 1033+ & L & 9, SAVE11 \\ 1033+ & ST & 10, SAVE11 \\ 1033+ & ST & 10, SAVE11 \\ 1033+ & L & 11, SAVE11 \\ 1039+ & L & 11, SAVE11 \\ 1039+ & L & 11, SAVE11 \\ 1039+ & L & 11, SAVE11 \\ 1044+ & S & 8, SF + 4 \\ 1045+ & L & 10, STACK(8) \\ 1046+ & S & 8, SF + 4 \\ 1045+ & L & 10, STACK(8) \\ 1046+ & ST & 9, SAVE0 \\ 1046+ & ST & 9, SAVE0 \\ 1047+ & C & 9, ZER0 \\ 1046+ & ST & 0, SAVE10 \\ 1050+ & BL & X4 \\ 1049+ & C & 10, L1 \\ 1053+ & BL & Y4 \\ 1051+ & B & Z4 \\ 1049+ & C & 10, L1 \\ 1055+ & L & 8, PT14TER \\ 1047+ & C & 9, ZER0 \\ 1046+ & ST & 0, ST & 0, ST \\ 1055+ & L & 8, PT14TER \\ 1047+ & C & 9, ZER0 \\ 1046+ & ST & 0, ST & 0, ST & 0, ST \\ 1055+ & L & 8, PT14TER \\ 1047+ & C & 9, ZER0 \\ 1055+ & L & 8, PT14TER \\ 1047+ & C & 9, ZER0 \\ 1055+ & L & 8, PT14TER \\ 1047+ & C & 9, ST & CX(8) \\ 1055+ & L & 8, PT14TER \\ 1055+ & L & 9, ST & CX(8) \\ 1055+ & L & 9, ST & CX(8) \\ 1055+ & L & 9, ST & CX(8) \\ 1055+ & L & 9, ST & CX(8) \\ 1055+ & L & 9, ST & CX(8) \\ 1055+ & L & 9, ST & CX(8) \\ 1055+ & L & 9, SAYE9 \\ 1075+ & L & 10, SAVE11 \\ 1075+ & B & K4 \\ 1055+4 & EU & 0 \\ 1075+4 & EU \\ 1075+4 & EU & 0 \\ 1075+4 & EU & 0 \\ 1075+4 & EU \\ 1075+4 & EU \\ 1075+4 & EU \\ 105+54 & E$						
1016*ST11,53/41017*LA10,01018*A10,0,01023*LA11,01023*LA11,0,01024*ST9,000000000000000000000000000000000000	1014*		9,SAVE9			
$ \begin{bmatrix} 0 17 + \\ 0 19 + \\ 10 19 + \\ 14 \\ 10 19 + \\ 14 \\ 11 + 6 \\ 10 20 + \\ 14 \\ 10 20 + \\ 14 \\ 10 21 + \\ 12 \\ 10 22 + \\$	1015+	ST	10,SAVE10			
1013+       A $10, -p + 1i$ 1020+       A $11, -p + 5i$ 1021+       LA $9, -p + 1i$ 1022+       A $9, -p + 1i$ 1022+       L $8, -p + 1i$ 1023+       L $8, -p + 1i$ 1024+       ST $9, -p + 1i$ 1025+       LA $8, -4(8)$ 1027+       LA $8, -4(8)$ 1027+       LA $8, -4(8)$ 1027+       LA $8, -4(8)$ 1027+       LA $8, -4(8)$ 1037+       L $10, -5A + C(8)$ 1033+       L $11, -5A + C(8)$ 1033+       ST $12, -5A + C(8)$ 1033+       ST $11, -5A + C(8)$ 1033+       ST $11, -5A + C(8)$ 1034+       S $8, -p + 14$ 1039+       ST $11, -5A + C(8)$ 1042+       S $8, -p + 14$ 1044+ <td< th=""><th>1016+</th><th>ST</th><th>11,SAVE11</th><th></th><th></th><th></th></td<>	1016+	ST	11,SAVE11			
$\begin{array}{llllllllllllllllllllllllllllllllllll$						
$\begin{array}{llllllllllllllllllllllllllllllllllll$						
$\begin{array}{llllllllllllllllllllllllllllllllllll$						
$\begin{array}{llllllllllllllllllllllllllllllllllll$						
$\begin{array}{llllllllllllllllllllllllllllllllllll$				-		
$\begin{array}{rrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrr$						
$\begin{array}{llllllllllllllllllllllllllllllllllll$						
1027 + LA = 8,4(8) $1028 + ST = 11,5TACK(8)$ $1030 + ST = 9,50'ATER$ $1031 + L = 8,5A'EB$ $1032 + L = 9,5A'EB$ $1033 + L = 10,5A'EB$ $1033 + L = 10,5A'EB$ $1033 + L = 11,5A'EB$ $1035 + LOOPA = ST = 9,5A'EB$ $1036 + ST = 9,5A'EB$ $1036 + ST = 11,5A'EB$ $1036 + ST = 11,5A'EB$ $1037 + ST = 10,5A'EB$ $1037 + ST = 10,5A'EB$ $1038 + ST = 11,5A'EB$ $1040 + S = 8,=E'+4$ $1041 + L = 11,5TACK(8)$ $1042 + S = 8,=E'+4$ $1043 + L = 9,5TACK(8)$ $1046 + ST = 9,7ACK(8)$ $1046 + ST = 9,7ACK(8)$ $1046 + ST = 0,7E'A'$ $1046 + ST = 9,7ACK(8)$ $1046 + ST = 9,7ACK(8)$ $1046 + ST = 9,7ACK(8)$ $1046 + ST = 0,7E'A'$ $1046 + ST = 0,7E'A'$ $1046 + ST = 0,7E'A'$ $1056 + ST = 9,7ACK(8)$ $1057 + L = 8,4(3)$ $1056 + ST = 9,5TACK(8)$ $1057 + L = 8,4(3)$ $1057 + L = 8,4(3)$ $1056 + ST = 9,5TACK(8)$ $1056 + L = 8,5VFB$ $1056 + L = 9,5VFB$ $1056 + L = 10,5VFB$ $1056 + L = 10,5$	1025+	LA				
$\begin{array}{llllllllllllllllllllllllllllllllllll$	1026+	ST	10,STACK(8)			
$\begin{array}{llllllllllllllllllllllllllllllllllll$			-	· · ·		
1030+ST $\theta_1$ POTNTER1031+L $\theta_1$ SAVE91032+L $\theta_1$ SAVE91033+L $10, SAVE10$ 1033+L $10, SAVE10$ 1035+ST $\theta_1$ SAVE91037+ST $10, SAVE10$ 1038+ST $11, SAVE11$ 1039+L $\theta_1$ POTNTER1040+S $\theta_1 = 14$ (SAVE1)1041+L $11, STACK(8)$ 1042+S $\theta_2 = 14$ (SAVE1)1043+L $10, STACK(8)$ 1044+S $\theta_2 = 14$ (SAVE1)1045+L $\theta_2 STACK(8)$ 1044+S $\theta_2 = 14$ (SAVE1)1045+L $\theta_3 STACK(8)$ 1046+ST $\theta_1 POTNTER$ 1047+C $9, ZERO$ 1048+BX41051+BZ41052+X4CR $10, 11$ 1053+BL $Y4$ 1055+L $\theta_1 POTNTER$ 1055+ST $9, STACK(8)$ 1056+ST $9, STACK(8)$ 1057+LA $0, 4(3)$ 1058+ST $10, STACK(8)$ 1062+ST $1, STACK(8)$ 1064+L $0, 4(4)$ 1059+LA $0, 4(3)$ 1059+LA $0, 4(3)$ 1060+L $0, SAVE9$ 1060+L $0, SAVE9$ 1063+L $0, SAVE9$ 1064+L $0, SAVE10$ 1064+L $0, SAVE10$ 1064+L $0, SAVE$			•			
1031+L $\theta_1 SAVEB$ 1032+L $\theta_1 SAVED$ 1033+L $10 SAVED$ 1033+L $10 SAVED$ 1035+ST $\theta_1 SAVED$ 1036+ST $\theta_1 SAVED$ 1037+ST $10 SAVED$ 1038+ST $11 SAVED$ 1039+L $\theta_1 POTNTEP$ 1041+L $11 SAVED$ 1042+S $\theta_2 = 144$ 1041+L $0 STACK(B)$ 1042+S $\theta_2 = 144$ 1043+L $0 STACK(B)$ 1044+S $\theta_2 = 144$ 1045+L $0 STACK(B)$ 1046+ST $\theta_1 POTNTER$ 1047+C $9 ZERD$ 1048+BLX41050+BHY41051+ $\delta$ Z41052+XCR $10,111$ 1053+BLY41055+L $\theta_1 POTNTER$ 1056+ST $9 STACK(B)$ 1057+LA $\theta_1 (G)$ 1059+LA $\theta_1 (G)$ 1051+ $\delta$ $24$ 1052+X4CR $10,111$ 1053+BL $Y4$ 1055+L $\theta_1 (G)$ 1056+L $\theta_1 (G)$ 1051+B $4(G)$ 1052+L $\theta_1 (G)$ 1054+L $\theta_1 (G)$ 1055+L $\theta_1 (G)$		_	•	•		
$\begin{array}{llllllllllllllllllllllllllllllllllll$						
$\begin{array}{llllllllllllllllllllllllllllllllllll$						
1034+L11, SAVE111035+LOOPAST0, SAVE91036+ST10, SAVE91037+ST10, SAVE101039+L0, SAVE111039+L0, SAVE111040+S0, = $\varepsilon^+ 4^+$ 1041+L11, STACK (8)1042+S0, = $\varepsilon^+ 4^+$ 1043+L10, STACK (8)1044+S0, = $\varepsilon^+ 4^+$ 1045+L9, STACK (8)1046+ST8, POINTER1046+ST8, POINTER1046+ST8, POINTER1047+C9, ZERO1048+BLX41049+CR10, 111050+BHY41051+BZ41055+L8, POINTER1056+ST9, STACK (8)1057+L8, 4(3)1059+L8, 4(3)1059+L8, 4(3)1059+L8, 4(3)1060+ST11, STACK (8)1061+L9, SAVE91064+L9, SAVE91064+L9, SAVE91064+L9, SAVE91064+L14, SAVE111067+BH41068+Y4L8, SAVE81069+L9, SAVE91070+L10, SAVE101071+L11, SAVE111072+SALOPA1073+SALOPA						
$\begin{array}{llllllllllllllllllllllllllllllllllll$		_	-			
1037+ST10, SAVE101038+ST11, SAVE111039+L8, PSTVATEP1040+S8, PSTVATEP1041+L11, STACK(8)1042+S8, PSTVATEP1043+L10, STACK(8)1044+S8, PSTVATEP1044+S8, PSTVATEP1044+S8, PSTVATEP1044+S8, PSTVATEP1044+S9, STACK(8)1044+S8, PSTVATEP1044+S9, STACK(8)1044+S8, PSTVATEP1047+C9, ZERO1048+BLX41050+BHY41051+6241055+L8, POLVITER1055+L8, POLVITER1055+L8, POLVITER1056+ST9, STACK(8)1059+L8, POLVITER1050+ST10, STACK(8)1050+ST10, STACK(8)1055+L8, POLVITER1056+ST9, STACK(8)1061+L8, SAVEB1062+ST8, POLVITER1063+L9, SAVE91063+L10, SAVE101064+L9, SAVEB1064+L9, SAVEB1064+L9, SAVEB1064+L9, SAVEB1064+L9, SAVEB1064+L9, SAVEB1065+L10, SAVE101071+ <th></th> <th></th> <th></th> <th>2* </th> <th>•</th> <th></th>				2* 	•	
1038+ST $11, SAVE 1$ $1039+$ L $8, PEY 4$ $1040+$ S $8, FY 4$ $1041+$ L $11, STACK(8)$ $1042+$ S $8, FY 4$ $1043+$ L $10, STACK(8)$ $1044+$ S $9, FY 4$ $1045+$ L $9, STACK(8)$ $1044+$ S $9, FY 4$ $1045+$ L $9, STACK(8)$ $1046+$ ST $8, POINTER$ $1047+$ C $9, ZERO$ $1048+$ BLX4 $1049+$ CR $10, 11$ $1050+$ BHY4 $1051+$ BZ4 $1054+$ R $10, 9$ $1055+$ L $8, POINTER$ $1056+$ ST $9, STACK(8)$ $1057+$ L $8, POINTER$ $1058+$ ST $10, STACK(8)$ $1059+$ L $8, 4(8)$ $1060+$ ST $11, STACK(8)$ $1063+$ ST $10, STACK(8)$ $1064+$ L $8, SAVE8$ $1064+$ L $8, SAVE8$ $1065+$ L $10, SAVE10$ $1065+$ L $10, SAVE10$ $1065+$ L $10, SAVE10$ $1068+Y4$ L $8, SAVE8$ $1069+$ L $9, SAVE9$ $1069+$ L $11, SAVE11$ $1071+$ L $11, SAVE11$ $1072+$ B $ALOOPA$	1036+	ST	9, SAVE9			
1039+L8,P01 NTEP1040+S8,=5*4*1041+L11,STACK(8)1042+S8,=5*4*1043*L10,STACK(8)1044*S8,=5*4*1045*L9,STACK(8)1046*ST8,P01NTER1046*ST8,P01NTER1047*C9,ZERO1048*BLX41050+BHY41051+BZ41053+L8,P01NTER1055*L8,P01NTER1055*L8,P01NTER1055*L8,P01NTER1055*L8,P01NTER1056*ST9,STACK(8)1057*LA8,4(3)1058*ST10,STACK(8)1064*ST1,STACK(8)1065*L8,AVE81064*L8,AVE81064*L9,SAVE91065*L10,SVE101064*L9,SAVE91065*L10,SVE101064*L9,SAVE91065*L10,SAVE101070*L10,SAVE101071*L11,SAVE111072*BALOPPA1073*EQU*						
$\begin{array}{llllllllllllllllllllllllllllllllllll$						
$1041+$ L $11, STACK(8)$ $1042+$ S $9, F*4^+$ $1043+$ L $10, STACK(8)$ $1044+$ S $8, F*4^+$ $1045+$ L $9, STACK(8)$ $1046+$ ST $8, POINTER$ $1047+$ C $9, ZERO$ $1048+$ BLX4 $1049+$ CR $10, 11$ $1050+$ BHY4 $1051+$ $8, Z4$ $1052+X4$ CR $10, 11$ $1053+$ BLY4 $1055+$ L $8, POINTER$ $1056+$ ST $9, STACK(8)$ $1057+$ L $8, 4(3)$ $1057+$ LA $8, 4(3)$ $1059+$ LA $8, 4(3)$ $1065+$ ST $8, POINTER$ $1065+$ ST $9, STACK(8)$ $1057+$ LA $8, 4(3)$ $1057+$ LA $8, 4(3)$ $1062+$ ST $8, POINTER$ $1063+$ L $9, SAVE8$ $1064+$ L $9, SAVE8$ $1064+$ L $9, SAVE8$ $1067+$ B $8/4$ $1068+Y4-$ L $8, SAVE8$ $1089+$ L $9, SAVE10$ $1071+$ L $11, SAVE11$ $1072+W4$ EQU*			-			
$\begin{array}{llllllllllllllllllllllllllllllllllll$	-		-			
1043+L10, STACK(8) $1044+$ SS, =F+4+ $1045+$ L9, STACK(8) $1046+$ ST8, POINTER $1047+$ C9, ZERO $1048+$ BLX4 $1049+$ CR10, 11 $1050+$ BHY4 $1051+$ 8Z4 $1052+X4$ CR10, 11 $1053+$ BLY4 $1055+$ L8, POINTER $1055+$ L8, POINTER $1055+$ L8, POINTER $1055+$ L8, POINTER $1056+$ ST9, STACK(8) $1057+$ LA8, 4(3) $1059+$ LA8, 4(3) $1062+$ ST8, POINTER $1063+$ L8, SAVE8 $1063+$ L8, SAVE8 $1064+$ L10, SAVF10 $1067+$ BW4 $1068+Y4-$ L8, SAVE8 $1069+$ L9, SAVE9 $1067+$ BW4 $1068+Y4-$ L8, SAVE8 $1069+$ L9, SAVE9 $1070+$ L10, SAVE10 $1071+$ L11, SAVE11 $1072+$ SALOPA $1072+$ SALOPA $1072+$ SALOPA $1072+$ SALOPA		с С	-			·
$1044+$ S $B_{+}=P+4^{+}$ $1045+$ L9,STACK(8) $1046+$ ST8,POINTER $1047+$ C9,ZERO $1048+$ BLX4 $1049+$ CR $10,11$ $1050+$ BHY4 $1051+$ BZ4 $1052+X4$ CR $10,11$ $1053+$ BLY4 $1055+$ L $8,POINTER$ $1055+$ L $8,POINTER$ $1055+$ L $8,POINTER$ $1055+$ L $8,POINTER$ $1056+$ ST $9,STACK(8)$ $1057+$ LA $8,4(3)$ $1059+$ LA $8,4(3)$ $1060+$ ST $11,STACK(8)$ $1060+$ ST $11,STACK(8)$ $1063+$ L $9,SAVE8$ $1063+$ L $9,SAVE8$ $1064+$ L $9,SAVE8$ $1067+$ BW4 $1068+Y4$ L $8,SAVE8$ $1069+$ L $10,SAVE10$ $1070+$ L $10,SAVE10$ $1070+$ L $10,SAVE10$ $1072+W4$ EQU*						
1046+       ST       8,PDINTER         1047+       C       9,ZERD         1048+       BL       X4         1049+       CR       10,11         1050+       BH       Y4         1051+       8       Z4         1052*X4       CR       10,11         1053+       BL       Y4         1055*X4       CR       10,11         1053+       BL       Y4         1055*X4       CR       10,71         1055+       L       8,PDINTER         1055+       L       8,PDINTER         1055+       L       8,YACK(8)         1057+       LA       8,4(8)         1058+       ST       10,STACK(8)         1059+       LA       8,4(8)         1060+       ST       11,STACK(8)         1061+       LA       8,4(8)         1062+       ST       8,POIVTER         1063+       L       8,SAVE8         1063+       L       9,SAVE9         1065+       L       10,SAVF10         1067+       B       H4         1059+       L       9,SAVE9         1070+						
1047+       C       9, ZER0         1048+       BL       X4         1049+       CR       10,11         1050+       BH       Y4         1051+       8       Z4         1052*X4       CR       10,11         1053+       BL       Y4         1055+       L       8, POINTER         1055+       L       8, POINTER         1056+       ST       9, STACK (8)         1057+       LA       8, 4(8)         1058+       ST       10, STACK (8)         1059+       LA       8, 4(8)         1059+       LA       8, 4(8)         1059+       LA       8, 4(8)         1060+       ST       11, STACK (8)         1061+       LA       8, 4(8)         1062+       ST       8, POINTER         1063+       L       8, SAVE8         1063+       L       9, SAVE9         1063+       L       10, SAVE10         1066+       L       11, SAVE8         1059+       L       9, SAVE9         1070+       L       10, SAVE10         1072+       B       ALO0PA <t< th=""><th>1045+</th><th></th><th>9, STACK (8)</th><th></th><th></th><th></th></t<>	1045+		9, STACK (8)			
1048+       BL       X4         1049+       CR       10,11         1050+       BH       Y4         1051+       B       Z4         1052+X4       CR       10,11         1053+       BL       Y4         1054+Z4       AR       10,9         1055+       L       8,POINTER         1056+       ST       9,STACK(8)         1057+       LA       8,4(3)         1056+       ST       9,STACK(8)         1057+       LA       8,4(3)         1058+       ST       10,STACK(8)         1059+       LA       8,4(8)         1060+       ST       11,STACK(8)         1061+       LA       8,4(8)         1062+       ST       8,POINTER         1063+       L       8,SAVE8         1063+       L       9,SAVE9         1066+       L       11,SAVE11         1067+       B       W4         1068+Y4       L       8,SAVE9         1070+       L       10,SAVE10         1070+       L       11,SAVE11         1072+       B       AL000PA         1073+W						
1049+       CR       10,11         1050+       BH       Y4         1051+       B       Z4         1052+X4       CR       10,11         1053+       BL       Y4         1055+       L       8,P01MTER         1055+       L       8,P01MTER         1055+       L       8,P01MTER         1056+       ST       9,STACK(8)         1057+       LA       8,4(3)         1058+       ST       10,STACK(8)         1059+       LA       8,4(8)         1050+       ST       11,STACK(8)         1061+       LA       8,4(8)         1062+       ST       8,P01MTER         1062+       ST       8,P01MTER         1063+       L       9,SAVE9         1064+       L       9,SAVE9         1065+       L       10,SAVF10         1066+       L       11,SAVE11         1067+       B       W4         1068+Y4       L       8,SAVE8         1059+       L       9,SAVE9         1070+       L       10,SAVE10         1071+       L       11,SAVE11         <		-	• • •			
1050+       BH       Y4         1051+       B       Z4         1052+X4       CR       10,11         1033+       BL       Y4         1055+       L       8,P01NTER         1056+       ST       9,STACK(8)         1053+       L       8,P01NTER         1056+       ST       9,STACK(8)         1057+       LA       8,4(8)         1059+       LA       8,4(8)         1060+       ST       11,STACK(8)         1061+       LA       8,4(8)         1062+       ST       8,P01NTER         1063+       L       8,SAVE8         1064+       L       9,SAVE9         1065+       L       10,SAVF10         1066+       L       11,SAVE11         1067+       B       W4         1068+Y4       L       8,SAVE8         1069+       L       9,SAVE9         1070+       L       10,SAVE10         1071+       L       11,SAVE11         1072+       B       ALODPA         1073+W4+       EQU       *						
1051+       8       Z4         1052+X4       CR       10,11         1053+       8L       Y4         1053+       8L       Y4         1055+       L       8,POINTER         1056+       ST       9,STACK(8)         1057+       LA       8,4(3)         1059+       LA       8,4(3)         1060+       ST       10,STACK(8)         1061+       LA       8,4(3)         1062+       ST       8,POINTER         1063+       L       8,SAVE8         1064+       LA       8,4(3)         1062+       ST       8,POINTER         1063+       L       8,SAVE8         1064+       L       9,SAVE9         1065+       L       10,SAVF10         1066+       L       11,SAVE11         1066+       L       11,SAVE11         1066+       L       9,SAVE9         1059+       L       8,SAVE8         1065+       L       9,SAVE9         1067+       B       W4         1068+Y4       L       8,SAVE8         1059+       L       9,SAVE9         1072+ </th <th></th> <th></th> <th></th> <th></th> <th></th> <th></th>						
1052+X4       CR       10,11         1053+       BL       Y4         1054+Z4       AR       10,9         1055+       L       8,P01MTER         1056+       ST       9,STACK(8)         1057+       LA       8,4(3)         1058+       ST       10,STACK(8)         1059+       LA       8,4(8)         1060+       ST       11,STACK(8)         1061+       LA       8,4(3)         1062+       ST       8,P01MTER         1063+       L       8,SAVE8         1063+       L       9,SAVE9         1064+       L       9,SAVE9         1065+       L       10,SAVE10         1067+       B       W4         1068+Y4       L       8,SAVE8         1069+       L       9,SAVE9         1070+       L       10,SAVE10         1070+       L       10,SAVE10         1071+       L       11,SAVE11         1072+       B       ALOOPA         1073+W4-       EQU       *						
1053+       BL       Y4         1054+Z4       AR       10,9         1055+       L       8,P0INTER         1056+       ST       9,STACK(8)         1057+       LA       8,4(3)         1053+       ST       10,STACK(8)         1059+       LA       8,4(8)         1060+       ST       11,STACK(8)         1061+       LA       8,4(8)         1062+       ST       8,P0INTER         1063+       L       8,SAVE8         1064+       L       9,SAVE9         1065+       L       10,SAVF10         1066+       L       11,SAVE11         1067+       B       W4         1068+Y4       L       8,SAVE8         1069+       L       9,SAVE9         1070+       L       10,SAVE10         1071+       L       11,SAVE11         1072+       B       ALODPA         1073+W4       EQU       *						
1055+       L       8,PDINTER         1056+       ST       9,STACK(8)         1057+       LA       8,4(3)         1053+       ST       10,STACK(8)         1059+       LA       8,4(8)         1061+       LA       8,4(3)         1062+       ST       8,PDINTER         1063+       L       8,SAVE8         1063+       L       8,SAVE8         1064+       L       9,SAVE9         1065+       L       10,SAVF10         1066+       L       11,SAVE11         1067+       B       W4         1068+Y4       L       8,SAVE8         1069+       L       10,SAVE9         1070+       L       10,SAVE10         1071+       L       11,SAVE11         1072+       B       ALODPA         1073+W4       EQU       *	1053+					
1056+       ST       9,STACK(8)         1057+       LA       8,4(3)         1059+       ST       10,STACK(8)         1059+       LA       8,4(8)         1060+       ST       11,STACK(8)         1061+       LA       8,4(3)         1062+       ST       8,POINTER         1063+       L       8,SAVE8         1064+       L       9,SAVE9         1065+       L       10,SAVF10         1066+       L       11,SAVE11         1068+Y4       L       8,SAVE8         1069+       L       9,SAVE9         1059+       L       10,SAVF10         1066+       L       11,SAVE11         1068+Y4       L       8,SAVE8         1059+       L       9,SAVE9         1070+       L       10,SAVE10         1071+       L       11,SAVE11         1072+       B       ALOOPA         1073+W4-       EQU       *		AR				
1057+       LA       8,4(3)         1053+       ST       10,STACK(3)         1059+       LA       8,4(8)         1060+       ST       11,STACK(8)         1061+       LA       8,4(3)         1062+       ST       8,POINTER         1063+       L       8,SAVE8         1064+       L       9,SAVE9         1065+       L       10,SAVF10         1066+       L       11,SAVE11         1067+       B       W4         1068+Y4       L       8,SAVE8         1059+       L       9,SAVE9         1070+       L       10,SAVE10         1071+       L       11,SAVE11         1072+       B       ALOOPA         1073+W4       EQU       *						
1053+       ST       10,STACK(3)         1059+       LA       8,4(8)         1060+       ST       11,STACK(8)         1061+       LA       8,4(3)         1062+       ST       8,POINTER         1063+       L       8,SAVE8         1064+       L       9,SAVE9         1065+       L       10,SAVF10         1066+       L       11,SAVE11         1067+       B       W4         1059+       L       8,SAVE8         1059+       L       9,SAVE9         1070+       L       10,SAVE10         1071+       L       11,SAVE11         1072+       B       ALOOPA         1073+W4+       EQU       *						
1059+       LA       8,4(8)         1060+       ST       11,STACK(8)         1061+       LA       8,4(3)         1062+       ST       8,POINTER         1063+       L       8,SAVE8         1064+       L       9,SAVE9         1065+       L       10,SAVF10         1066+       L       11,SAVE11         1067+       B       W4         1068+Y4       L       8,SAVE8         1070+       L       0,SAVE10         1071+       L       11,SAVE11         1072+       B       ALOOPA         1073+W4       EQU       *			•			
1050+       ST       11,STACK(8)         1061+       LA       8,4(3)         1062+       ST       8,POINTER         1063+       L       8,SAVE8         1064+       L       9,SAVE9         1065+       L       10,SAVF10         1066+       L       11,SAVE11         1067+       B       W4         1068+Y4       L       8,SAVE8         1059+       L       9,SAVE9         1070+       L       10,SAVE10         1071+       L       11,SAVE11         1072+       B       ALOOPA         1073+W4       EQU       *						
1061*       LA       8,4(3)         1062*       ST       8,POINTER         1063*       L       8,SAVE8         1064*       L       9,SAVE9         1065*       L       10,SAVF10         1066*       L       11,SAVE11         1067*       B       W4         1068*Y4       L       8,SAVE8         1070*       L       9,SAVE9         1071*       L       10,SAVE10         1071*       L       11,SAVE11         1072*       B       ALOOPA         1073*W4       EQU       *			•			
1062+       ST       8,POINTER         1063+       L       8,SAVE8         1064+       L       9,SAVE9         1065+       L       10,SAVE10         1066+       L       11,SAVE11         1067+       B       W4         1068+Y4       L       8,SAVE8         1070+       L       9,SAVE9         1070+       L       10,SAVE10         1071+       L       11,SAVE11         1072+       B       ALOOPA         1073+W4       EQU       *						
1064+       L       9,SAVE9         1065+       L       10,SAVF10         1066+       L       11,SAVE11         1067+       B       W4         1068+Y4       L       8,SAVE8         1059+       L       9,SAVE9         1070+       L       10,SAVE10         1071+       L       11,SAVE11         1072+       B       ALOOPA         1073+W4       EQU       *	1062+		8, POINTER			
1065+       L       10,SAVF10         1066+       L       11,SAVE11         1067+       B       W4         1068+Y4       L       8,SAVE8         1059+       L       9,SAVE9         1070+       L       10,SAVE10         1071+       L       11,SAVE11         1072+       B       ALOOPA         1073+W4       EQU       *		L				
1066+       L       11,SAVE11         1067+       B       W4         1068+Y4       L       8,SAVE8         1059+       L       9,SAVE9         1070+       L       10,SAVE10         1071+       L       11,SAVE11         1072+       B       ALOOPA         1073+W4       EQU       *						
1067+     B     W4       1068+Y4     L     8,SAVE8       1059+     L     9,SAVE9       1070+     L     10,SAVE10       1071+     L     11,SAVE11       1072+     B     ALOOPA       1073+W4     EQU     *						
1068+Y4       L       8,SAVE8         1059+       L       9,SAVE9         1070+       L       10,SAVE10         1071+       L       11,SAVE11         1072+       B       ALOOPA         1073+W4       EQU       *						
1069+ L 9,SAVE9 1070+ L 10,SAVE10 1071+ L 11,SAVE11 1072+ B ALDOPA 1073+W4 EQU *						
1070+ L 10,SAVE10 1071+ L 11,SAVE11 1072+ B ALDOPA 1073+W4 EQU *						
1072+ B ALOOPA 1073+W4 EQU *		L.				
1073+W4 EQU *						
					н. Табрата (1996)	
A 4 € T 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1		EQU			·	
			사는 박용주변호원호			

1075 A 5+=F\*1\* 6,=F'1' 1076 A 1077 SNAP ID=2, DCB=DUMPAREA, PDATA=REGS 1078 +CNOP 0,4 1079+ BAL 1, THB0005 BRANCH AROUND PARAM LIST 1080+ 00 AL1(2) ID NUMBER 1081+ DC ALL()) 1082+ DC AL1(130) OPTION FLAGS 10834 DC AL1(32) OPTION FLAGS 1084+ DC A(DUMPAREA) DCB ADDRESS 1085+ DC A(O) TCB ADDRESS A(O) ADDRESS OF SNAP-SHOT LIST 1086+ DC 1087+JHB0005 05 0H 1088+ SVC 51 1089 \* . 1090 00 WHILE=A, WREGA=8, WOP=LT, WLITB=10; LABEL=LOOPB 1091+L00P8 EQU 潄 10,SAVE10 1092+ ST 1093+ ST 11, SAVE11 1094+ LR 10,8 1095+ LA 11,0 1096+ A 11,=F\*10\* .5 1097+ CR 10,11 1098+ 8NL Τ6 1099 +Ł 10, SAVE10 1100 +11+SAVE11 L 1101+ 8 ٧6 1102+T6 10, SAVE10 L 11, SAVE11 1103+ Ł 1104+ В AL00P8 1105+V6 EQU \* 1106 8.=F\*1\* Α 9,=F!1! 1107 A 1108 ENDDO LABEL=LOOPB 1109 +8 L0098 1110+CLOOPB 8,SAVE8 Ĺ 1111+ L 9, SAV 59 1112+BL000B 10, SAVE10 L 1113+ 11, SAVE11 L 1114+ALOOPB EQU 20 1115 \* 1116 SNAP ID=3, DCB=DUMPAREA, PDATA=REGS CNOP 1117+ 0,4 1118 +BAL 1, IHB0008 BRANCH AROUND PARAM LIST DC 11194 AL1(3) ID NUMBER 1120+ 00 AL1(0) 1121+ DC ALI(130) OPTION FLAGS 1122+ DC AL1(32) OPTION FLAGS A (DUMPAREA) DCB ADURESS 1123+ 0C 1124+ DC A(O) TCB ADDRESS 1125+ DC A(O) ADDRESS OF SNAP-SHOT LIST 1126+TH80008 DS ОH 1127+ SVC 51 1128 14,=F\*1\* Δ 1129 A. 15,=F\*1\* 1130 ENDDO LABEL=LOOPA 1131+ 8 LOOPA 1132+CL00PA 8,SAVE8 L. 1133+ L 9,SAVE9 1134+8L00PA L 10, SAVE10 1135+ L 11, SAVE11 1136+ALOOPA 600 źc 1137 \* 1138 \* 1139 CASE LABEL=TESTA, REGA=4, OPRATOR=EQ, REGB=15 1140+ sr 10, SAVE10 1141+ S٢ 11,SAVE11

1142+	L۹.	10,4
1143+	Ľ٦	11,15
1144+	CR	10,11
1145+	BNE	810
1146+	B	C10
1147*810	L ·	10, SAVE10
1148+	L	11, SAVE11
1149+	8	TESTA
-		•
1150+010	L	10, SAVE10
1151+	L.	11, SAVE11
1152		LA 4,0
1153		LA 5,0
1154		14 6,3
1155		LA 8,0
1156		LA 9,0
1157		LA 14,0
1158		LA 15,0
1159		SNAP ID=4, DCB=DUMPAREA, PDATA=REGS
1160+	CNOP	0+4
1161+	BAL	1.IHBOO11 BRANCH ARDUND PARAM LIST
1162+	DC	ALI(4) ID NUMBER
1163+	DC	AL1(0)
1164*	DC	ALI(130) OPTION FLAGS
1165+ -	00	ALI(32) OPTION FLAGS
1166+	DC	ALDUMPAREAT DCB ADURESS
1167+	DC .	A(O) TCB ADDRESS
11684	00	A(0) ADDRESS OF SNAP-SHOT, LIST
1169+1H80011	DS	OH
1170+	SVC	51
1171	EVOCA	
1172+TESTA	EQU	*
1173 *	000	<b>*</b>
1174 *		
1176	C A C C	I ADDI YCOTO I YTA IO ODDITOD NE DECO IO
1175	CASE	LABEL=TESTB,LITA=0,DPRATON=NE,REGB=12
1176+	ST	10, SAVE 10
1176+ 1177+	S T S T	10,SAVE10 11,SAVE11
1176+ 1177+ 1178+	ST ST LA	10,SAVE10 11,SAVE11 10,0
1176+ 1177+ 1178+ 1179+	ST ST LA A	10, SAVE10 11, SAVE11 10, 0 10, = F*O*
1176+ 1177+ 1178+ 1179+ 1180+	ST ST LA LR	10, SAVE10 11, SAVE11 10, 0 10, = F*O* 11, 12
1176+ 1177+ 1178+ 1179+ 1180+ 1181+	ST ST LA LR CR	10, SAVE10 11, SAVE11 10,0 10,=F*0* 11,12 10,11
1176+ 1177+ 1178+ 1179+ 1180+ 1181+ 1182+	ST LA A LR CR BE	10, SAVE10 11, SAVE11 10, 0 10, = F*O* 11, 12 10, 11 B13
1176+ 1177+ 1178+ 1179+ 1180+ 1181+ 1182+ 1183+	ST LA LR CR BE B	10, SAVE10 11, SAVE11 10,0 10,=F*0* 11,12 10,11 B13 C13
1176+ 1177+ 1178+ 1179+ 1180+ 1181+ 1182+ 1183+ 1184+813	ST ST LA LR CR BE B L	10, SAVE10 11, SAVE11 10,0 10,=F*O* 11,12 10,11 B13 C13 10,SAVE10
1176+ 1177+ 1178+ 1179+ 1180+ 1181+ 1182+ 1183+ 1184+813 1185+	ST LA A LR CR BE B L	10, SAVE10 11, SAVE11 10,0 10,=F*O* 11,12 10,11 B13 C13 10, SAVE10 11, SAVE11
1176+ 1177+ 1178+ 1179+ 1180+ 1181+ 1182+ 1183+ 1184+813 1185+ 1186+	ST LA A LR CR BE L L B	10, SAVE10 11, SAVE11 10,0 10,=F*O* 11,12 10,11 B13 C13 10, SAVE10 11, SAVE10 11, SAVE11 TESTB
1176+ 1177+ 1178+ 1179+ 1180+ 1181+ 1182+ 1183+ 1183+ 1185+ 1186+ 1187*CI3	ST LA A LR CR BE L L B L	10, SAVE10 11, SAVE11 10,0 10,=F*O* 11,12 10,11 B13 C13 10, SAVE10 11, SAVE10 11, SAVE11 TESTB 10, SAVE10
1176+ 1177+ 1178+ 1179+ 1180+ 1181+ 1182+ 1183+ 1185+ 1185+ 1186+ 1187+CI3 1183+	ST LA A LR CR BE L L B	10, SAVE10 11, SAVE11 10, 0 10, =F*O* 11, 12 10, 11 B13 C13 10, SAVE10 11, SAVE11 TESTB 10, SAVE10 11, SAVE10 11, SAVE11
1176+ 1177+ 1178+ 1179+ 1180+ 1181+ 1182+ 1183+ 1185+ 1185+ 1186+ 1187+CI3 1183+ 1189	ST LA A LR CR BE L L B L	10, SAVE10 11, SAVE11 10,0 10,=F*O* 11,12 10,11 B13 C13 10, SAVE10 11, SAVE10 11, SAVE11 TESTB 10, SAVE10 11, SAVE11 LA 4,5
1176+ 1177+ 1178+ 1179+ 1180+ 1181+ 1182+ 1183+ 1185+ 1185+ 1186+ 1187+CI3 1183+ 1189	ST LA A LR CR BE L L B L	10, SAVE10 11, SAVE11 10,0 10,=F*O* 11,12 10,11 B13 C13 10, SAVE10 11, SAVE10 11, SAVE11 TESTB 10, SAVE10 11, SAVE11 LA 4,5 LA 5,5
1176+ 1177+ 1178+ 1179+ 1180+ 1181+ 1182+ 1183+ 1185+ 1185+ 1186+ 1187+CI3 1183+ 1189 1190 1191	ST LA A LR CR BE L L B L	10, SAVE10 11, SAVE11 10,0 10,=F*O* 11,12 10,11 B13 C13 10, SAVE10 11, SAVE10 11, SAVE11 TESTB 10, SAVE10 11, SAVE11 LA 4,5
1176+ 1177+ 1178+ 1179+ 1180+ 1181+ 1182+ 1183+ 1184+813 1185+ 1186+ 1186+ 1187+C13 1183+ 1189 1190 1191 1192	ST LA A LR CR BE L L B L	10, SAVE10 11, SAVE11 10,0 10,=F*O* 11,12 10,11 B13 C13 10, SAVE10 11, SAVE10 11, SAVE11 TESTB 10, SAVE10 11, SAVE11 LA 4,5 LA 5,5
1176+ 1177+ 1178+ 1179+ 1180+ 1181+ 1182+ 1183+ 1185+ 1186+ 1186+ 1186+ 1183+ 1183+ 1183+ 1189 1190 1190 1191 1192 1193	ST LA A LR CR BE L L B L	10, SAVE10 11, SAVE11 10,0 10,=F*O* 11,12 10,11 B13 C13 10, SAVE10 11, SAVE10 11, SAVE11 TESTB 10, SAVE10 11, SAVE11 LA 4,5 LA 5,5 LA 6,5
1176+ 1177+ 1178+ 1179+ 1180+ 1181+ 1182+ 1183+ 1185+ 1186+ 1186+ 1183+ 1183+ 1183+ 1183+ 1189 1190 1191 1192 1193 1194	ST LA A LR CR BE L L B L	10, SAVE10 11, SAVE11 10, 0 10, = F * 0 * 11, 12 10, 11 B13 C13 10, SAVE10 11, SAVE10 11, SAVE11 TESTB 10, SAVE10 11, SAVE11 LA 4, 5 LA 5, 5 LA 6, 5 LA 8, 5 LA 9, 5 SNAP 10=5, DCB=0J4PAPEA, PDATA=REGS
1176+ 1177+ 1178+ 1179+ 1180+ 1181+ 1182+ 1183+ 1184+813 1185+ 1186+ 1186+ 1187+C13 1183+ 1189 1190 1191 1192 1193 1194 1195+	ST ST LA A LR CR BE B L L L L CNOP	10, SAVE10 11, SAVE11 10, 0 10, = F * 0 * 11, 12 10, 11 B13 C13 10, SAVE10 11, SAVE10 11, SAVE10 11, SAVE10 11, SAVE10 11, SAVE11 LA 4, 5 LA 5, 5 LA 5, 5 LA 6, 5 LA 8, 5 LA 9, 5 SNAP ID=5, DCB=0JMPAPEA, PDATA=REGS 0, 4
1176 + 1177 + 1178 + 1179 + 1180 + 1181 + 1182 + 1183 + 1183 + 1185 + 1185 + 1186 + 1187 + C13 1183 + 1189 1190 1191 1192 1193 1194 1195 + 1196 +	ST ST LA A LR CR BE B L L B L L CNOP BAL	10, SAVE10 11, SAVE11 10, 0 10, = F * 0 * 11, 12 10, 11 B13 C13 10, SAVE10 11, SAVE10 11, SAVE10 11, SAVE10 11, SAVE10 11, SAVE11 LA 4, 5 LA 5, 5 LA 5, 5 LA 6, 5 LA 8, 5 LA 9, 5 SNAP ID=5, DCB=0JMPAPEA, PDATA=REGS 0, 4
1176 + 1177 + 1178 + 1179 + 1180 + 1181 + 1182 + 1183 + 1183 + 1183 + 1185 + 1186 + 1187 + C13 1183 + 1189 1190 1191 1192 1193 1194 1195 + 1196 + 1197 +	ST ST LA A LR CR BE B L L L L CNOP	10, SAVE10 11, SAVE11 10, 0 10, = F * 0 * 11, 12 10, 11 B13 C13 10, SAVE10 11, SAVE10 11, SAVE11 TESTB 10, SAVE10 11, SAVE11 LA 4, 5 LA 5, 5 LA 6, 5 LA 8, 5 LA 9, 5 SNAP 10=5, DCB=0J4PAPEA, PDATA=REGS
1176 + 1177 + 1178 + 1179 + 1180 + 1181 + 1182 + 1183 + 1183 + 1185 + 1186 + 1187 + C13 1183 + 1190 1190 1191 1192 1193 1194 1195 + 1196 + 1197 + 1198 +	ST ST LA A LR CR BE B L L L CNOP BAL DC DC	10, SAVE10 11, SAVE11 10, 0 10, =F*0* 11, 12 10, 11 B13 C13 10, SAVE10 11, SAVE10 11, SAVE11 TESTB 10, SAVE10 11, SAVE11 LA 4, 5 LA 5, 5 LA 6, 5 LA 6, 5 LA 8, 5 LA 9, 5 SNAP ID=5, DCB=DJMPAPEA, PDATA=REGS 0,4 1, IHB0014 BRANCH ARDUND PARAM LIST AL1(5) ID NUMBER AL1(0)
1176 + 1177 + 1178 + 1179 + 1180 + 1181 + 1182 + 1183 + 1183 + 1185 + 1185 + 1186 + 1187 + C13 1183 + 1190 1190 1191 1192 1193 1194 1195 + 1195 + 1196 + 1197 + 1198 + 1197 + 1199 + 1197 + 1197 + 1198 + 1197 + 1198 + 1197 + 1197 + 1198 + 1197 + 1197 + 1198 + 1197 + 11	ST ST LA A LR CR BE B L L L CNOP BAL DC DC DC	10, SAVE10 11, SAVE11 10, 0 10, =F*0* 11, 12 10, 11 B13 C13 10, SAVE10 11, SAVE10 11, SAVE11 TESTB 10, SAVE10 11, SAVE11 LA 4, 5 LA 5, 5 LA 6, 5 LA 6, 5 LA 8, 5 LA 9, 5 SNAP ID=5, DCB=DJMPAPEA, PDATA=REGS 0,4 1, IHB0014 BRANCH ARDUND PARAM LIST AL1(5) ID NUMBER AL1(0) AL1(130) DPTION FLAGS
1176 + 1177 + 1178 + 1179 + 1180 + 1181 + 1182 + 1183 + 1183 + 1185 + 1185 + 1186 + 1187 + C13 1183 + 1190 1190 1191 1192 1193 1194 1195 + 1195 + 1195 + 1196 + 1197 + 1198 + 1197 + 1190 + 1200 +	ST ST LA A LR CR BE B L L L CNOP BAL DC DC DC DC	10, SAVE10 11, SAVE11 10, 0 10, =F*0* 11, 12 10, 11 B13 C13 10, SAVE10 11, SAVE10 11, SAVE11 LA 4, 5 LA 5, 5 LA 6, 5 LA 8, 5 LA 8, 5 LA 9, 5 SNAP ID=5, DCB=DJMPAPEA, PDATA=REGS 0,4 1, IHB0014 BRANCH ARDUND PARAM LIST AL1(5) ID NUMBER AL1(0) AL1(130) OPTION FLAGS
1176 + 1177 + 1178 + 1179 + 1180 + 1181 + 1182 + 1183 + 1183 + 1185 + 1186 + 1187 + C13 1183 + 1190 1190 1191 1192 1193 1194 1195 + 1195 + 1195 + 1197 + 1198 + 1197 + 1193 + 1192 + 1192 + 1193 + 1192 + 1193 + 1192 + 1192 + 1192 + 1192 + 1193 + 1192 + 1193 + 1192 + 1193 + 11	ST ST LA A LR CR BE B L L L CNOP BAL DC DC DC DC DC DC	10, SAVE10 11, SAVE11 10, 0 10, =F*0* 11, 12 10, 11 B13 C13 10, SAVE10 11, SAVE10 11, SAVE11 TESTB 10, SAVE10 11, SAVE11 LA 4, 5 LA 5, 5 LA 6, 5 LA 6, 5 LA 8, 5 LA 9, 5 SNAP ID=5, DCB=DJMPAPEA, PDATA=REGS 0,4 1, IHB0014 BRANCH ARDUND PARAM LIST AL1(5) ID NUMBER AL1(0) AL1(130) DPTION FLAGS
1176 + 1177 + 1178 + 1179 + 1180 + 1181 + 1182 + 1183 + 1183 + 1185 + 1186 + 1187 + C13 1183 + 1189 1190 1191 1192 1193 1194 1195 + 1195 + 1196 + 1197 + 1200 + 1201 + 1202 +	ST ST LA A LR CR BE B L L L B L L CNOP BAL DC DC DC DC DC DC DC	10, SAVE10 11, SAVE11 10, 0 10, =F*0* 11, 12 10, 11 B13 C13 10, SAVE10 11, SAVE11 TESTB 10, SAVE10 11, SAVE11 LA 4, 5 LA 5, 5 LA 6, 5 LA 8, 5 LA 9, 5 SNAP ID=5, DCB=DJMPAPEA, PDATA=REGS 0,4 1, IHB0014 BRANCH ARDUND PARAM LIST AL1(5) ID NUMBER AL1(130) DPTION FLAGS AL0 MPAREA) DCB ADDRESS A(0) TCB ADDRESS
1176 + 1177 + 1178 + 1179 + 1180 + 1181 + 1182 + 1183 + 1183 + 1185 + 1186 + 1187 + C13 1183 + 1190 + 1190 + 1192 + 1195 + 1195 + 1196 + 1197 + 1198 + 1197 + 1201 + 1201 + 1202 + 1203 +	ST ST LA A LR CR BE B L L L B L L CNOP BAL DC DC DC DC DC DC DC DC	10, SAVE10 11, SAVE11 10, 0 10, =F+0+ 11, 12 10, 11 B13 C13 10, SAVE10 11, SAVE10 11, SAVE11 LA 4, 5 LA 5, 5 LA 6, 5 LA 6, 5 LA 8, 5 LA 9, 5 SNAP ID=5, DCB=DJMPAPEA, PDATA=REGS 0,4 1, IHB0014 BRANCH ARDUND PARAM LIST AL1(5) ID NUMBER AL1(130) OPTION FLAGS AL(132) OPTION FLAGS A(0) MDDPESS OF SNAP-SHOT LIST
1176+ 1177+ 1178+ 1179+ 1180+ 1181+ 1182+ 1183+ 1183+ 1185+ 1186+ 1187+C13 1183+ 1190 1190 1191 1192 1193 1194+ 1195+ 1196+ 1197+ 1198+ 1197+ 1201+ 1202+ 1204+TH80014	ST ST LA A LR CR BE B L L B L L CNOP BAL DC DC DC DC DC DC DC DC DC DC DC DC DC	10, SAVE10 11, SAVE11 10, 0 10, =F*0* 11, 12 10, 11 B13 C13 10, SAVE10 11, SAVE11 TESTB 10, SAVE10 11, SAVE11 LA 4, 5 LA 5, 5 LA 6, 5 LA 6, 5 LA 6, 5 LA 8, 5 SNAP ID=5, DCB=DJMPAPEA, PDATA=REGS 0,4 1, IHB0014 BRANCH ARDUND PARAM LIST AL1(5) ID NUMBER AL1(3) OPTION FLAGS A(0) MDDPESS OF SNAP-SHOT LIST 04
1176+ 1177+ 1178+ 1179+ 1180+ 1181+ 1182+ 1183+ 1185+ 1185+ 1186+ 1187+C13 1183+ 1189 1190 1191 1192 1193 1194+ 1195+ 1195+ 1196+ 1197+ 1201+ 1202+ 1202+ 1204+1H80014 1205+	ST ST LA A LR CR BE B L L L B L L CNOP BAL DC DC DC DC DC DC DC DC	10, SAVE10 11, SAVE11 10, 0 10, =F+0+ 11, 12 10, 11 B13 C13 10, SAVE10 11, SAVE10 11, SAVE11 LA 4, 5 LA 5, 5 LA 6, 5 LA 6, 5 LA 8, 5 LA 9, 5 SNAP ID=5, DCB=DJMPAPEA, PDATA=REGS 0,4 1, IHB0014 BRANCH ARDUND PARAM LIST AL1(5) ID NUMBER AL1(130) OPTION FLAGS AL(132) OPTION FLAGS A(0) MDDPESS OF SNAP-SHOT LIST
1176+ 1177+ 1178+ 1179+ 1180+ 1181+ 1182+ 1183+ 1183+ 1185+ 1186+ 1187+C13 1183+ 1187+C13 1183+ 1197+ 1192 1193 1194 1195+ 1196+ 1197+ 1198+ 1197+ 1201+ 1202+ 1203+ 1204+TH80014 1205+ 1206 *	ST ST LA A LR CR BE B L L L B L L L CNOP BAL DC DC DC DC DC DC DC S S V C	10, SAVE10 11, SAVE11 10, 0 10, = F'O' 11, 12 10, 11 B13 C13 10, SAVE10 11, SAVE10 11, SAVE11 LA 4, 5 LA 5, 5 LA 6, 5 LA 6, 5 LA 8, 5 LA 9, 5 SNAP ID=5, 0CB=DJMPAPEA, PDATA=REGS 0,4 1, IHB0014 BRANCH ARDUND PARAM LIST AL1(5) ID NUMBER AL1(3) DPTION FLAGS AL1(32) DPTION FLAGS A(0) MDDPESS OF SNAP-SHOT LIST 04 51
1176+ 1177+ 1178+ 1179+ 1180+ 1181+ 1182+ 1183+ 1185+ 1185+ 1186+ 1187+C13 1183+ 1189 1190 1191 1192 1193 1194+ 1195+ 1195+ 1196+ 1197+ 1201+ 1202+ 1202+ 1204+1H80014 1205+	ST ST LA A LR CR BE B L L B L L CNOP BAL DC DC DC DC DC DC DC DC DC DC DC DC DC	10, SAVE10 11, SAVE11 10, 0 10, =F'O' 11, 12 10, 11 B13 C13 10, SAVE10 11, SAVE10 11, SAVE11 LA 4, 5 LA 5, 5 LA 6, 5 LA 8, 5 LA 9, 5 SNAP ID=5, 0CB=0JMPAPEA, PDATA=REGS 0,4 1, HB0014 BRANCH ARDUND PARAM LIST AL1(5) ID NUMBER AL1(5) ID NUMBER AL1(130) DPTION FLAGS A(0)MPAREA) DCB ADDRESS A(0) ADDRESS OF SNAP-SHOT LIST 04

1203 + 1209 + 1210 + 1211 + 1212 + 1212 + 1213 + 1213 + 1215 + 1215 + 1215 + 1215 + 1220 + 1220 + 1221 + 1220 + 1220 + 1220 + 1220 + 1220 + 1220 + 1220 + 1220 + 1220 + 1220 + 1220 + 1220 + 1220 + 1220 + 1220 + 1220 + 1230 + 1240 + 1250 +	STTTRA R LTATATAT L L L STTTTT SLSLSLSL BCBB BCBLAL SLATATAT L L L B	8, SAV58 9, SAV59 10, SAV59 10, SAV510 11, SAV511 10, 4 11, 0 11, =F + 100 9, 4 8, POINTER 9, STACK(8) 8, 4(3) 10, STACK(8) 8, 4(3) 11, STACK(8) 8, 4(3) 11, STACK(8) 8, 4(3) 8, 9, SAV59 10, SAV50 8, =F +4 10, STACK(8) 8, =F +4 10, STACK(8) 8, =F +4 10, STACK(8) 8, POINTER 9, ZER0 X15 10, 11 Y15 10, 9 8, POINTER 9, STACK(8) 8, 4(8) 11, STACK(8) 11, STACK(8	
1257+ 1258+ 1259+ 1260+ 1261+¥15 1262+ 1263+	և Լ Ց. Լ Լ	9,54VE9 10,54VE10 11,54VE11 W15 8,54VE8 9,54VE9 10,54VE9	
1264+ 1265+ 1266+W15 1267 1268 1269 1270+ 1271+CEEE 1272+ 1273+BEEE 1274+	L 8 590 1 8 1 1 1 1		=F* <u>1</u> * =F* <u>1</u> *

1275+AEEE	EQU	ht.
1276 *		
1277		SNAP ID=6,DC8=DJMPAREA,PDATA≅REGS
1278+	CNOP	0,4
	BAL	1, IHBOO17 BRANCH ARDUND PARAM LIST
1279+		
1289+	DC	ALI(6) ID NUMBER
1281+	DC	AL1(0)
1282+	DC	ALI(130) OPTION FLAGS
1283+	00	AL1(32) OPTION FLAGS
1284+	DC	A (DUMPAREA) DC8 ADDRESS
1285+	00	A(D) TCB ADDRESS
1286+	00	A(O) ADDRESS OF SNAP-SHOT LIST
1287+IH80017	D \$	он .
1288+	SVC	51
1289	ENDCAS	SE LABEL=TESTB,OPTION=2
1290+TESTB	EQU	*
1291 *		
1292 *		
	011.0	
	SNAP	
1294+	CNOP	0,4
1295+	BAL	1, IH30019 BRANCH ARDUND PARAM LIST
1296+	DC	AL1(7) ID NUMBER
1297+	00	AL1(0)
1298+	DC	ALI(130) OPTION FLAGS
1299+	00	AL1(32) OPTION FLAGS
1300+	DC	A(DUMPAREA) DCB ADDRESS .
1301+	DC .	A(0) TCB ADDRESS .
1302+	DC	A(O) ADDRESS OF SNAP-SHOT LIST
1303+IH80019	05	OH
1304+	SVC	51
	3*0	
1305 *		· · · ·
1306 *		
1307	EXIT	
1308+	L	13,4(,13) POP UP SAVE AREA
1309+	LM.	14,12,12(13) RESTORE REGISTERS
1310+	NV I	
		12(13),X*FF* FLAG EXIT
1311+	82 .	14 RETURN
1312	CLOSE	DUMPAREA
1313+	CNOP	0,4 ALIGN LIST TO FULLWORD
1314+	BAL	1,*+8 LOAD REGI W/LIST ADDR
1315+	ÐC	AL1(128) OPTION BYTE
1316+	ĐC	AL3(DUMPAREA) DCB ADDRESS
1317+		
13114	SVC	20 ISSUE CLOSE SVC
1318 SAVEAREA	DC .	184(0)
1319	DS	0 F
1320 ZERO	DC	1 F* * O *
1321 SAVE8	DS	14
1322 SAVE9	DS	10
1323 SAVELO	DS	16
1324 SAVE11	DS .	1 F
1325 POINTER	DC	1 = 0 +
1326 STACK	DS	100F
1327 DUMPAREA	008	DONAME=TEAM, DSDRG=PS, RECEM=VBA,
	-	MACRF=W, BLKSIZE=882, LRECL=125
		The Commander State Gozy the Ferty St
1100.1		
1329+*		DATA CONTROL BLOCK
1330+*		
1331+DUMPAREA	DC	OFTO* ORIGIN ON WORD BOUNDARY
		•
1333+*		DIRECT ACCESS DEVICE INTERFACE
1335+	00	BL16*0* FD40, DVT8L
1336+	DC	
8.0.10 C	ما 1/1	A(O) KEYLE, DEVT, TRBAL

.

183

Х

1338+*		COMMON ACCESS METHOD INTERFACE
1340+	DC	ALI(0) BUFNO
1341+	00	AL3(1) BUFCB
1342+	0Č	AL2(0) BUFL
1343+	DC	8L2*01000000000000000 DSDRG
1344+	DC	A(1) IOBAD
73441		ALLI IODAU
1346+*		FOUNDATION EXTENSION
1348+	DC	BL1'00000001 BFTEK, BFLN, HIARCHY
1349+	DC	AL3(1) ECDAD
1350+	DC	BL1+01010100+ RECEM
1351+	DC	AL3(0) EXLST
1353+*		FOUNDATION BLOCK
1355+	οc	CL8*TEAM* DDNAME
1356+	DC	BL1*00000010* 0FLGS
1357+	DC	BL1 0000000 IFLG
1358+	DC 1	BL2*000000000000000 MACR
1360+*		BSAM-BPAM-QSAM INTERFACE
1362+	DC -	BL1*00000000* RERÍ
1363+	DČ	AL3(1) CHECK, GERR, PERR
1364+	00	A(1) SYNAD
1365+	00	H'0' CIND1, CIND2
1366+	DC	AL2(882) BLKSIZE
1367+	- DC	F'O' WCPO, WCPL, OFFSR, OFFSW
1368*	DC	A(1) 1084
1369+	0C	ALI(O) NCP
1370+	00	AL3(1) EOBR, EOBAO
1372+*		BSAM-BPAM INTERFACE
1374+	00	A(1) EOBW
1375+	DC	H*O* DIRCT
1376+	20	AL2(125) LRECL
1377+	DC	A(1) CNTPL, NOTE, POINT
1378	END	
1379		∞F*1*
1380		= F 1 5 1
1381		≠F*41 ·
1382		=F*10*
1383		#F°0'
1384		≖F*100°

#### BIBLIOGRAPHY

#### Books

- Chapin, Ned, <u>360/370</u> Programming in <u>Assembly</u> Language, New York, McGraw-Hill Book Co., <u>1968</u>.
- Dahl, O. J. and others, <u>Structured Programming</u>, London, Academic Press, Inc., Ltd., 1972.
- Hannula, Reino, <u>Computers and Programming</u>: <u>A System/</u> <u>360-370 Assembler Language Approach</u>, Boston, Ma., <u>Houghton Mifflin Co., 1974</u>.
- Organick, Elliott I., The Multics System: An Examination of Its Structure, Cambridge, Ma., The MIT Press, 1972.
- Weinberg, G., The Psychology of Computer Programming, New York, Von Nostrand Reinhold Co., 1971.
- Wirth, Miklaus, Systematic Programming: An Introduction, Englewood Cliffs, N.J., Prentice-Hall, Inc., 1973.

#### Articles

- Baker, F. T., "System Quality Through Structured Programming," Proc. FJCC, (1972), 339-343.
  - and H. D. Mills, "Chief Programmer Teams," Datamation, XIX (December, 1973), 58-61.
- Bauer, F. L., "Software and Software Engineering," SIAM Review, XV (April, 1973), 469-480.
- Benson, Jeoffrey, "Structured Programming Techniques," <u>Record of the 1973 IEEE Symposium on Computer Software</u> <u>Reliability, New York, (April, 1973), 143-147.</u>
- Bohm, C. and G. Jacopini, "Flow Diagrams, Turing Machines and Languages with only Two Formation Rules," <u>Communications of the ACM</u>, IX (May, 1966), 366-371.
- Denning, P. J., "Is It Not Time to Define 'Structured Programming'?" <u>SIGPLAN</u> Notices, (February, 1974), 6-7.

Dijkstra, E. W., "The Humble Programmer," <u>Communications of</u> the ACM, XV (October, 1972), 859-866.

, "Structured Programming," Software Engineering, Report on a Conference Sponsored by the NATO Science Committee, Rome, Italy, (October, 1969), 84-88.

, "The Structure of the 'THE' Multiprogramming System," <u>Communications</u> of the <u>ACM</u>, XI (May, 1968), 341-356.

, "GOTO Statement Considered Harmful," Communications of the ACM, XI (March, 1968), 147-148.

, "Complexity Controlled by Hierarchecal Ordering of Function and Variability," <u>Software</u> Engineering, Report on a Conference Sponsored by the NATO Science Committee, Garmish, Germany, (October, 1968), 181-185.

- Donaldson, James, "Structured Programming," Datamation, XIX (December, 1973), 52-54.
- Fisher, David, "A Survey of Control Structures in Programming Languages," <u>SIGPLAN</u> <u>Notices</u>, (November, 1972), 1-13.
- Hopkins, Martin, "A Case for the GOTO," <u>SIGPLAN</u> <u>Notices</u>, (November, 1972), 59-62.
- Kent, William, "Assembler-Language Macroprogramming: A Tutorial Oriented Toward the IBM 360," <u>Computing</u> Surveys, (December, 1969), 183-196.
- Leavenworth, B. M., "Programming With(out) the GOTO," SIGPLAN Notices, (November, 1972), 54-58.
- McCracken, Daniel, "Revolution in Programming: An Overview," Datamation, XIX (December, 1973), 50-52.
- Miller, E. and G. Landamood, "Structured Programming: Top-Down Approach," <u>Datamation</u>, XIX (December, 1973), 55-57.
- Mills, H. D., "On the Development of Large Reliable Programs," <u>Record of the 1973 IEEE Symposium on</u> <u>Computer Software Reliability</u>, New York, (May, 1973), 155-159.

- Parnas, D. L., "Information Distribution Aspects of Design Methodology," Information Processing 71, North-Holland Publishing Co., 339-344.
- Peterson, W. W., et al, "On the Capabilities of While, Repeat, and Exit Statements," <u>Communications of the</u> ACM, XVI (August, 1973), 503-512.
- Rosen, Saul, "Electronic Computers: A Historical Survey," Computing Surveys, (March, 1969), 7-36.
- Tenny, Ted, "Structured Programming in FORTRAN," Datamation, XX (July, 1974), 110-115.
- Wulf, William, "A Case Against the GOTO," <u>SIGPLAN</u> Notices, (November, 1972), 63-69.

#### Reports

- IBM System/360 Operating System Assembler Language, IBM Form No. GC28-6514-8.
- IBM System/360 Operating System Principles of Operation, IBM Form No. GA22-6821-8.
- Improved Programming Technologies Management Overview, IBM Corporation, Data Processing Division, Systems Marketing, Installation Productivity Programs Department, August, 1973.
- Kessler, M. M., <u>\*CONCEPTS\*</u> Report 4, OS/360 Assembly Language Block Structured Programming Macros, IBM.
- Mills, H. D., <u>Chief Programmer Teams</u>: <u>Principles and</u> <u>Procedures</u>, <u>Report No. FSC 71-5108</u>, <u>IBM Federal</u> <u>Systems Division</u>, <u>Gaithersburg</u>, <u>Maryland</u>.
  - , How to Write Correct Programs and Know It, Report No. FSC 73-5998, IBM, Gaithersburg, Maryland, December, 1972.
- , Mathematical Foundations for Structured <u>Programming</u>, Report No. FSC 72-6012, IBM, Gaithersburg, Maryland, February, 1972.
- Special Interest Group on Programming Languages, <u>Special</u> <u>Issue on Control Structures in Programming Languages</u>, New York, New York, Association for Computing Machinery. Vol. VII (November, 1972).